UNIVERSIDAD NACIONAL AUTč NOMA DE M£ XICO

FACULTAD DE CIENCIAS

LOS SHIFTS Y AUTč MATAS DE ĆRBOLES

T E S I S

QUE PARA OBTENER EL TčTULO DE:

MATEMĆTICO

P R E S E N T A :

PABLO GONZĆLEZ MOCTEZUMA

DR. RICARDO Gč MEZ AčZA

Cd. Universitaria, D.F. 2014

1.-Datos del alumno
González
Moctezuma
Pablo
Teléfono: 55347833
Universidad Nacional Autónoma de México, Facultad de Ciencias
Carrera: Matemáticas
Número de cuenta: 40900182-4

2.-Datos del tutor
Dr.
Ricardo
Gómez
Aíza

3.- Datos sinodal 1
Dr.
Pedro Eduardo
Miramontes
Vidal

4.- Datos sinodal 2
Dr.
Juan José
Montellano
Ballesteros

5.- Datos sinodal 3
Dr.
José de Jesús
Galaviz
Casas

6.- Datos sinodal 4
Dr.
Sergio
Rajsbaum
Gorodeski

7.-Datos de trabajo escrito
Los shifts y autómatas de árboles
114 páginas
2014

Agradecimientos

A Luz Elena y Antonio por darme el regalo de la vida y demostrarme que es maravillosa. A mi abuelo Jorge por su amor al mundo y a su familia, a mi abuela Elena por quererme y cuidarme con todo su corazón. A Tita y Pepe por apoyarme desde allá arriba con una sonrisa. A mis hermanos Pedro y Francisco, amigos de infancia y colegas de pasiones. A mi novia Mariana, compañera de museos, inquietudes, corazón y vida. A Fernando González R. por sus siempre atentas preguntas, su interés y el gran apoyo en mi estudio. A mis queridísimos amigos Santi, Max, Jano, Chucho, Diego Heredia, Betito, Ernesto Vásquez, Rígel J, Alancito, James G, Tocayo del Cueto, Lore, Marina, Liz Hernández, Santi Filo, Tania, Huguito, Rodrigo y Andrea, Ivan Paz, Fabian, El macizo, Renato, Andrew, Arpón, Paco, Ivan, Jeronimus, Octavio Piano, Ricardo San, Irma San, Jero mi primo por hacer de esa etapa la más divertida de mi vida. Al equipo de los mejores tacos del mundo, Don Manuel y Doña Rosa, por ayudarme a mantenerme sano y contento todos los días de estudio. A mi estupendo asesor Ricardo Gómez por todas las tardes de trabajo e iluminación y por este trabajo que también es suyo. A mis sinodales J.J Montellano, P. Miramontes, S. Rajsbaum y J. Galaviz, por su apoyo durante esta etapa tan importante. A los grandes profesores E. Vásquez, H.Argueta, H. Méndez, J.A Amor, R. Barbachano, Lascu, Carlos Prieto, P. Sollich, Wilfred Meyer, Ruth, Chucho y muchos otros, ejemplo de humanismo y nobleza A mis tíos Jorge, Lorena, Lourdes, Roberto, Ana, Francisco y Mónica por todo el apoyo y el siempre caluroso recibimiento en su casa. A Luis Felipe por enseñarme todo lo que se puede lograr con dedicación y esfuerzo.

Prefacio

Decidí estudiar matemáticas porque creo que es el lenguaje de la naturaleza y si uno puede entender su origen, puede entenderse a sí mismo. Ellas me presentaron a mis grandes amigos, me ayudaron a entender que el mundo es tremendamente complejo y la vida muy sencilla, que cuando uno crea que ha acabado de entender tiene que empezar a escoger (Gödel), que transformarse es emocionante e inevitable, que hay muchas cosas que existen aunque no se tenga prueba de ello y que son las más importantes, que la mente es tremendamente poderosa, que siempre se puede encontrar una solución creativa, entre muchas otras. La UNAM me enseñó el respeto, el poder del esfuerzo y la disciplina propia, la libertad y la gran responsabilidad que viene siempre con ella, la diversidad del mundo que es su mayor riqueza, que viajar es vivir pero vivir es mucho más, que tengo muchos hermanos, que llevo en mi sangre la sangre de todos, que cada quien es maravilloso y que el mundo está hecho de cuentos y sueños. La gran ciudad de México con su tremenda riqueza crió a mis abuelos y a mis padres, me enfrentó a mi cultura y me consolidó como un ciudadano del mundo dispuesto a servir a manos llenas.

Con este trabajo cierro todos estos aprendizajes muy contento, muy orgulloso y honrado de terminar una etapa tan maravillosa de la vida. Me puse en contacto con Ricardo porque confié en su tremenda calidad humana y no me equivoqué. A los dos meses de investigar los mares de la dinámica simbólica empecé a hojear artículos sobre el tema. El que escogimos nos llamó la atención por ser una propuesta novedosa que se adentra, más que en un problema, en la construcción de una nueva forma de ver un universo y creo que haberlo conseguido en tan poco tiempo es una lección muy valiosa. Generar una alternativa creativa sí se puede y se consigue con buen ánimo. Una vez escogido el tema nos pusimos a trabajar con constancia y logramos tras unos meses desarrollarlo con claridad. Fue un trabajo efectivo y muy formativo. Me di cuenta que en mi carrera aprendí mucho y que mi personalidad se ha desarrollado para bien en muchos aspectos.

Esta tesis consta de dos partes. La primera es una introducción a los espacios shift vistos

como sucesiones de símbolos de un alfabeto ya sea sobre $\mathbb{N}$ o $\mathbb{Z}$. Los espacios pueden definir a sus puntos a partir de caminos infinitos sobre una gráfica con aristas etiquetadas con letras del alfabeto, a partir del lenguaje de un autómata con ciertas propiedades o por un conjunto de subsucesiones prohibidas de cualquier cardinalidad. Al final de la primera parte hemos incluído una breve introducción a los autómatas por ser objetos que cada día ayudan más al desarrollo de mis disciplinas favoritas: teoría de los lenguajes, sistemas dinámicos y ciencias de la computación. La segunda parte toma los conceptos básicos de la primera y trata de cambiar la presentación de los símbolos en sucesiones por una en árboles. Estos árboles son gráficas verticales que comienzan en un nodo y ramifican en todas las posibles combinaciones que pueden formarse con sufijos sobre ese nodo. La intención del trabajo es dar una buena explicación de la redefinición de los conceptos clásicos en la teoría que utiliza árboles y probar el teorema de descomposición para ambos shifts por árboles y shifts por sucesiones sobre $\mathbb{N}$.

Índice general

Introducción

Antecedentes

La dinámica simbólica nació en un trabajo de Jaques Hadamard que en el año 1898 utilizó letras para partir una variedad y estudiar un flujo sobre ella. En vez de representar con puntos la posición de un objeto bajo el flujo asociaría la letra correspondiente a la parte donde se encontraba en ese tiempo. Esto arroja una sucesión de aquellos símbolos que están enumerando a las partes si se discretiza el tiempo. Desde entonces, de esta idea se derivaron muchas otras utilizando las mismas sucesiones y han sido estudiadas por muchos especialistas en diversas áreas. En el área de geometría diferencial, esta forma de estudio de los flujos resultó particularmente útil en el análisis de los difeomorfismos hiperbólicos. Alrededor de la década de los setenta un grupo de investigación en sistemas dinámicos de Berkeley precidido por Rufus Bowen comenzó a matematizar la teoría y la llamó dinámica simbólica, en la cual se definieron espacios de sucesiones y se descubrieron métricas útiles e invariantes. Poco después el espectro de uso de esta herramienta creció cuando el padre de los códigos Claude Shannon los utilizó para desarrollar formas efectivas de almacenar información. De allí que esta materia se montó a una de las ramas de las matemáticas de mayor crecimiento hoy en día, las ciencias de la computación. Hay muchas otras áreas en las que la dinámica simbólica se ha abierto camino como en la teoría ergódica, la teoría de la simulación, teoría de gráficas y hasta álgebra lineal.

Generalizaciones

La aportación de este trabajo es exponer detalladamente una de las múltiples posibles generalizaciones de los espacios medulares de la dinámica simbólica. Estudiados a profundidad por Nathalie Auburn y Marie-Pierre Béal en los últimos cinco años, los espacios shift de árbol son una forma alternativa de definir el concepto central de espacio shift. Muchas veces en las matemáticas es útil tener varias versiones del mismo objeto, pues puede que una prueba sea mucho mas sencilla utilizando una definición específica. El problema central que se tratará en este trabajo sigue siendo un problema abierto en su versión de sucesiones doblemente infintas (i.e sucesiones sobre $\mathbb{Z}$) y en espacios shift multidimensionales, los cuales no se han podido estudiar cabalmente por el gran abanico de opciones de generailización existentes. El problema de conjugación de espacios shift es muy importante porque nos permite generar una partición del universo de todos los espacios que desde el punto de vista de la teoría son completamente equivalentes. La teoría de los espacios shift de árboles es una extensión de la de los espacios shift de sucesiones unidireccionales sobre $\mathbb{N}$ por lo que todo lo que logremos probar en la extensión es válido en la teoría misma y así, en este trabajo, también estamos probando la decidibilidad y el teorema de descomposición para el caso de espacios de sucesiones unidireccionales. Si el lector ya está familiarizado con la teoría de los espacios shift recomendamos que vaya directamente al capítulo 4 donde se presentan los espacios shift de árbol y una clase de autómatas que se utiliza para probar la decidibilidad de la descomposición. Si el lector no está familiarizado con los conceptos hemos tratado de hacer una presentación clara y completa de todos los elementos de la dinámica simbólica que es necesario estudiar para comprender por qué el capítulo 4 es una generalización. La presentación fue hecha en el universo de los $\mathbb{Z}$ shifts porque los árboles del capítulo 4 son muy parecidos a los $\mathbb{N}$ shifts. Pusimos mucho énfasis en incluir gráficas y diagramas para que la idea general de cada una de las pruebas se pueda comprender por inspección. Recomendamos al nuevo lector que realice una primera leída a las definiciones y ejemplos para primero generar un panorama y después adentrarse en la lectura detallada. Cualquier tema que sea necesario profundizar puede acudir a [1] o en un nivel más avanzado a [3] ambos en la biblioteca de la Facultad de Ciencias.

Capítulo 1

Espacios shift

En este capítulo presentamos los espacios fundamentales de la teoría de la dinámica simbólica siguiendo como guía el libro [1]. Damos los primeros ejemplos de estos espacios y presentamos algunas características que pueden ayudarnos a formar muchos otros. También incluimos la definición de código que resultará fundamental durante todo el trabajo.

1.1. Shifts completos

Hoy en día es muy común representar información como una sucesión de símbolos. Los códigos binarios, presentes en todas las computadoras y teléfonos celulares, son el ejemplo más cercano que tenemos de un código inambiguo. El lenguaje escrito es otro código importante, aunque cuenta muchas veces con cierta dosis de ambigüedad. Cada vez que nos imaginemos un alfabeto podemos pensar en cualquier conjunto de símbolos escritos que nos ayudarán a transmitir información de manera eficaz. Los códigos son estructuras que permiten interpretar la información de manera única, lo que los hace muy importantes para la programación (tanto de computadoras como de humanos). Los diferentes tipos de espacios shift tendrán ciertas propiedades útiles. Los códigos usados para programar los discos compactos, por ejemplo, teóricamente no son un espacio shift de tipo finito, pero si son un shift sófico.

Definición 1.1. Un *alfabeto* $\mathcal{A}$ es un conjunto finito de símbolos (también llamados *letras*)

Los símbolos de un espacio shift son todos aquellos que participan en la transmisión de un mensaje. En la mayor parte de este trabajo nos reduciremos por practicidad al alfabeto {0,1} pero podríamos usar cualquier otro que nos resultara práctico. Por ejemplo, los símbolos de un texto en español no solo deben incluir a las letras sino a todos los signos de puntuación que son utilizados, pues juegan un papel dentro del texto, lo que hace tediosa la enumeración pero no complicada.

Definición 1.2. El shift completo de cierto alfabeto $\mathcal{A}$, el *shift* $\mathcal{A}$ *completo*, es el conjunto de todas las sucesiones sobre $\mathbb{Z}$ de símbolos de nuestro alfabeto $\mathcal{A}$.

Denotamos al shift completo como $\mathcal{A}^{\mathbb{Z}} = \{(x_i)_{i\in Z} | x_i \in \mathcal{A}\ \forall i \in \mathbb{Z}\}$ y escribiremos a estas sucesiones como $x = ...x_{-3}x_{-2}x_{-1}\ .x_0\ x_1x_2x_3....$ Colocamos un punto a la izquierda del primer término no negativo para ubicar el inicio de la numeración.

Ejemplo 1.1. El número 29 en su representación binaria en una sucesión sobre $\mathbb{Z}$ se escribiría como sigue:

$$29 = ...000.10111000...$$

También podemos tener sucesiones sobre los números naturales que crecen sólo hacia la derecha.

Definición 1.3. El shift completo del alfabeto $\mathcal{A}$ hacia un solo lado *shift derecho* $\mathcal{A}$ *completo* es el conjunto de todas las sucesiones sobre $\mathbb{N}$ de símbolos de nuestro alfabeto $\mathcal{A}$.

En este caso, nuestro número se vería así: $x = x_0\ x_1x_2x_3...$ por ejemplo $29 = 10111000...$ Durante este primer capítulo trabajaremos sobre shifts completos a menos que se indique otra cosa.

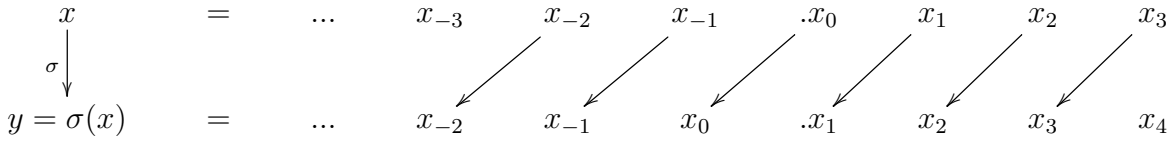
Los shifts completos son todas las posibles sucesiones de símbolos de nuestro alfabeto. La definición es muy ambiciosa. Si nuestro alfabeto fuera el del español, por ejemplo, podríamos tener un punto del shift completo que tuviera a todos los libros que hay en la biblioteca Porrúa y todavía tendríamos que completarla con una infinidad de espacios. Así, los puntos o elementos del espacio shift completo son sucesiones. Estas sucesiones pueden tener subsucesiones finitas. A las sucesiones finitas de símbolos consecutivos de un punto les llamaremos *bloques*. Un bloque de la sucesión que representa el número 29 en nuestro Ejemplo 1.1 es, digamos, 101 porque son los símbolos en la posición 1,2 y 3. Un bloque puede pensarse también como la imagen de un segmento de números consecutivos de $\mathbb{Z}$ en

la sucesión. Así, el bloque 1111 no es bloque de la sucesión binaria del número 29 porque aunque ésta sí contiene cuatro veces el símbolo "1"no se puede encontrar una subsucesion de símbolos *consecutivos* en el punto. Un *n-bloque* es un bloque que tiene n símbolos de nuestro alfabeto. 1111 es un 4-bloque y 101 es un 3-bloque. Sólo hay un bloque de tamaño 0, que es el bloque vacío que escribiremos como ϵ y denotaremos la longitud de un bloque b como $|b|$. El conjunto de todos los $k-$ *bloques* sobre $\mathcal{A}$ será denotado como $\mathcal{A}^k$

Dos bloques a y b pueden ser *concatenados* para formar un nuevo bloque de longitud $|ab| = |a| + |b|$ donde $ab = a_0a_1...a_{|a|-1}b_0...b_{|b|-1}$ y así a^n es la concatenación de a n veces.

Una forma abreviada de escribir a los bloques es: $x_{[i,j]} = x_ix_{i+1}...x_j$ con $i \leq j$.

La función más importante dentro de los espacios shift es justamente el shift. Esta consiste en desplazar la sucesión un lugar hacia la izquierda de tal manera que, podemos conocer el símbolo que se encuentra en el lugar n solamente viendo el lugar cero y aplicando n veces la función $y = \sigma(x)$.
Definición 1.4. El *shift* en un espacio shift es la función $\sigma : \mathcal{A}^{\mathbb{Z}} \longrightarrow \mathcal{A}^{\mathbb{Z}}$ tal que $y_i = \sigma(x)_i = x_{i+1}$ $\forall i \in \mathbb{Z}$.



Como esta función es un automorfismo (función biyectiva de un espacio en él mismo) de $\mathcal{A}^{\mathbb{Z}}$ tiene una inversa σ^{-1} que recorre una sucesión dada un lugar hacia la derecha.

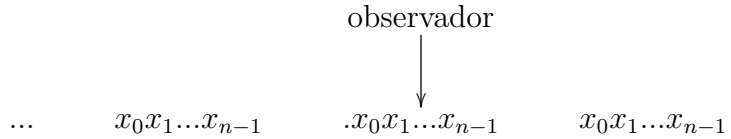
Dentro de los espacios shift completos, nuestro shift está bien definido, no así para los shift derechos completos en donde la función sigma no tiene una inversa única cuando nuestro alfabeto es no trivial, pues podríamos colocar cualquier elemento a del alfabeto como primer símbolo de $x' = ax_0x_1x_2x_3...$ y tras aplicar σ obtener el mismo elemento x en cualquier caso. Es decir, $\sigma^{-1}(x)$ es una función 1 a r donde $r = |\mathcal{A}|$. Así pues, para espacios shift derechos nos quedamos con σ recorriendo toda la sucesión un lugar a la izquierda pero no muy seguido hablaremos de su inversa . Para la composición de la función σ usaremos la notación convencional $\underbrace{\sigma \circ \sigma \circ ... \circ \sigma(x)}_{\text{k veces}} = \sigma^k$

Definición 1.5. Un *punto periódico* x es aquel para el cual existe un $n \in \mathbb{N}$ no cero tal que $\sigma^n(x) = x$.

Puede haber varios números $n \in \mathbb{N} - \{0\}$ para los cuales x sea periódico. Al más pequeño de estos se le llama el periodo de x. Si este número es no existe decimos que x no es periódico o es aperiódico. Sea x de periodo n y tomemos k de tal manera que $\sigma^k(x) = x$ entonces podemos efectuar la división k/n de donde $k = an + r$ con $0 \leq r \leq |n|$ y así $x = \sigma^k(x) = \sigma^{an+r}(x) = \sigma^{na} \circ \sigma^r(x) = \sigma^r(\sigma^{na}(x)) = \sigma^r(x)$ de donde concluímos que r debe ser nulo pues en otro caso sería el periodo pues es menor que n y sin embargo supusimos que n es el periodo.

En la prueba anterior se dice que $\sigma^{an}(x) = x$ y esto ocurre porque podemos descomponer $\sigma^{an}(x) = \underbrace{\sigma^n \circ \sigma^n \circ ... \circ \sigma^n}_{\text{a veces}}(x)$ y luego si x tiene periodo n entonces tiene periodo an para todo $a \in \mathbb{N}$

Podemos contar la cantidad de puntos periódicos en un espacio shift. Por ejemplo, para el shift derecho completo $\{0, 1, 2, ..., 9\}^{\mathbb{N}}$ tenemos una correspondencia inyectiva natural con los números racionales $\mathbb{Q}$ siendo todos los puntos periódicos un racional del intervalo $[0, 1]$ de manera que $f : \{0, 1, 2, ..., 9\}^{\mathbb{N}} \to \mathbb{Q}$ tal que $a \mapsto .a$. Podemos hacer lo mismo con cualquier alfabeto de cualquier tamaño. Si quisiéramos contar la cantidad de puntos de periodo n en un shift completo de alfabeto de cardinalidad $|\mathcal{A}| = a$ podemos hacer lo siguiente: si nos fijamos únicamente en la posición inicialmente ocupada por x_0 sabemos que tras aplicar n veces la función shift debe empezar de nuevo a ver los símbolos del bloque con el que comenzamos (ver la figura) y sólo de esta forma podemos formar un punto periódico por lo que la cantidad de puntos periódicos de periodo n debe ser $p = |\{x|\sigma(x)^n = x\}| = a^n$.



1.2. Subshifts y ejemplos

Desde el punto de vista de la información los espacios shift completos son un universo demasiado vasto pues tienen todas las posibles secuencias de símbolos. Muchas veces necesitamos restringir un poco nuestras sucesiones para conseguir un espacio manejable por diferentes razones. Un buen ejemplo tomado del español es la prohibición de bloques que no pueden ser pronunciados oralmente, como los de cuatro letras consonantes consecutivas, tres veces la misma letra, y hasta la prohibición de la concatenación de dos palabras que terminen y empiecen con la misma vocal: $< agua >$ es femenino pero su artículo es $< el >$

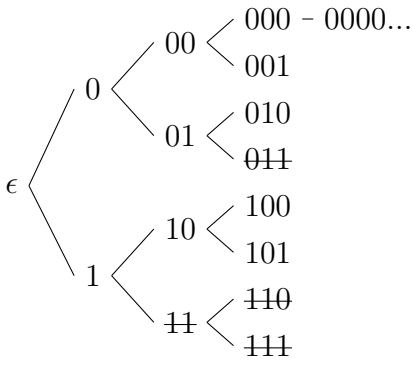
porque no es fácil pronunciar "la agua". En el caso de los discos compactos la restricción es de vida o muerte. Si tuviéramos dos magnetos cargados eléctricamente en la banda legible y estuvieran demasiado cerca generarían interferencia y no podríamos escuchar nuestra música nítidamente. En lenguajes de computación, por ejemplo en el del compilador Latex, no es posible que un comando contenga sólo un símbolo $ o un único paréntesis, pues su función es justamente actuar como agrupadores de objetos enmedio de ellos. Siguiendo este principio, a continuación prohibiremos ciertas sucesiones a partir de sus bloques. Decimos que b es un *bloque* de x si $x_{[i,j]} = b$ para algunas $i \leq j$ de $\mathbb{Z}$. Nombraremos $\mathcal{F}$ al conjunto de bloques prohibidos, es decir, que no pueden aparecer en ningún tiempo (o posición) de la sucesión y $\mathsf{X}_{\mathcal{F}} \subseteq \mathcal{A}^{\mathbb{Z}}$ el conjunto de todos aquellos puntos del shift completo que no tienen como bloque a alguno de los elementos de $\mathcal{F}$, es decir,

$$\mathsf{X}_{\mathcal{F}} = \{x \in \mathcal{A}^{\mathbb{Z}} |\ \forall k \in \mathbb{Z},\ \forall\, n \geq 0, x_{[k,k+n]} \notin \mathcal{F}\}.$$

Definición 1.6. Un *espacio shift* (también *subshift* o *shift*) es un subconjunto X del shift completo tal que $X = \mathsf{X}_{\mathcal{F}}$ para algún $\mathcal{F}$ conjunto de bloques prohibidos de $\mathcal{A}$.

Si un espacio shift X está contenido dentro de otro Y decimos que X es un subshift de Y. Cuando $\mathcal{F} = \emptyset$ tenemos $\mathsf{X}_{\mathcal{F}} = \mathcal{A}^{\mathbb{Z}}$ y opuestamente, $\mathcal{F} = \mathcal{A}$ nos da $\mathsf{X}_{\mathcal{F}} = \emptyset$. Resaltamos que $\mathsf{X}_{\mathcal{F}}$ es la operación de formar el shift juntando en un conjunto a todas las sucesiones sobre un alfabeto que no tienen bloques de $\mathcal{F}$ mientras que X es simplemente un conjunto.

Ejemplo 1.2. Sea el alfabeto $\mathcal{A} = \{0, 1\}$. Tomaremos un conjunto de palabras prohibidas $\mathcal{F}$ subconjunto de $\mathcal{A}^*$ que es el conjunto de posibles bloques sobre $\mathcal{A}$. Podemos tomar a $\mathcal{F} = \{11\}$ de tal manera que al generar $X = \mathsf{X}_{\mathcal{F}}$, éste contiene todas las sucesiones que no tienen dos números uno consecutivos. X es un subshift del shift completo $Y = \{0, 1\}^{\mathbb{Z}}$. Este subshift es muy importante en todo lo que resta del trabajo así que hagamos algunas observaciones: El conjunto de palabras prohibidas$\mathcal{F}' = \{11, 111\}$ genera exactamente el mismo shift de donde aprendemos que $\mathcal{F}$ puede no ser único. También X puede pensarse también como sucesiones de bloques de n ceros separados por un número uno. Podríamos escribir cualquer número de forma biunívoca en base decimal como una secuencia en la que el número de ceros anteriores al primer uno a partir del lugar x_0 representa la cantidad de unidades, los ceros que están entre el primer uno y el segundo son las decenas y así sucesivamente. De esta forma estamos "codificando" la información de una notación a otra. El número 134 se representaría como 134= ... 000.00001000101000.... . Le llamaremos el shift áureo o de la razón dorada porque la cantidad de n-bloques permitidos en este shift crece exponencialmente como la sucesión de fibonacci. Para ello observamos el diagrama:



Nos fijamos en la columna de bloques de tamaño 3. Los bloques de longitud 3 que serán aceptados serán todos los que fueron aceptados de tamaño 2 con un cero a su derecha mas todos los que fueron aceptados de tamaño 1 concatenados con el bloque 01 a su derecha. Así, inductivamente podemos deducir que si l_n es el conjunto de los bloques aceptados de longitud n por el shift áureo, éstos se pueden calcular recursivamente con la ecuación $|l_n| = |l_{n-1}| + |l_{n-2}|$. Sabemos que se relaciona la sucesión de Fibonacci con la razón áurea por el $\lim_{n\to\infty} \frac{|l_{n+1}|}{|l_n|} = \varphi$ la razón áurea. De aquí podemos deducir que la cantidad de n-bloques aceptados por nuestro shift son exactamente los mismos que el n-ésimo término de la sucesión de Fibonacci.

Ejemplo 1.3. Podemos también requerir a las sucesiones sobre el alfabeto binario $\mathcal{A} = \{0, 1\}$ tener sólo una cantidad par de ceros consecutivos. Es decir, podemos tener cualquier cantidad de unos juntos, pero siempre que veamos un cero, este pertenecerá a un bloque con una cantidad par de ceros entre dos símbolos 1. Para esto podemos declarar el conjunto $\mathcal{F}$ como $\{10^{2n+1}1 | n \in \mathbb{N}\}$ y generar $X = \mathsf{X}_{\mathcal{F}}$. Este conjunto se llama el *shift par*. Una de las propiedades especiales de este shift es que su conjunto $\mathcal{F}$ debe ser infinito y esta es la prueba: Supongamos que tenemos $\mathcal{J}$ finito tal que $\mathsf{X}_{\mathcal{J}}$ es un shift sobre $\mathcal{A}$. Podemos considerar $b_0 = \text{máx}\{b \mid \exists t \in \mathcal{J} \text{ con } b = 0^l \text{ subbloque de } t \text{ para alguna } l \in \mathbb{N}\}$ por lo que b_0 es el bloque de ceros de mayor longitud en $\mathcal{J}$. Supongamos $|b_0| = k$. Si formamos el bloque $z = 10^{2k+3}1$ el punto z^∞ no está permitido en el shift par y sin embargo pretenece a $\mathsf{X}_{\mathcal{J}}$ que por lo tanto no puede ser el shift par.

Ejemplo 1.4. Sea X el shift en el alfabeto binario {0,1} tal que todas sus secuencias tienen una infinidad de 1s y entre cada par de ellos hay uno, dos o tres ceros. Podemos definir en forma más general para cualquier par de números naturales (d, k) , $d \leq k$ el mismo shift donde el número de unos es por lo menos d y a lo más k. Se llama el shift de *carrera limitada* y es el que se usa para programar los discos y las memorias USB. En el primer caso nosotros estamos hablando del shift $\mathsf{X}(1, 3)$

Ejemplo 1.5. En una generalización del ejemplo pasado, podemos dar una lista $S = \{n_1, n_2, ...\}$ de números naturales y teniendo la fila infinita de símbolos 1 intercalar entre ellos una can-

tidad $n_i \in S$ de ceros. Así obtendríamos sucesiones de la forma $x = \ldots 10^{n_i}10^{n_{(i+1)}}10^{n_{(i+2)}}1...$ puntos de $\mathsf{X}(S)$. $\mathsf{X}(1,3)$ resulta ser generado por $S = \{1,2,3\}$. Cuando S es infinito es necesario que las sucesiones puedan comenzar o terminar con una secuencia infinita de ceros: Supongamos lo contrario, entonces hay un máximo k en la lista de los tamaños de las sucesiones de ceros y de allí que la lista S es, a lo más, de tamaño k. Cuando nos encontramos con un S infinito, preferimos llamar a X el *shift de salto en S*. De esa forma podemos ver que el shift áureo es el shift de salto de la lista $S = \{1,2,...\}$ y el shift par es el de la lista $S = \{0,2,4,...\}$. Otro bonito ejemplo seria el *shift de salto primo* para el cual tendríamos la lista $S = \{2,3,5,7,11...\}$ y un bloque permitido en él sería 1000100000001000001. Vale la pena observar que aunque $|S| = \infty$ esto no implica que todos los conjuntos conjuntos de palabras prohibidas que pueden generar el shift $\mathsf{X}(S)$ tienen que ser infinitos como en el caso del shift áureo.

Ejemplo 1.6. Dado $c \in \mathbb{N}$ el *shift de carga restringida a c* sobre $\mathcal{A} = \{+1,-1\}$ es aquel para el cual todos los bloques que aparecen de cualquier tamaño tienen una suma aritmética s de sus símbolos restringida por la constante c tal que $-c \leq s \leq c$. Por ejemplo cualquier bloque del punto $x = \cdots -11-11-11...$ tiene carga 0, 1 o -1 por lo que pertenece al shift de carga restringida 1. Este shift tiene aplicaciones en la ingeniería de software.

Ejemplo 1.7. Tomando $\mathcal{A} = \{a,b,c\}$ podemos definir el shift en el que a solo puede ser seguido por c y c solo puede ser seguido por ba de tal manera que tenemos un conjunto de bloques prohibidos $\mathcal{F} = \{cbb, cbc, ab\}$. Este es un *shift de tipo finito* o *STF* porque $\mathcal{F}$ es finito.

Ejemplo 1.8. Sea $\mathcal{A} = \{e,f,g\}$. El *shift de contexto libre* es aquel que acepta un bloque de la forma ef^ag^be sólo si $a = b$. Este shift es muy importante en la teoría de los autómatas porque no puede ser reconocido por un autómata determinista pero sí por uno más poderoso llamado pushdown autómata. Pertenece a una familia de lenguajes generada por las gramáticas de contexto libre y de allí el nombre.

Podemos generar muchos a partir de conjuntos $\mathcal{F}$. Simplemente en el alfabeto binario, la cantidad de conjuntos de palabras prohibidas es no numerable, pues cada elemento de la potencia del conjunto de bloques $\mathcal{B}(\{0,1\}^{\mathbb{Z}})$ es un posible $\mathcal{F}$ aunque hay que acordarnos que un solo shift puede ser generado por muchos conjuntos de palabras prohibidas. Todo shift definido a partir de un conjunto de palabras prohibidas tiene la propiedad de ser *shift invariante*, es decir, que su construcción no depende de la posición de sus elementos sino de una propiedad de cerradura heredada de forma automática. En versión corta, $\sigma(\mathsf{X}_\mathcal{F}) = \mathsf{X}_\mathcal{F}$. De allí que podemos identificar conjuntos de sucesiones que no son espacios shift, por ejemplo el conjunto que consta de una única sucesión binaria $X = \{x = ...0101.0101...\}$ pues de ser espacio shift sería shift invariante pero $\sigma(x) = (10)^\infty = ..,10,101...$ que no está en el conjunto. Sin embargo, que un conjunto sea shift invariante no es suficiente para que sea un espacio shift. Hace falta también una "cerradura" ilustrada en el siguiente ejemplo:

Ejemplo 1.9. Consideremos $\mathcal{A} = \{0, 1\}$ y X el conjunto de todas las sucesiones bilaterales que tienen únicamente un número uno seguido y precedido de puros ceros. Es shift invariante porque el número uno sigue siendo único al aplicar el shift. Si fuera un espacio shift, en $\mathcal{F}$ no podríamos prohibir ninguna cadena de ceros lo que permitiría a la sucesión z= ...000... formar parte del conjunto, lo cual no era parte de nuestro plan, pues esta última no tiene números uno. Este es un ejemplo de un conjunto sigma invariante que no es espacio shift.

Sin embargo, los espacios shift sí se pueden caracterizar por ser sigma invariantes y cerrados como veremos más adelante.

1.3. Lenguajes

Podemos describir un espacio shift a partir de los bloques que están permitidos. Un bloque $b = b_0 b_1 ... b_n$ está *permitido* en un shift $\mathsf{X}_{\mathcal{F}}$ si se cumple que $\forall i, j\ 0 \leq i \leq j \leq n\ b_{[i,j]} \notin \mathcal{F}$.

Definición 1.7. Sea X un espacio shift. El *lenguaje* de X es el conjunto de todos los bloques permitidos dentro de nuestro espacio. Sea $\mathcal{B}_n(X)$ el conjunto de todos los bloques de tamaño n que aparecen en X. El *lenguaje* del shift es el conjunto $\mathcal{L} = \mathcal{B}(X) = \bigcup_{n=1}^{\infty} \mathcal{B}_n(X)$

Ejemplo 1.10. El lenguaje del shift $Y = \{0, 1, 2\}^{\mathbb{Z}}$ es

$$\mathcal{B}(Y) = \{\epsilon, 0, 1, 2, 00, 01, 02, 10, 11, 12, 20, 21, 22, 000, ...\}$$

y el lenguaje del shift áureo es

$$\mathcal{L} = \{\epsilon, 0, 1, 00, 01, 10, 000, 100, 010, 001, 101, ...\}.$$

El término "lenguaje" vinene de la teoría de los lenguajes regulares. Una palabra pertenece a un lenguaje si puede cerrar un ciclo dentro de una máquina teórica llamada autómata que presentaremos más adelante. $\mathcal{B}(X)$ son todos los bloques que aparecen en algún punto de nuestro espacio. En lo que sigue, denotaremos como $\mathcal{C}^c$ al complemento de un conjunto de bloques $\mathcal{C}$ relativo a la colección de bloques sobre $\mathcal{A}$

Proposición 1.1. *(1) Sea X un espacio shift y $\mathcal{L} = \mathcal{B}(X)$ su lenguaje, si $w \in \mathcal{L}$, entonces:*
(a)Propiedad de subbloques. Todo subbloque de w pertenece a $\mathcal{L}$
(b)Propiedad de extensión. Existen u, v bloques no vacíos de tal manera que $uwv \in \mathcal{L}$
(2) Los lenguajes de los espacios shift están caracterizados por (1), esto es: si $\mathcal{L}$ es una colección de bloques sobre $\mathcal{A}$ entonces $\mathcal{L} = \mathcal{B}(X)$ para algún espacio shift X si y sólo si $\mathcal{L}$

cumple la condición (1)
(3) El lenguaje de un espacio shift determina al espacio shift, aún mejor, para cualquier espacio shift X se tiene que $X = \mathsf{X}_{\mathcal{B}(X)^c}$, luego, dos shifts son iguales si y sólo si tienen el mismo lenguaje

Demostración. (1) Sea $w \in \mathcal{L}$ entonces $w \in \mathcal{B}_n(X)$ para alguna $n \in \mathbb{N}$ y particularmente pertenece a $\mathcal{B}_n(x)$ para algún punto x de X. Pero entonces todo subbloque v de w pertenece a $\mathcal{B}_k(x)$ para alguna $k \leq n$ y por ende $v \in \mathcal{B}^n(X)$ de donde se sigue (a). Para obtener (b) observamos que como $v \in \mathcal{B}^n(x)$ entonces $w = x_{[i,j]}$ para alguna $i \leq j \in \mathbb{Z}$. Tomando $u = x_{i-1}, v = x_{j+1}$ ambos no vacíos obtenemos un nuevo bloque $uwv = x_{[i-1,j+1]}$ que pertenece a $\mathcal{L}$ por ser bloque de x . (2) La prueba de (1) es justamente el sólo si de esta parte. Así pues lo que queda por probar es que si $\mathcal{L}$ cumple ambas propiedades (a) y (b) entonces existe un espacio shift del cual es lenguaje. Sea $\mathcal{L}$ que cumple (a) y (b) y tomemos $X = \mathsf{X}_{\mathcal{L}^c}$. Probaremos que $\mathcal{L} = \mathcal{B}(X)$. Sea $w \in \mathcal{B}(X)$ entonces $w \notin \mathcal{L}^c$ de donde $w \in \mathcal{L}$ y así $\mathcal{B}(X) \subseteq \mathcal{L}$. Ahora supongamos que $w = x_0x_1...x_n \in \mathcal{L}$ podemos extender a w por la propiedad (1b) a un punto x de tal manera que $x_{[0,n]} = w$ y cualquier $x_jj \notin [0, n]$ sea el obtenido por la extensión. Este punto pertenece a X pues todos sus bloques pertenecen a $\mathcal{L}$ y por ésto $x \in \mathcal{B}(\mathsf{X}_{\mathcal{L}^c}) = \mathcal{B}(X)$, luego $\mathcal{B}(X) \supseteq \mathcal{L}$. (3) Dado $x \in X$ todos sus bloques pertenecen a $\mathcal{B}(X)$ y no hay ninguno en $\mathcal{B}(X)^c$ por lo que $x \in \mathsf{X}_{\mathcal{B}(X)^c}$ que es precisamente el shift que tiene a todos los puntos que cumplen que ninguno de sus bloques está en $\mathcal{B}(X)^c$. Por una razón similar, como X es un shift, existe un $\mathcal{F}$ de tal manera que $X = \mathsf{X}_{\mathcal{F}}$. Dado $x \in \mathsf{X}_{\mathcal{B}}(X)^c$ ninguno de sus bloques está en $\mathcal{B}(X)^c$ por lo que todos son bloques de X y de allí que no pueden estar en $\mathcal{F}$. Luego $w \in \mathcal{B}(X) = \mathsf{X}_{\mathcal{F}}$ de donde $\mathsf{X}_{\mathcal{F}} \subseteq \mathsf{X}_{\mathcal{B}(X)^c}$. □

Este resultado se basa en el uso de $\mathcal{B}(X)^c$ que es el conjunto maximal de bloques prohibidos para cualquier lenguaje, pues justamente prohibe todos aquellos bloques de X que no son sus bloques. Estas equivalencias se pueden sintetizar en las siguientes dos ecuaciones:

$$\mathcal{L}(X) = \mathcal{B}(\mathsf{X}_{\mathcal{B}(X)^c})$$

$$X = \mathsf{X}_{\mathcal{B}(X)^c}.$$

De la parte (3) de la proposición también obtenemos el siguiente corolario.

Corolario 1.1. *Sea $X \subseteq \mathcal{A}^{\mathbb{Z}}$ entonces X es un subshift si y sólo si dado $x \in \mathcal{A}^{\mathbb{Z}}$ siempre que $\forall (i, j) \in \mathbb{Z}^2$, $i \leq j$ $x_{[i,j]} \in \mathcal{B}(X)$ ocurre que $x \in X$*

Demostración. La condición de que todo bloque de x sea bloque de X nos asegura que $x \in \mathsf{X}_{\mathcal{B}(X)^c} = X$ □

Un punto importante en el estudio de los espacios shift es que se puede generar un nuevo espacio shift a partir de dos conocidos uniendo sus conjuntos de palabras prohibidas

Lema 1.1. *Sea $\mathcal{A}$ un alfabeto. Dados X_1 y X_2 shifts sobre $\mathcal{A}$, $X_1 \cap X_2$ es un shift y su conjunto de palabras prohibidas es $\mathcal{F}_1 \cup \mathcal{F}_2$*

Demostración. Sean $X_1 = \mathsf{X}_{\mathcal{F}_1}, X_2 = \mathsf{X}_{\mathcal{F}_2}$ entonces $X_1 \cap X_2 = \mathsf{X}_{\mathcal{F}_1} \cap \mathsf{X}_{\mathcal{F}_2} = \{x \in \mathcal{A}^{\mathbb{Z}} | \forall b$ bloque de $x, b \notin \mathcal{F}_1\} \cap \{x \in \mathcal{A}^{\mathbb{Z}} | \forall b$ bloque de $x, b \notin \mathcal{F}_2\} = \{x \in \mathcal{A}^{\mathbb{Z}} | \forall b$ bloque de $x, b \notin \mathcal{F}_1 \cap \mathcal{F}_2\} = \mathsf{X}_{\mathcal{F}_1 \cap \mathcal{F}_2}$ □

Podemos construir un conjunto mínimo de palabras prohibidas en base a los "primeros ofensores" de X, donde $w \notin \mathcal{B}(X)$ es un primer ofensor cuando para todo v subbloque propio de w ocurre que $v \in \mathcal{B}(X)$. Llamaremos $\mathcal{O}$ al conjunto de todos los primeros ofensores de X

Proposición 1.2. *Sea X un espacio shift. Entonces*
(a) $X = \mathsf{X}_{\mathcal{O}}$.
(b) Si $X = \mathsf{X}_{\mathcal{F}}$ entonces para cada $w \in \mathcal{O}$ existe una $v \in \mathcal{F}$ de tal manera que w es subbloque de v pero v no contiene ningún otro primer ofensor.
(c) Por (b) $\mathcal{O}$ es un conjunto mínimo de palabras prohibidas, es decir, si $\mathcal{F} \subseteq \mathcal{O}$ y $X = \mathsf{X}_{\mathcal{F}}$ entonces $\mathcal{F} = \mathcal{O}$.

Demostración. (a) Sea $\mathcal{L}$ el lenguaje de X y sea $\mathcal{L}'$ el de $\mathsf{X}_{\mathcal{O}}$. Vamos a ver que ambos son el mismo lenguaje. Sea $b \in \mathcal{L} = \mathcal{B}(\mathsf{X}_{\mathcal{O}})$ entonces $b \notin \mathcal{O}$ por lo que $b \in \mathcal{B}(X) = \mathcal{L}$. Además sea $b \in \mathcal{B}(X) = \mathcal{L}$ entonces $b \notin \mathcal{O}$ de donde $b \in \mathcal{B}(X_{\mathcal{O}}) = \mathcal{L}'$ de donde $\mathcal{L} = \mathcal{L}'$ (b) Naturalmente existe $v \in \mathcal{F}$ tal que $w \in \mathcal{O}$ es subbloque de ella pues de lo contrario el punto $x = w^{\infty}$ pertenecería a X y por lo tanto w no sería ofensor. El bloque v no puede tener otro primer ofensor z porque la construcción de w es única quitando símbolos de los extremos derecho e izquierdo hasta que conseguimos un bloque que sí pertenece a X. (c) Para cada bloque de $\mathcal{O}$ debe haber uno en $\mathcal{F}$ lo que preserva la desigualdad $|\mathcal{O}| \leq |\mathcal{F}|$. □

La Proposicion 1.1 nos asegura que dada una palabra w de un shift X es posible "extenderla" hacia ambos lados pero es importante saber si dadas dos palabras $v, u \in \mathcal{B}(X)$ es posible encontrar una w que las una. Esto no ocurre en todos los espacios, por ejemplo, en el shift que tiene tres puntos X= {...01.0101..., ...10.1010..., 0^{∞}} no es posible unir a 00 con 10, ambos bloques de X. Los shift que tienen esta propiedad son especiales.

Definición 1.8. Un espacio shift X es *irreducible* si para cualquier par ordenado de bloques $v, u \in \mathcal{B}(X)$ existe un $w \in \mathcal{B}(X)$ tal que $vwu \in \mathcal{B}(X)$.

Es importante estar al tanto de que como en la definición hay orden, dados u y v debemos encontrar w_1 y w_2 de tal manera que $uw_1v \in \mathcal{B}(X)$ y $vw_2u \in \mathcal{B}(X)$

Nuestros ejemplos 1.5 y 1.6 son ambos irreducibles. En 1.6 podemos unir cualesquiera dos bloques dados $u = u_0...u_k$ y $v = v_0...v_l$ primero evaluando sus cargas $c_1 = \sum_{0\leq j\leq k} u_j$ y $c_2 = \sum_{0\leq j\leq l} v_j$ y luego generando un bloque neutralizador formado de dos elementos n_1, n_2 definidos como sigue, recordando que tenemos una carga cota c para $i = 1, 2$,

$$n_i = \begin{cases} (-1)^{c_i} & \text{si } c_i \text{ es positivo} \\ (1)^{|c_i|} & \text{si } c_i \text{ es negativo} \end{cases}$$

$$n_c = \begin{cases} (-1)^{c} & \text{si } c_1 + c_2 \text{ es positivo} \\ (1)^{c} & \text{si } c_1 + c_2 \text{ es negativo} \end{cases}$$

Y concatenando luego $un_1n_cn_2v$ el cual tiene carga $signo(c_1 + c_2)n_c$ obtenemos nuestro bloque buscado. Notemos que este termino sigue funcionando bajo la permutación de u y v

Para el ejemplo 1.5 una vez que tenemos nuestros bloques u y v diseñamos $w_1 = 0^k10^l$ de tal manera que k completa la paridad en la fila de ceros del extremo derecho del bloque u y lo mismo hace l con el extremo izquierdo de v.

Los shifts que son reducibles pueden ser partidos en piezas irreducibles y a cada una de las piezas podemos aplicarle la teoría. Bajo este supuesto el concepto de irreducible es general en los shifts y se convertirá en una propiedad medular para poder aplicar el álgebra lineal en la teoría.

1.4. Shifts en Presentaciones por Bloques

Una de las construcciones fundamentales de la dinámica simbólica son las presentaciones por bloques de los puntos de un shift. Podemos sustraer la atención puesta en un símbolo de nuestro alfabeto para fijarla ahora en bloques, considerándolos letras de un alfabeto nuevo. Este proceso conocido como "pasar a una presentación de bloques" es muy conveniente y será usado recurrentemente. Nos provee de una presentación alternativa del mismo espacio.

Este trabajo se enfoca en desarrollar la presentación de los shifts en su forma de árbol. Así como la presentación de bloques ha simplificado tremendamente el trabajo con los shifts,

tenemos la esperanza de que la presentación de árboles nos permita tener una perspectiva alternativa de los mismos espacios. Así pues, esta sección fundamenta el desarrollo de nuevas presentaciones de los espacios shift al haber resultado tan útil en la investigación a lo largo de estos últimos cincuenta años.

Sea X un espacio shift y $\mathcal{A}$ su alfabeto, sea $\mathcal{A}_X^{[N]} = \mathcal{B}_N(X)$ la colección de todos los bloques permitidos de tamaño N. Podemos considerar a $\mathcal{A}_X^{[N]}$ como un nuevo alfabeto y formar su shift $(\mathcal{A}_X^{[N]})^{\mathbb{Z}}$. Definimos el *código de la N-ésima potencia de bloques de X*, $\beta_N : X \to (\mathcal{A}_X^{[N]})^{\mathbb{Z}}$ como:

$$(\beta_N(x))_{[i]} = x_{[i,i+N-1]}$$

Entonces β_N reemplaza al símbolo x_i por el bloque de tamaño N de x que comienza en x_i. La imagen de x bajo β_N se representa como sigue:

$$\beta_N(X) = ... \begin{bmatrix} x_{N-2} \\ \vdots \\ x_0 \\ x_{-1} \end{bmatrix} . \begin{bmatrix} x_{N-1} \\ \vdots \\ x_1 \\ x_0 \end{bmatrix} \begin{bmatrix} x_N \\ \vdots \\ x_2 \\ x_1 \end{bmatrix} \begin{bmatrix} x_{N+1} \\ \vdots \\ x_3 \\ x_2 \end{bmatrix} ... \in (\mathcal{A}_X^{[N]})^{\mathbb{Z}}$$

Definición 1.9. Sea X subshift. El *shift de bloques de tamaño N* o *la presentación en N-bloques* de X es la imagen de X bajo $\beta_N(X) = X^{[N]}$ en el shift completo $\mathcal{A}_X^{[N]}$

Vemos que en la presentación hay un traslapamiento de bloques pues si u y v son N-bloques entonces puede darse el caso de que $u_i u_{i+1} ... u_N = v_1 v_2 ... v_{N-i}$. Diremos que dos bloques se *traslapan progresivamente* cuando $i = 2$. Además podemos ver que tomando el último símbolo de cada bloque podemos reconstruir fácilmente a x, por lo que decimos que simplemente es otra presentación de X, o formalmente, que son conjugados.

Ejemplo 1.11. Si recordamos el ejemplo 1.2 X el shift áureo estándar podemos tomar su alfabeto de bloques de tamaño 2 que es el siguiente

$$\mathcal{A}_X^{[2]} = \{a = 00, b = 01, c = 10\}$$

Y el shift áureo de bloques de orden 2 queda descrito como $X^{[2]} = \mathsf{X}_{\mathcal{F}}$ con el siguiente conjunto de restricciones $\mathcal{F} = \{ba, ca, cb, cc\}$ porque estos bloques no se traslapan progresivamente . El bloque 11 no aparece en el shift original X por lo que podemos estar seguros de que no aparecerá en el shift por bloques porque el contradominio del código es el conjunto de bloques permitidos en X.

Como vimos en el ejemplo anterior, para pasar a la presentación en bloques necesitamos prohibir dos cosas: (1) que no aparezcan palabras prohibidas en el shift orignial X dentro de los elementos del nuevo alfabeto $\mathcal{A}^N$ y también (2) que los bloques se traslapen progresivamente. Analizamos este proceso en el siguiente teorema

Proposición 1.3. *La presentación en bloques de un shift es también un espacio shift*

Demostración. Sea X un shift sobre $\mathcal{A}$ y $N \geq 1$. Para X existe una colección de bloques sobre $\mathcal{A}$ de tal manera que $X = \mathsf{X}_{\mathcal{F}}$. Generamos una nueva colección $\tilde{\mathcal{F}}$ donde cada bloque de tamaño $|u| \leq N$ es reemplazado por todos aquellos de tamaño N sobre $\mathcal{A}$ que lo tienen como subbloque. $X = \mathsf{X}_{\tilde{\mathcal{F}}}$ porque aumentar el tamaño de los bloques no tiene consecuencia pues de todas formas los bloques aumentados ya están prohibidos. Así cada bloque de $\tilde{\mathcal{F}}$ ya tiene un tamaño mayor a N. Ahora definimos nuestra lista de bloques prohibidos que se deriva de nuestra condición (1). Para cada $w = w_1w_2...w_m \in \tilde{\mathcal{F}}$ sea $w^{[N]} = (w_1w_2...w_N)(w_2w_3...w_{N+1})...(w_{m-N+1}w_{m-N+2}...w_m)$ el correspondiente $(m - N + 1)$ bloque sobre $\mathcal{A}^N$. Sea $\mathcal{F}_1$ el conjunto de todos los bloques sobre $\mathcal{A}^N$ de la forma $w^{[N]}$ para algún $w \in \tilde{\mathcal{F}}$. En este conjunto están pues todos los bloques en $\mathcal{A}^N$ que contienen alguna palabra prohibida dentro del shift X. Luego $X^{[N]} \subseteq \mathsf{X}_{\mathcal{F}_1}$. Para la condición (2) sea $\mathcal{F}_2 = \{uv | u \in \mathcal{A}^N, v \in \mathcal{A}^N, u \text{ y } v \text{ no se traslapan progresivamente}\}$. Luego $X^{[N]} \subseteq \mathsf{X}_{\mathcal{F}_2}$ y por el lema 1.1 de la unión de lenguajes $X^{[N]} \subseteq \mathsf{X}_{\mathcal{F}_1} \cap \mathsf{X}_{\mathcal{F}_2} = \mathsf{X}_{\mathcal{F}_1 \cup \mathcal{F}_2}$

Supongamos ahora que $y \in X = \mathsf{X}_{\mathcal{F}} = \mathsf{X}_{\mathcal{F}_1 \cup \mathcal{F}_2}$ y sea x el punto reconstruído usando las coordenadas de la base a partir de la definición 1.9. Entonces $x \in X = \mathsf{X}_{\mathcal{F}}$ porque y satisface las condiciones de $\mathcal{F}_1$ porque todo elemento prohibido en $\mathcal{F}$ está prohibido en $\mathcal{F}_1$ y su configuración por bloques es $y = \beta_N(x)$ así que se traslapa progresivamente por definición y cumple con lo requerido por $\mathcal{F}_2$. Luego $X^{[N]} \supseteq \mathsf{X}_{\mathcal{F}_1 \cup \mathcal{F}_2}$ y asi $X^{[N]} = \mathsf{X}_{\mathcal{F}_1 \cup \mathcal{F}_2}$ □

La configuración por bloques de X utiliza bloques que se traslapan pero podemos hacer otra construcción evitando este fenómeno. Así pues definimos el *N-ésimo código de potencia* $\gamma_N : X \to (\mathcal{A}_X^{[N]})^{\mathbb{Z}}$ como $(\gamma_N(x))_{[i]} = x_{[iN, iN+N-1]}$

En este caso γ_N corta los primeros N símbolos de $\mathcal{A}$ de la sucesión y los coloca en un vector de N entradas de $\mathcal{A}$ o un punto de $\mathcal{A}_X^{[N]}$ y lo mismo hace con los segundos N y así consecutivamente. Por ejemplo, la imagen de $x = (x_i)_{i \in \mathbb{Z}}$ bajo γ_4 es de la forma

$$\gamma_4(X) = ... \begin{bmatrix} x_{-1} \\ x_{-2} \\ x_{-3} \\ x_{-4} \end{bmatrix} \cdot \begin{bmatrix} x_3 \\ x_2 \\ x_1 \\ x_0 \end{bmatrix} \begin{bmatrix} x_7 \\ x_6 \\ x_5 \\ x_4 \end{bmatrix} \begin{bmatrix} x_{11} \\ x_{10} \\ x_9 \\ x_8 \end{bmatrix} ... \in (\mathcal{A}_X^{[4]})^{\mathbb{Z}}.$$

Si comparamos esta presentación con la anterior podemos darnos cuenta que en este caso los símbolos inferiores no son suficientes normalmente para reconstruir al punto x

Definición 1.10. Sea X shift. La *configuración de bloques sin traslape* X^N de X es la imagen $X^N = \gamma_N(X)$ sobre el shift completo $(\mathcal{A}_X^{[N]})^{\mathbb{Z}}$.

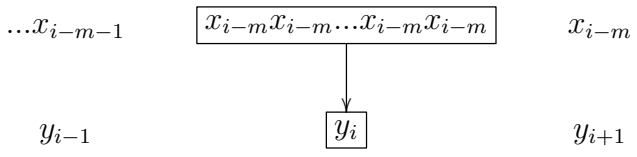
1.5. Códigos de bloques

En esta sección nos gustaría estudiar el siguiente problema: supongamos que queremos convertir a un punto $x = ...x_{-1}.x_0x_1...$ de un alfabeto $\mathcal{A}$ a una nueva sucesión de símbolos $y = ...y_{-1}.y_0y_1...$ de un nuevo alfabeto $\mathfrak{A}$. Podemos fijar $m, n \in \mathbb{Z}$ tales que $-m \leq n$. Para calcular la i-ésima coordenada y_i de la secuencia transformada usamos una función Φ que depende de la ventana de coordenadas de x desde $i-m$ hasta $i+n$. Luego $\Phi : \mathcal{B}_{m+n+1}(X) \to \mathfrak{A}$ es un *mapeo fijo de bloques* llamado $(m+n+1)$ *mapeo de bloques* o *regla local* de los $(m+n+1)$-bloques permitidos de X a símbolos de $\mathfrak{A}$, y así

$$y_i = \Phi(x_{i-m}x_{i-m+1}x_{i+n}) = \Phi(x_{[i-m,i+n]}).$$

Definición 1.11. Sea X un espacio shift sobre $\mathcal{A}$ y $\Phi : \mathcal{B}_{m+n+1}(X) \to \mathfrak{A}$ un mapeo de bloques. Entonces la función $\phi : X \to \mathfrak{A}^{\mathbb{Z}}$ definida por $y = \phi(x)$ con y_i dado por la ecuacion de arriba es llamado el *código de bloques de memoria m y anticipación n inducido por* Φ. Denotaremos esta relación como $\phi = \Phi_\infty^{[-m,n]} = \Phi_\infty$ usando el tercer término de la igualdad cuando la memoria y anticipaicón de ϕ estén sobreentendidas. Siempre que no se especifique, la memoria será considerada nula. Si Y es un espacio shift contenido en $\mathfrak{A}^{\mathbb{Z}}$ podemos escribir $\phi : X \to Y$.

La figura siguiente ilustra cómo actúa el código de bloques:



Los más simples de los códigos son aquellos que no tienen ni memoria ni anticipación, es decir, para los que $m = n = 0$. En estos la i-ésima coordenada de la imagen depende sólo de x_i. Éstos son llamados *1-códigos* o *códigos de 1-bloques*.

Ejemplo 1.12. Sea X un shift sobre $\mathcal{A} = \mathfrak{A}$, $m = 0$, $n = 1$ y $\Phi(a_0a_1) = a_1$. Entonces $\phi = \Phi_\infty^{[0,1]} = \Phi_\infty$ es el shift σ_X. ¿Qué pasa si tomamos ahora $\Phi(a_0a_1) = a_0$? Esta función es la identidad aunque uno tiende a pensar que es la inversa del shift. Pero si tomamos $m = 1, n = 0$ y $\Phi(a_{-1}a_0) = a_{-1}$ entonces sí $\psi = \Psi_\infty^{[-1,0]} = \sigma_X^{-1}$ es la función inversa del shift tal que $\psi(\phi(x)) = x = \phi(\psi(x))$ para toda $x \in X$. Resaltamos que si $\Theta(a) = a$ para toda $a \in \mathcal{A}$ entonces $\phi = \Theta_\infty^{[1,1]}$ y $\psi = \Theta_\infty^{[-1,-1]}$. Así pues hay muchas maneras de representar cada función como en este caso la inversa de la función shift.

Ejemplo 1.13. Sean $\mathcal{A} = \{0, 1\} = \mathfrak{A}$ el alfabeto binario, $X = \mathcal{A}^{\mathbb{Z}}, m = 0, n = 1$ y $\Psi(a_0a_1) = a_0 + a_1 \mod 2$. Cada uno de los símbolos se reemplaza por la suma de él con su vecino como se puede apreciar en la figura siguiente

$$\begin{array}{lccc} x = & \ldots 1100101 & \boxed{01} & 01011100111\ldots \\ \psi \downarrow & & \Psi \downarrow & \\ \psi(x) = & 0101111 & \boxed{1} & 1100101100 \end{array}$$

Hay dos tecnicismos que es bueno resaltar en la definicion de un código de bloques. El primero consiste en extender la "ventana" de Φ de tal manera que su memoria y anticipación sean mayores. Tomamos $M \geq m, N \geq n$ y definimos $\widehat{\Phi} : \mathcal{B}_{m+n+1}(X) \to \mathfrak{A}$ por $\widehat{\Phi}(x_{[M,N]}) = \Phi(x_{[m,n]})$ la cual tiene una ventana mayor aunque desprecia los símbolos extra que fueron incluidos en su ventana. La segunda es que podemos pasar de $\Phi : \mathcal{B}_{m+n+1}(X) \to \mathfrak{A}$ a una Φ que manda $(m + n + k)$-bloques de X a k-bloques de $\mathfrak{A}$ como sigue. Dado $x_{[-m,n+k-1]} \in \mathcal{B}_{m+n+k}(X)$ realizamos $\Phi(x_{[-m,n+k-1]}) = \Phi(x_{[-m,n]})\Phi(x_{[-m+1,n+1]})...\Phi(x_{[-m+k-1,n+k-1]})$. En el ejemplo de arriba $k = 18$.

Ejemplo 1.14. Sea $\mathcal{A} = \mathfrak{A} = \{0, 1\}, X$ el shift aureo del ejemplo 1.2 y Y el shift par del ejemplo 1.3 exceptuando los puntos que tienen colas de ceros. Sea Φ la función sobre 2-bloques definida como $\Phi(00) = 1, \Phi(01) = 0 = \Phi(10)$. No necesitamos definirla para el bloque 11 porque nunca aparece en nuestro dominio. Afirmamos que $\phi = \Phi_\infty : X \to Y$ es suprayectiva. Sea $l = 10^k1$, con k par, un bloque de Y. Para l construímos el bloque de X $b_l = 0(01)^r0$ con $2r = k$ lo cual podemos hacer porque k es par y b_l no tiene a 11. Como todo punto $y \in Y$ tiene una partición en bloques de la forma de l concatenamos todos los b_l para formar nuestro punto $x = ...b_lb_{l+1}b_{l+2}... \in X$ tal que $\phi(x) = y$ y así nuestra función es suprayectiva.

Una observación importante es que aquí el código de anticipación 1 barre bloques de cualquier tamaño pero no necesitamos modificar las ventanas porque simplemente obtendríamos la misma imagen. Es decir, para encontrar la imagen de bloques lo único que necesitamos es que cumplan con ser más grandes que el tamaño de la ventana.

Una propiedad fundamental de los códigos de bloques es que conmutan con el shift:

Proposición 1.4. *Sean X y Y dos espacios shift. Si $\phi : X \to Y$ es un código de bloques entonces $\phi \circ \sigma_X = \sigma_Y \circ \phi$ es decir, el siguiente diagrama conmuta:*

$$\begin{array}{ccc} X & \xrightarrow{\sigma_X} & X \\ {\scriptstyle\psi}\downarrow & & \downarrow{\scriptstyle\psi} \\ Y & \xrightarrow{\sigma_Y} & Y \end{array}$$

Demostración. Sea ϕ la función inducida por $\Phi : \mathcal{B}_{m+n+1}(X) \to \mathfrak{A}$ de memoria m y anticipación n. Para $x \in X$ tenemos $(\sigma_Y \circ \phi)(x)_{[i]} = \phi(x)_{[i+1]} = \Phi(x_{[i+1-m,i+1+n]})$, mientras que $(\sigma_X \circ \phi)(x)_{[i]} = \phi(\sigma(x))_{[i]} = \Phi(\sigma_X(x)_{[i-m,i+n]})\Phi((x)_{[i+1-m,i+1+n]})$ y al coincidir coordenada por coordenada, las sucesiones imagen en Y son iguales. □

No todas las funciones entre espacios shift que conmuten con el shift son códigos de bloques

Ejemplo 1.15. Sea X el espacio shift completo sobre $\mathcal{A} = \{0, 1\}$. Definimos para cada $x \in X$ la función $\zeta(x_i) =$máx $\{k | x_i x_{i+1} ... x_{i+k} = 0^k\}$ mód 2 que asigna a cada término la longitud del mayor bloque de ceros consecutivos que podemos encontrar inmediatamente después de él. Este no es un código de bloques porque no podemos definir su anticipación pues el bloque de ceros puede ser de tamaño arbitrario. Sin embargo conmuta con el shift pues sólo depende de los símbolos posteriores.

Es por esto necesario requerir a la función basarse en una ventana:

Proposición 1.5. *Sean X y Y espacios shift. Una función $\phi : X \to Y$ es un código de bloques si y sólo si $\phi \circ \sigma_X = \sigma_Y \circ \phi$ y existe una $N \in \mathbb{N}$ tal que $\phi(x)_0$ es una función de $x_{[-N,N]}$.*

Demostración. La suficiencia es automática pues basta extender la ventana del código de bloques a una que tenga la misma memoria y anticipación, lo cual ya fue comentado y de donde obtenemos para toda i el símbolo $\phi(x)_i$. Para la necesidad definimos una nueva función $\Phi(w) = \phi(x)_0$ donde $w = x_{[-N,N]}$ para algún punto x de X. Para conocer pues la imagen de un bloque tomamos un punto que lo tenga como N-bloque central y ϕ nos arroja su correspondiente bloque en Y. □

Cuando un código de bloques $\phi : X \to Y$ es *suprayectivo* es llamado un *código factor de X en Y* y decimos que Y es un *factor* de X cuando existe uno de dichos códigos. En cierta forma, X es más grande y Y es visto como una pieza de X. De igual forma cuando $\phi : X \to Y$ es *inyectivo* es llamado un *encaje* de X en Y. Los códigos de bloques $\beta_N : X \to (\mathcal{A}_X^{[N]})^{\mathbb{Z}}$ son encajes. Cuando un código $\phi : X \to Y$ tiene una inversa $\psi : Y \to X$ tal que $\phi(\psi(y)) = y \ \forall y \in Y,\ x = \psi(\phi(x)) \ \forall x \in X$, es decir, si ϕ y ψ son tanto encajes como factores, decimos que ϕ es *invertible* y escribimos $\phi = \psi^{-1}$.

La siguiente es una definición fundamental para nuestro trabajo pues el teorema de descomposición lo tiene como hipótesis fuerte y clasifica a los espacios con respecto una propiedad derivada de ella.

Definición 1.12. Un código de bloques $\phi : X \to Y$ es una *conjugación* entre dos subshifts X y Y si es invertible y su inversa es un código de bloques. Dos espacios shift X y Y son *conjugados* (escribimos $X \cong Y$) si hay una conjugación entre X y Y.

En la definición anterior también se pide la función inversa de ϕ sea invertible pero no lo hemos requerido porque en el teorema 1.5 probaremos que estas funciones inversas siempre son códigos de bloques.

Dos shifts que son conjugados pueden ser interpretados como el mismo objeto aunque "recodificados". En la literatura las conjugaciones son llamadas"conjugaciones topológicas"

Ejemplo 1.16. Sea X un subshift sobre $\mathcal{A}$ y $X^{[N]}$ la representación en N-bloques de X. Por la proposición 1.3 $\beta_N : X \to X^{[N]}$ es un código de bloques y tomando $\Psi : \mathcal{A}_X^{[N]} \to \mathcal{A}$ como $\Psi(a_0a_1...a_{N-1}) = a_0$ y con $\psi = \Psi_\infty : X^{[N]} \to X$ obtenemos una inversa para β_N de tal manera que es una conjugación y luego $X \cong X^{[N]}$. De igual manera podemos construir la inversa para γ_N con $\xi : \mathcal{A}_X^{[N]} \to \mathcal{A}^N$ donde $\Xi(a_0a_1...a_{N-1}) = a_0a_1...a_{N-1}$ y extendemos a $\xi = \Xi_\infty^{[0,N]} : X^N \to X$.

Ahora veamos como se comportan los puntos periódicos, que presentamos en la definición 1.5, bajo los códigos de bloques.

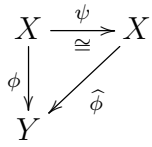
Proposición 1.6. *Sea $\phi : X \to Y$ un código de bloques. Si $x \in X$ es de periodo n bajo σ_X entonces $\phi(x)$ tiene periodo n bajo σ_Y y el periodo mínimo de $\phi(x)$ divide al periodo mínimo de x. Encajes y conjugaciones preservan el periodo mínimo de un punto.*

Demostración. Si x tiene periodo n entonces $\sigma_X^n(x) = x$ de donde $\sigma_Y^n(\phi(x)) = \phi(\sigma_X^n(x)) = \phi(x)$, así que $\phi(x)$ tiene periodo n y luego su periód mínimo divide a n como mencionamos en la página 6. Cuando ϕ es inyectiva podemos aplicar el razonamiento para $\phi(x)$ y para $\phi^{-1}(y)$ lo que arroja que ambos periodos se dividen y de allí que son iguales. □

La proposición nos asegura que dos shifts conjugados tienen la misma cantidad de puntos periódicos, es decir, que el número de puntos periódicos es un *invariante* bajo conjugación. Esta es una buena forma de probar que dos shifts no pueden ser conjugados, pues pueden no tener la misma cantidad de puntos de periodo n.

Podemos transformar a un código de m-bloques $\phi : X \to Y$ en un código de 1-bloques reconfigurando el alfabeto. Esto simplifica muchas demostraciónes pues es mucho más fácil pensar en códigos de 1-bloques que en aquellos en los que tenemos que navegar por los bloques.

Proposición 1.7. *Sea $\phi : X \to Y$ un código de bloques. Entonces estiste una presentación en bloques $\widehat{X}$ de X y una conjugación $\psi : X \to \widehat{X}$ y un 1-código $\widehat{\phi} : \widehat{X} \to Y$ de tal manera que $\widehat{\phi} \circ \psi = \phi$ es decir, el siguiente diagrama conmuta:*



Demostración. Supongamos que ϕ es inducido por la función de bloques Φ de memoria m y anticipación n. Sea $\mathfrak{A} = \mathcal{B}_{m+n+1}(X)$ y definimos $\psi : X \to \mathfrak{A}^{\mathbb{Z}}$ por $\psi(x)_{[i]} = x_{[i-m,i+n]}$. Luego $\psi = \sigma^{-m} \circ \beta^{m+n-1}$. Sabiendo que $\widehat{X} = \psi(X) = X^{[m+n+1]}$ es un espacio shift y σ y β son conjugaciones en consecuencia también lo es ψ. Sea $\widehat{\phi} = \phi \circ \psi^{-1}$. □

Notemos que $\widehat{\phi}$ es un código de 1-bloques pues su alfabeto son precisamente los bloques que procesa ϕ. Ahora probamos que la imagen de un espacio shift bajo un código de bloques es un espacio shift.

Teorema 1.1. *Sean X, Y shifts. Si $\phi : X \to Y$ es un código de bloques entonces $\phi(Y)$ es espacio shift.*

Demostración. Podemos asumir que $\phi : X \to Y$ es un código de 1-bloques por la proposición 1.7. Sea Φ el mapeo de bloques que induce ϕ. Sea $\mathcal{L} = \{\Phi(w) | w \in \mathcal{B}(X)\}$ veremos que $\phi(X) = \mathsf{X}_{\mathcal{L}^c}$ lo cual nos garantizaría que la imagen de X es un shift. Si $X \in x$ entonces todos sus bloques pertenecen a $\mathcal{L}$ de donde $\phi(X) \in \mathsf{X}_{\mathcal{L}^c}$. Esto prueba que $\phi(X) \subseteq \mathsf{X}_{\mathcal{L}^c}$ Supongamos ahora que $y \in \mathsf{X}_{\mathcal{L}^c}$. Para cada $n \in \mathbb{N}$ el $(2n+1)$-bloque central de y es la imágen bajo Φ del $(2n+1)$-bloque central de algún punto $x^{(n)}$ de X, es decir $\Phi(x^{(n)}_{[-n,n])}) = \phi(x^{(n)})_{[-n,n]} = y_{[-n,n]}$.

Usaremos a los $x^{(n)}$ para generar un punto $x \in X$ tal que $\phi(x) = y$ Primero consideremos la coordenada cero. Formamos el conjunto $S_0 = \{x \in X | \Phi(x_0) = y_0\}$ que es infinito porque la cantidad de sucesiones con el mismo símbolo central son infinitas (consideramos $|\mathcal{A}| \geq 2$). Estos elementos son los $x^{(0)}$. De allíseguimos con $S_1 = \{x \in X | \Phi(x_{[-1,1]}) = y_{[-1,1]}\}$. Vemos que $S_1 \subseteq S_0$ porque x_0 es subbloque de $x_{[-1,1]}$ pero S_1 sigue siendo infinito. Para cada $k \in \mathbb{N}$ encontramos su respectivo $S_k \subseteq S_{k-1}$ y definimos la sucesión x tal que $x[-k,k] = x^{(n)}_{[-k,k]}$ para cada $n \in S_k$. Vemos que el bloque $x_{[-k,k]}$ tiene como $(2n-1)$-bloque central a $x_{[-k+1,k-1]}$ y que además todo bloque de X ocurre en algún $x_{[-k,k]} = x^{(n)}_{[-k,k]} \in \mathcal{B}$ por lo que $x \in X$. Luego, para cada $k \geq 0$ y $n \in S_k$ tal que $n \geq k$ tenemos $\Phi(x_{[-k,k]}) = \Phi(x^{(n)}_{[-k,k])}) = \phi(x^{(n)})_{[-k,k]} = y_{[-k,k]}$ por lo que $\phi(x) = y$. Esto prueba que $\phi(X) \supseteq \mathsf{X}_{\mathcal{L}^c}$ □

Este es el argumento de diagonalización de Cantor. Usándolo también podemos demostrar que la función inversa de un código de bloques es un código de bloques.

Teorema 1.2. *Un código de bloques entre dos subshifts que es inyectivo y suprayectivo tiene una inversa que es un código de bloques y por lo tanto es una conjugación.*

Demostración. Dados X, Y shifts. Si $\phi : X \to Y$ podemos recodificar a ϕ de tal forma que sea un 1-código por 1.7. Sea Φ el código de bloques que induce ϕ. Sea $\psi : Y \to X$ la función inversa punto por punto de ϕ. Observamos que $y = \phi(x)$ para un único x y que $\phi(\sigma_X(x)) = \sigma_Y(\phi(x))$ por lo que $\sigma_X(\psi(y)) = \sigma_X(x) = \psi(\phi(\sigma_X(x))) = \psi(\sigma_Y(\phi(x))) = \psi(\sigma_Y(y))$ así que ψ conmuta con el shift. Por la proposicion 1.5 necesitamos una $n \geq 0$ tal que el $(2n+1)$-bloque central de cada y determine la coordenada $\psi(y)_0$ de su imágen. Supongamos que para cada $n \geq 0$ hay dos puntos $y^{(n)}, \widehat{y}^{(n)}$ de Y de tal manera que $y^{(n)}_{[-n,n]} = \widehat{y}^{(n)}_{[-n,n]}$ pero que $\psi(y^{(n)}_0) \neq \psi(\widehat{y}^{(n)}_0)$. Sean $x^{(n)} = \psi(y^{(n)}), \widehat{x}^{(n)} = \psi(\widehat{y}^{(n)})$ Podemos hacer una suposición de que la diferencia entre $x^{(n)}$ y $\widehat{x}^{(n)}$ se encuentra en su coordenada cero de donde $x^{(n)}_0 = \psi(y^{(n)}_0) = a$ y $\widehat{x}^{(n)}_0 = \psi(\widehat{y}^{(n)}_0) = b$ con $a \neq b$ y podemos formar el conjunto infinito S_0 con las n que cumplen tal condición. De allí pasamos a los 3-bloques formando $S_1 \subseteq S_0$ tal que $x^{(n)}_{[-1,1]}$ son iguales para toda $n \in S_1$ y los $\widehat{x}^{(n)}_{[-1,1]}$ son iguales para toda $n \in S_1$. De la misma forma generamos a los conjuntos $S_k \subseteq S_{k-1}$ de forma que $x^{(n)}_{[-k,k]}$ son iguales para toda $n \in S_k$ y los $\widehat{x}^{(n)}_{[-k,k]}$ son iguales para toda $n \in S_k$. Esto nos permite, como en la prueba anterior, construir puntos x y $\widehat{x}$ de forma que $x_{[-k,k]} = x^{(n)}_{[-k,k]}$ y $\widehat{x}_{[-k,k]} = \widehat{x}^{(n)}_{[-k,k]}$ para toda $n \in S_k$. Notamos que $x_0 = a \neq b = \widehat{x}_0$ así que $\widehat{x} \neq x$. Pero dado $n \in S_k$ y $n \geq k$ tenemos la siguiente igualdad $\Phi(x_{[-k,k]}) = \Phi(x^{(n)}_{[-k,k]}) = \phi(x^{(n)}_{[-k,k]}) = y^{(n)}_{[-k,k]} = \widehat{y}^{(n)}_{[-k,k]} = \phi(\widehat{x}^{(n)}_{[-k,k]}) = \Phi(\widehat{x}^{(n)}_{[-k,k]}) = \Phi(\widehat{x}_{[-k,k]})$ pero esto implica que $\phi(x) = \phi(\widehat{x})$ por lo que nuestra función no sería inyectiva. Esto muestra que la ventana debe existir y ψ debe ser un código de bloques.

□

Enfocaremos nuestra atención al problema de la conjugación por lo que las siguientes dos proposiciones son importantes.

Proposición 1.8. *Dados $\phi : X \to Y$ y $\psi : Y \to Z$ códigos de bloques, $\psi \circ \phi : X \to Z$ es entonces también código de bloques y además, si ϕ y ψ son códigos factores entonces $\psi \circ \phi$ es un código factor, al igual que para encajes y conjugaciones.*

Demostración. Usaremos la proposición 1.5 para probar que $\psi \circ \phi : X \to Z$ es un código de bloques. Sabemos que $\phi \circ \sigma_X = \sigma_Y \circ \phi$ y que $\psi \circ \sigma_Y = \sigma_Z \circ \psi$ así que $\psi \circ \phi(\sigma_X(x)) = \psi(\phi(\sigma_X(x))) = \psi(\sigma_Y(\phi(x))) = \sigma_Z(\psi(\phi(x))) = \sigma_Z(\psi \circ \phi(x))$. Además sea $N \in \mathbb{N}$ aquella tal que $y_0 = \Psi(x_{[-N,N]})$ y $K \in \mathbb{N}$ aquella tal que $z_0 = \Phi(z_{[-K,K]})$, entonces z_0 depende únicamente de $x_{[-N-K,N+K]}$ pues $\Psi \circ \Phi(x_{[-N-K,N+K]}) = \Psi(\Phi(x_{[-N-K,N-K]})\Phi(x_{[-N-K+1,N+1]})...\Phi(x_{[-N,N+K]})) = \Psi(y_{-K}y_{-K+1}...y_K) = \Psi(y_{[-K,K]} = z_0$. Por esto $\psi \circ \phi$ es un código de bloques. Al ser particularmente funciones, la preservación de la inyectividad y suprayectividad son heredadas naturalmente. □

Proposición 1.9. *La conjugación $\cong$ entre espacios shift es una relación de equivalencia.*

Demostración. Sean X, Y, Z espacios shift. (a) $X \cong X$ porque definimos $\Phi : X \to X$ tal que $\Phi(x_i) = x_i$ entonces $\phi = Id_X$ es una conjugación de X. (b) Sea $X \cong Y$ entonces existe $\phi : X \to Y$ código de bloques biyectivo. Pero por el teorema 1.2 tenemos una inversa $\phi^{-1} : Y \to X$ que ademas es un código de bloques por lo que $Y \cong X$ (c) Sea $X \cong Y, Y \cong Z$ tenemos $\phi : X \to Y$ y $\psi : Y \to Z$ por la proposición anterior sabemos que $\psi \circ \phi : X \to Z$ es una conjugación de donde $X \cong Z$ □

1.6. Metrizando los espacios shift

A los espacios shift se les puede ver como espacios topológicos y podemos definir una métrica en ellos. Estas construcciones pueden ser muy útiles porque nos acercan a herramientas poderosas de las matemáticas que pueden facilitar las pruebas y nos ayudan a entender mejor su comportamiento. Para definir una métrica nos interesa una función que logre capturar la noción de cercanía dentro del espacio. En este caso, si los puntos coinciden en su bloque central diremos que son cercanos.

Definición 1.13. Sea $X \subseteq \mathcal{A}^{\mathbb{Z}}$ un espacio shift. Definirmos para $\mathcal{A}^{\mathbb{Z}}$ la función $\rho : \mathcal{A}^{\mathbb{Z}} \times \mathcal{A}^{\mathbb{Z}} \to \mathbb{R}^+$ llamada la *métrica de cantor* tal que

$$\rho(x,y) = \begin{cases} 2^{-k} & \text{si } x \neq y \text{ y } k \geq 0 \text{ es el máximo tal que } x_{[-k,k]} = y_{[-k,k]} \\ 0 & \text{si } x = y \end{cases}$$

Es una métrica porque
(a) $\rho(x,x) = 0$
(b) $\rho(x,y) = 2^{-k}$ implica $x_{[-k,k]} = y_{[-k,k]}$ de donde $y_{[-k,k]} = x_{[-k,k]}$ y por ello $\rho(y,x) = 2^{-k} = \rho(x,y)$ y además
(c) si $\rho(y,z) = 2^{-k}$ entonces $y_{[-l,l]} = z_{[-l,l]}$ y si $m =$mín$\{k,l\}$ entonces $x_{[-m,m]} = z_{[-m,m]}$ y $\rho(x,z) \leq 2^{-m} \leq 2^{-k} \leq 2^{-k} + 2^{-l} = \rho(x,y) + \rho(y,z)$

Definición 1.14. Sea X un espacio shift. Sea $u \in \mathcal{B}(X)$ y $k \in \mathbb{Z}$ definimos el *cilindro de u trasladado por k* como $C_k(u) = C_k^X(u) = \{x \in X | x_{[k,k+|u|-1]} = u\}$, es decir el conjunto de puntos en los que ocurre u empezando en el punto k.

Para $x \in X$ recordemos que las *bolas abiertas de radio r con centro en x* consisten en $B_r(x) = \{y \in X | \rho(x,y) < r\}$ con $r \in \mathbb{R}$. Las bolas abiertas son una reescritura de los cilindros alrededor de la coordenada cero pues $C_{-n}(x_{[-n,n]}) = B_{2^{(-(n-1))}}(x)$. Además si $x \in C_k(u)$ y $n =$ máx$\{|k|, |k+|u|-1|\}$ entonces $B_{2^{(-(n-1))}}(x) \subseteq C_k(u)$ probando que los cilindros son conjuntos abiertos. Un cilindro es cerrado porque $C_k(u)^c = \bigcup_b \{C_k(b) | b$ bloque de $X, |b| = |u|$ y $b \neq u\}$ es una unión finita de conjuntos abiertos.

Teorema 1.3. *La topología inducida por la métrica de cantor ρ es la misma que la que toma como base de sus abiertos a $\tau = \{C_k(u) | u$ bloque de $X\}$.*

Demostración. Sea un abierto de la topología inducida por la métrica de Cantor $B_{2^{-l}}(x)$ Buscamos un $k \geq 0, u \in \mathcal{B}(x)$ de tal manera que $C_k(u) \subseteq B_{2^{-l}}(x)$. Gracias a la observación del párrafo anterior tenemos que con $u = x_{[-l,l]}$ y $k = -l$ tenemos que $C_{-l}(x_{[-l,l]}) = \{x \in X | x_{[-l,l]}\} = B_{2^{-l}}(x)$. Para encontrar el radio de la bola basta también tomar $r = l$. □

Definición 1.15. Una función $\phi : X \to Y$ es *continua* si siempre que $x_n \to x$ en X entonces $\phi(x_n) \to \phi(x)$ en Y. Si ϕ es continua, inyectiva, suprayectiva y tiene una inversa contínua entonces llamamos a ϕ un *homeomorfismo*.

Nos gustaría resaltar que en sistemas dinámicos tambien puede definirse un concepto parecido de nombre muy similar. Un *homomorfismo* es una función $\theta : X \to Y$ contínua que

conmuta con la función shift, i.e $\theta \circ \sigma_X = \sigma_Y \circ \theta$. Durante este trabajo la usaremos en la sección de gráficas pero advertimos que es fácil confundir los nombres.

Teorema 1.4. *(Curtis-Lyndon-Hedlund para shifts) Supongamos que tenemos (X, σ_X) y (Y, σ_Y) dos espacios shift y que $\phi : X \to Y$ es una función (no necesariamente contínua). Entonces ϕ es un código de bloques si y sólo si es un homomorfismo.*

Demostración. Primero probaremos que los códigos de bloques son funciones continuas. Para esto sea un código de N-bloques $\phi : X \to Y$ de memoria m y anticipación a donde $m+a = N$. Sin pérdida de generalidad supongamos que $a = 0$. Tomemos un punto $x^{(0)} \in X$ y vamos a ver que para cualquier $\epsilon \geq 0$ existe una $\delta \geq 0$ tal que si $x^{(1)} \in B_\delta(x^{(0)})$ entonces $\phi(x^{(1)}) \in B_\epsilon \phi(x^{(0)})$. Sea $\epsilon > 0$, ya sabemos que existe $K \in \mathbb{N}$ tal que $2^{-K} \leq \epsilon$ y además por definición de la distancia de Cantor, aquellos puntos en Y cuyos bloques centrales de tamaño $2K+1$ son iguales al bloque central de tamaño $2K+1$ de $\phi(x^{(0)})$, estarán en $B_\epsilon \phi(x^{(0)})$, que es lo que deseamos. Sabemos que un N-bloque en nuestro punto $x^{(1)}$ se convierte en una coordenada en $\phi(x^{(1)})$ por lo que si queremos igualar a lo menos $2K+1$ coordenadas tras haber sido transformado nuestro punto $x^{(1)}$ por ϕ necesitamos calcular $R = \text{máx}\{K, N\}$ para asegurarnos que tomamos la "ventana"tan grande como la necesitamos y luego movernos en una ventana de $2K+1$ símbolos alrededor de la coordenada $x_0^{(1)}$. Esto nos lleva tomar $\delta = 2^{-(2(K+R)+1)}$ y así tenemos que $\Phi(x_{[-R-K,R+K])}$ tiene como subbloque a $\Phi(x_{[-N-K,K]}) = \Phi(x_{[-N-K,-K]})\Phi(x_{[-N-K+1,-K+1]})\Phi(x_{[-N-K+2,-K+2]})...\Phi(x_{[K-N,K]}) = (y_{-K}y_{-K+1}...y_K) = y_{[-K,K]} = \phi(x)_{[-K,K]}$ que es el bloque central de tamaño $2K+1$ del punto $x^{(0)}$ lo cual es justamente lo que queríamos. Ya habíamos visto que cualquier código de bloques conmuta con el shift y por lo tanto ϕ es homomorfismo. Ahora supongamos que ϕ es homomorfismo. Sea $\mathcal{A}$ el alfabeto de X y $\mathfrak{A}$ el de Y. Para cada $b \in \mathfrak{A}$ sea $C_0(b)$ el cilindro $\{y \in Y | y_0 = b\}$. Los conjuntos $C_0(b), b \in \mathfrak{A}$ son ajenos porque todos los elementos de dos cilindros diferentes difieren al menos en la coordenada y_0, y son compactos porque si tomamos una sucesión convergente esta está formada por puros puntos de Y que tienen su coordenada central igual, y por lo tanto, el límite debe tener esa misma primera coordenada. Por resultados de topología sabemos que sus imágenes inversas $E_b = \phi^{-1}(C_0(b))$ son ajenas y compactas en X. Luego existe una $\delta \geq 0$, tal que puntos en conjuntos diferentes E_b están por lo menos a distancia $\delta = inf\{\rho(x, y) | x \in E_{b_1}, y \in E_{b_2}\}$. Escogemos n tal que $2^{-n} \leq \delta$. Entonces cada par de puntos $x, x' \in X$ tales que $x_{[-n,n]} = x'_{[-n,n]}$ deben estar en el mismo conjunto E_b tal que $\phi(x)_0 = b = \phi(x')_0$. Luego la coordenada cero de la imagen depende de un $(2n+1)$-bloque central de x y por la proposición 1.5, ϕ es un código de bloques. □

Teorema 1.5. *Un subconjunto X de $\mathcal{A}^{\mathbb{Z}}$ es un espacio shift si y sólo si es shift invariante y cerrado.*

Demostración. Primero supongamos que X es un espacio shift y demostremos que es shift

invariante y cerrado. Como X es espacio shift está definido a partir de un conjunto de palabras prohibidas $\mathcal{F}$. Buscamos una doble contención de X y $\sigma(X)$ pero basta observar que todo bloque de X es bloque de $\sigma(X)$ y viceversa por lo que $\mathcal{L}(X) = \mathcal{L}(\sigma(X))$. Gracias a la proposición 1.1 en su apartado (3) podemos concluir que $X = \sigma(X)$. Para probar que X es cerrado usaremos el teorema que nos asegura que toda sucesion convergente en un cerrado converge a un punto de él. Sea $x^{(n)} \subseteq X$ sucesion convergente al punto x. Supongamos que $x \notin X$ por lo que existe un $b \in \mathcal{F}$ tal que $b = x_{[l,l+k]}$. Sin embargo sabemos que para $\epsilon = 2^{-|l+k|}$ existe n_0 tal que $\rho(x^{(n_0)}, x) < \epsilon$ pero además $b = x^{(n_0)}_{[l,l+k]}$ lo que implica que $x^{(n_0)} \notin X$ que es una contradicción. Por lo tanto X es cerrado.

Ahora supongamos que X es cerrado y shift invariante y probemos que es espacio shift. Como X es cerrado, $\mathcal{A}^{\mathbb{Z}} - X$ es abierto. Luego para cada $y \in \mathcal{A}^{\mathbb{Z}} - X$ hay una $k = k(y)$ tal que si u_y es el bloque $y_{[-k,k]}$ entonces $C_k^{\mathcal{A}^{\mathbb{Z}}}(u_y) \subseteq \mathcal{A}^{\mathbb{Z}} - X$. Sea $\mathcal{F} = \{u_y | y \in \mathcal{A}^{\mathbb{Z}} - X\}$. Entonces $X = \mathsf{X}_{\mathcal{F}}$ porque dado $x \in X$ ningún bloque de x es una u_y porque precisamente estos pertenecen a su complemento en $\mathcal{A}^{\mathbb{Z}}$ y si $x \in \mathsf{X}_{\mathcal{F}}$ entonces ninguno de sus bloques pertenece a $\mathcal{A}^{\mathbb{Z}} - X$ por lo que todos pertenecen a X y luego $x \in X$. □

1.7. Shifts de tipo finito

Definición 1.16. Los espacios shift X para los que existe un conjunto *finito* $\mathcal{F}$ de palabras prohibidas tal que $X = \mathsf{X}_{\mathcal{F}}$ se llaman *Shifts de Tipo Finito o STF.*

Ya conocemos varios, por ejemplo el shift completo es de tipo finito con $\mathcal{F} = \emptyset$, el shift áureo tiene a $\mathcal{F} = \{11\}$ y el shift del ejemplo 1.8 también tiene un conjunto finito de palabras prohibidas.

Supongamos que $X \subseteq \mathcal{A}^{\mathbb{Z}}$ es un shift de tipo finito con $\mathcal{F}$ finito y $X = \mathsf{X}_{\mathcal{F}}$. Sea N la mayor de todas las longitudes de los bloques de $\mathcal{F}$. Podemos formar la colección $\mathcal{F}_N$ de todos los bloques de longitud N que tienen a un elemento de $\mathcal{F}$ como subbloque. Entonces $\mathsf{X}_{\mathcal{F}} = X = \mathsf{X}_{\mathcal{F}_N}$ y todos los bloques de $\mathcal{F}_N$ tienen la misma longitud. Por ejemplo si $\mathcal{A} = \{0, 1\}$ y $\mathcal{F} = \{10, 111\}$ entonces $\mathcal{F}_3 = \{101, 100, 010, 110, 111\}$. Este procedimiento se asume como realizado muchas veces para simplificar los razonamientos. Si todos los bloques de $\mathcal{F}$ tienen longitud N para evaluar si un punto x pertenece a $X = \mathsf{X}_{\mathcal{F}}$ basta con "deslizar" x por una ventana de tamaño N y ver si $x_{[i,i+N-1]} \notin \mathcal{F}$ para cada $i \in \mathbb{Z}$. Este criterio es útil también para evaluar si un shift es de tipo finito.

Ejemplo 1.17. El shift par no es de tipo finito pues si tuviéramos una ventana de tamaño N

para evaluar los bloques entonces el punto $x = 0^\infty 10^{2N+1} 10^\infty$ no podría ser reconocido pues necesitamos ver el bloque $0^{2N+1}1$ completo para decidir.

Definición 1.17. Un shift de tipo finito es de *M-pasos* o tiene *memoria M* si puede ser descrito por una colección de bloques prohibidos $\mathcal{F}$ en la que todo elemento tiene longitud $M+1$

Esta definición es motivada por las máquinas de Turing pues si tuviéramos una máquina que decide si cierto punto x pertenece a nuestro shift X ésta necesitaria recordar M símbolos anteriores al que está en su marcador para saber si x contiene un bloque prohibido. Notemos que un shift de memoria 0 solo puede ser el shift completo pues estaríamos descartando letras del alfabeto si descartásemos algo. Los shift de 1-paso simplemente tienen prohibiciones sobre los vecinos inmediatos de los símbolos, por lo que admiten una representación como digráfica que veremos más adelante.

Proposición 1.10. *Si X es un shift de tipo finito entonces existe una $M \geq 0$ para la cual X es de memoria M*

Demostración. Este resultado se sigue del párrafo previo al ejemplo. Sólo hay que tener cuidado con el detalle de que dados todos los N tales que $\mathcal{F}_N$ (que consta de puros bloques de tamaño N) tomemos $M = \text{mín}\{N \in \mathbb{N} \,|\, X = \mathsf{X}_{\mathcal{F}_N}\} - 1$ porque la memoria M evalúa bloques de tamaño $M+1$. □

Teorema 1.6. *Un SFT X es de M-pasos si y sólo si siempre que $u, v, w \in \mathcal{B}(X)$ y $|v| \geq M$ entonces $uvw \in \mathcal{B}(X)$.*

Demostración. Supongamos primero que X es de M-pasos tal que $X = \mathsf{X}_{\mathcal{F}}$ para una $\mathcal{F}$ consistente en sólo $M+1$-bloques. Supongamos que $u, v, v, w \in \mathcal{B}(X)$ y $|v| = n \geq M$. Entonces hay puntos $x, y \in X$ con $x_{[-k,n]} = uv$ y $y_{[1,l]} = vw$ de tal forma que $x_{[1,n]} = y_{[1,n]} = v$ (en este caso estamos pensando que para cualquier bloque de x, podemos usar σ para hacerlo aparecer en la posición que queramos). El punto $z = x_{[-\infty,0]} v y_{[n+1,\infty]} \in X$ pues si algún bloque de $\mathcal{F}$ ocurriese en z entonces debe ocurrir en $x_{[-\infty,0]}v = x_{[-\infty,n]}$ ó en $vy_{[n+1,\infty]} = y_{[1,\infty]}$ pues $|v| = n \geq M$, lo que nos asegura que la cola de x no se intersecta con la de y lo cual contradiría que x y y pertenecen a X ambos. Ahora supongamos que para X shift sobre $\mathcal{A}$ existe M con la propiedad de que si $uv, vw \in \mathcal{B}(X)$ con $|v| \geq M$ entonces $uvw \in \mathcal{B}(X)$. Sea $\mathcal{F}$ el conjunto de todos los $M+1$-bloques de $\mathcal{A}$ que no aparecen en $\mathcal{B}_{M+1}(X)$. Se prueba que $X = \mathsf{X}_{\mathcal{F}}$ de donde X es SFT de M pasos. Si $x \in X$ entonces ninguno de sus bloques puede ocurrir en $\mathcal{F}$ por lo que $x \in \mathsf{X}_{\mathcal{F}}$ y $X \subseteq \mathsf{X}_{\mathcal{F}}$. Ahora sea $x \in \mathsf{X}_{\mathcal{F}}$. Luego $x_{[0,M]}$ y $x_{[1,M+1]}$ se traslapan en M símbolos por lo que $x_{[0,M+1]} \in \mathcal{B}(X)$ (usamos el teorema anterior con $u = x_0, v = x_{[1,M]}, w = x_{M+1}$). Ahora $x_{[2,M+2]} \in \mathcal{B}(X)$ y se traslapa en M símbolos con $x_{[0,M+1]}$ por lo que $x_{[2,M+2]} \in \mathcal{B}(X)$. Repitiendo este proceso en cada dirección tenemos que $x_{[-k,l]} \in \mathcal{B}(X)$para toda $k, l \in \mathbb{N}$ y por el corolario 1.1 tenemos que $X \supseteq \mathsf{X}_{\mathcal{F}}$ □

Teorema 1.7. *Un espacio shift que es conjugado a un espacio shift de tipo finito es él mismo un shift de tipo finito*

Demostración. Supongamos X es conjugado a un shift de tipo finito Y y Φ la función de bloques que induce la conjugación y Ψ la que induce su inversa. Tratamos de usar nuestro teorema 1.6 tomando bloques cuyas imágenes bajo Φ se traslapan lo suficiente para poder pegarlos. Cuando regresamos con Ψ estos bloques se acortan de manera que tenemos que hacer un truco para alargarlos. De acuerdo con el teorema 1.6 necesitamos una $M \geq 1$ tal que si $v \in \mathcal{B}(X)$ con $|v| \geq M$ y $uv, vw \in \mathcal{B}(X)$ entonces $uvw \in \mathcal{B}(X)$ Sea ϕ la conjugación que induce Φ y $\psi = (\phi)^{-1}$ su inversa inducida por Ψ. Podemos alargar la ventana que sea necesaria para que ϕ y ψ tengan la misma memoria y anticipación l. Como $\psi(\phi(x)) = x$ para todo $x \in X$ entonces la función $\Psi \circ \Phi : \mathcal{B}_{4l+1}(X) \to \mathcal{B}_1(X)$ simplemente escoge el símbolo central de cada bloque ($\Psi \circ \Phi$ mapea un bloque de tamaño $4l + 1$ en uno de $2l + 1$ por el argumento dado en 1.4, para el cual es asignado un único símbolo bajo Ψ). Como Y es de tipo finito por el teorema 1.6 existe una $N \geq 1$ de tal forma que dos bloques que se traslapen en por lo menos N lugares pueden pegarse a lo largo de su intersección para formar un bloque de Y Así, sea $M = N + 4l$. Para verificar que cumplimos las condiciones del teorema 1.6 y probar que X es de tipo finito, sea $uv, vw \in \mathcal{B}(X)$ con $|v| \geq M$. Por la proposicion 1.1 existen palabras $s, t \in \mathcal{B}_{2l(X)}$ tales que $suv, vwt \in \mathcal{B}(X)$. Como cada (4l+1)-bloque está en $\mathcal{B}(X)$ el razonamiento de arriba nos muestra (aunque no sabemos que $suvwt \in \mathcal{B}(X)$) que $\Psi \circ \Phi(suvwt) = uvw$. Ahora $\Phi(suv) = u'\Phi(v)$ y $\Phi(vwt)$ donde $u', w' \in \mathcal{B}(Y)$ y $|\Phi(v)| = |v| - 2l \geq N$ así que $u'\Phi(v)$ y $\Phi(v)w'$ pueden pegarse para conseguir $u'Phi(v)w' \in \mathcal{B}(Y)$. Además $uvw = \Psi(\Phi(suvwt)) = \Psi(u'\Phi(v)w') \in \mathcal{B}(X)$ por lo que X es de tipo finito. □

1.8. Las gráficas y matrices de los espacios shift

En esta sección presentaremos una forma de construir espacios shift a partir de digráficas. Esta forma de presentar un shift tiene una gran ventaja: podemos asociar una matriz de adyacencia y aplicar resultados de álgebra lineal para obtener datos muy importantes del espacio. Todos los STFs pueden asociarse a una gráfica. Además esta gráfica nos da la pauta para relacionar el lenguaje de nuestro shift con un autómata con todos sus estados como estados finales que puede reconocerlo.

Definición 1.18. Una *digráfica* (durante este trabajo *gráfica*) consiste de un conjunto finito $\mathcal{V} = \mathcal{V}(G)$ de *vértices* o *estados* junto con un conjunto finito de *aristas* $\mathcal{E} = \mathcal{E}(G)$. Cada arista $e \in \mathcal{E}(G)$ inicia en un vértice denominado $i(e) \in \mathcal{V}(G)$ denominado *estado inicial* y

termina en $t(e) \in \mathcal{V}(G)$ denominado *estado terminal* (que puede ser igual que $i(e)$. Puede existir más de una arista entre un estado inicial y un estado terminal y al conjunto de ellos se le llama un conjunto de *aristas múltiples*. Una arista e con $i(e) = t(e)$ se llama un *loop*.

Para cada vértice I llamamos $\mathcal{E}_I = \mathcal{E}_I(G)$ al conjunto de aristas salientes desde I, es decir, $\mathcal{E}_I = \{e \in \mathcal{E} | i(e) = I\}$ y de igual manera $\mathcal{E}^I = \{e \in \mathcal{E} | t(e) = I\}$ es el conjunto de aristas entrantes. El número $|\mathcal{E}_I|$ es el grado exterior de I y $|\mathcal{E}^I|$ es el grado interior de I.

La siguiente figura ilustra una gráfica G con vértices $\mathcal{V} = I, J$ y $\mathcal{E}$ con ocho aristas donde f, g son loops y e tiene como inicial al vértice $I = i(e)$ y como terminal al vértice o estado $J = t(e)$.

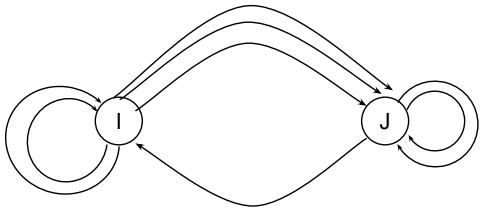


Figura 1.1: Una gráfica típica

Definición 1.19. Sean G y H gráficas. Un *homomorfismo de G en H* consiste en un par de funciones $\partial\Phi : \mathcal{V}(G) \to \mathcal{V}(H)$ y $\Phi : \mathcal{E}(G) \to \mathcal{E}(H)$ tales que $i(\Phi(e)) = \partial\Phi(i(e))$ y $t(\Phi(e)) = \partial\Phi(t(e))$ para todos los vértices $e \in \mathcal{E}(G)$. En caso de existir se denota $(\partial\Phi, \Phi) : G \to H$. Un homomorfismo de gráficas es un *encaje de gráficas* si ambos $\partial\Phi, \Phi$ son inyectivos y es un *isomorfismo de gráficas* si ambas funciones $\partial\Phi, \Phi$ son a la vez inyectivas y suprayectivas. En ese caso escribimos $\partial\Phi, \Phi : G \cong H$ y dos gráficas son isomorfas si hay un isomorfismo entre ellas.

Para muchas aplicaciones necesitaremos definir un orden en nuestro conjunto de vértices. Así pues siempre que no sea expícito como en el caso en que escribimos $\mathcal{V} = \{I_1, I_2, I_3, ...I_n\}$ tomaremos el orden numérico o alfabético del conjunto. Una vez hecho esto podemos definir una matriz

Definición 1.20. Sea G gráfica con conjunto de vértices $\mathcal{V}$. Para $I, J \in \mathcal{V}$ sea A_{IJ} el número de aristas en G con estado inicial I y estado terminal J. Entonces la *matriz de adyacencia de* G es $\mathsf{A} = [A_{IJ}]$ y su formación a partir de G se escribe como $A = \mathsf{A}(G) = \mathsf{A}_G$. Por ejemplo la matriz de adyacencia de la gráfica anterior es

$$\mathsf{A}_G = \begin{bmatrix} 2 & 3 \\ 1 & 2 \end{bmatrix}.$$

Hemos tomado el orden alfabético para los vértices pero realmente si este orden se viera alterado sus consecuencias en la matriz A_G no son graves, basta conjugarla por la matriz de permutaciones P que tiene para cada símbolo J un número 1 en el lugar $A_{JJ'}$ donde J' es el lugar de J en el nuevo orden y cero en toda otra entrada de la fila J. Así $A' = PAP^{-1}$ es la nueva matriz del G con sus vértices en el orden nuevo.

Es interesante que para cualquier matriz con entradas en los números naturales también podemos construir una gráfica.

Definición 1.21. Sea $A = [A_{IJ}]$ una matriz de $r \times r$ con entradas enteras no negativas. Entonces la *gráfica* de A es la gráfica $G = \mathsf{G}(A) = \mathsf{G}_A$ con conjunto de vértices $\mathcal{V}(G) = \{1, 2, ..., r\}$ y con tantas aristas distintas como A_{IJ} entre el estado inicial I y el terminal J.

Las operaciones: generar la matriz a partir de la gráfica A y generar la gráfica a partir de la matriz G son inversas la una a la otra de tal forma que $A = \mathsf{A}(\mathsf{G}(A))$ y $G \cong \mathsf{G}(\mathsf{A}(G))$.

Ahora sí podemos definir un shift de tipo finito para cada matriz de la siguiente forma:

Definición 1.22. Sea G una grárfica con conjunto de aristas $\mathcal{E}$ y matriz de adyacencia A. El *shift de aristas* $\mathsf{X}_A = \mathsf{X}_G$ es el shift sobre el alfabeto $\mathcal{A} = \mathcal{E}$ dado por $\mathsf{X}_A = \mathsf{X}_G = \zeta = (\zeta_i)_{i\in\mathbb{Z}} \in \mathcal{E}^{\mathbb{Z}} \,|\, \mathsf{t}(\zeta_i) = \mathsf{i}(\zeta_{i+1}) \forall i \in \mathbb{Z}$. La función shift en X_A o X_G se llama la función shift de aristas y la escribimos σ_G o σ_A.

De acuerdo con esta definición una sucesión bi-infinita está en X_G siempre que el estado terminal de cada arista es el estado inicial de la siguiente. Esto, en la gráfica G es llamado describir una *caminata bi-infinita* o un *paseo bi-infinito.*

Proposición 1.11. *Si G es una gráfica con matriz de adyacencia A entonces el shift de aristas $\mathsf{X}_A = \mathsf{X}_G$ es un shift de tipo finito de 1-paso.*

Demostración. Sea $\mathcal{A} = \mathcal{E}$ el alfabeto de X_G. Consideremos la colección finita $\mathcal{F} = \{e, f | e, f \in \mathcal{A}, \mathsf{t}(e) \neq \mathsf{i}(f)\}$ de 2-bloques sobre $\mathcal{A}$. De acuedo con la definición 1.21 un punto $\zeta \in \mathcal{A}^{\mathbb{Z}}$

está en X_G exactamente cuando ningún bloque de $\mathcal{F}$ ocurre en ζ. Por ello $\mathsf{X}_\mathcal{F} = \mathsf{X}_G$ y además X_G es de tipo finito porque la cantidad de 2-bloques es finita. Como todos los bloques prohibidos tienen longitud 2, $\mathsf{X}_\mathcal{F}$ es de 1-paso. □

Ejemplo 1.18. Sea $r \geq 1$ y A la matriz de adyacencia de 1×1 $A = [r]$. Luego G_A tiene un vértice y r loops en el mismo vértice. Si a cada una de las aristas las llamamos 0,1,...,r-1 entonces X_G es el r-shift completo. El cual denotamos por $\mathsf{X}_{[r]}$

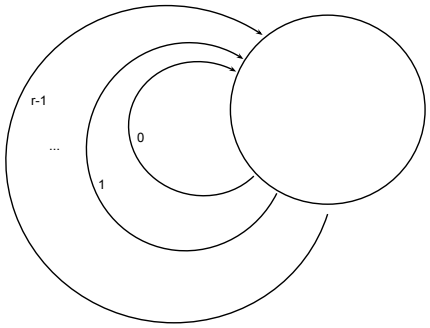


Figura 1.2: La gráfica para el shift r-completo

Hay una restricción que nos conviene añadir a las gráficas para simplificarnos la vida. Éstas pueden tener vértices que nunca aparecerán en X_G pues si el vértice sólo tiene una arista de entrada pero ninguna "sale" entonces este se convierte en un callejón sin salida y no aparece nunca en el shift de sucesiones bi-infinitas porque no puede formar parte de una caminata bi-infinita. Diremos que un vértice $I \in \mathcal{V}$ está *varado* si ninguna arista lo tiene como vértice inicial o si ninguna arista lo tiene como vértice terminal. En términos de la matriz de adyacencia esta condición se traduce a que la I-ésima fila o columna tengan solo entradas nulas.

Definición 1.23. Una gráfica es *esencial* si ningún vértice en ella está varado.

Definición 1.24. Sea G gráfica. Decimos que H es una subgráfica de G siempre que $\mathcal{V}(H) \subseteq \mathcal{V}(G)$, $\mathcal{E}(H) \subseteq \mathcal{E}(G)$ y además las aristas de H comiencen y terminen en los mismos vértices que en G.

Proposición 1.12. *Si G es una gráfica, entonces existe una única subgráfica H de G tal que H es esencial y $\mathsf{X}_G = \mathsf{X}_H$.*

Demostración. Sea $\mathcal{E}(H)$ el conjunto de todas las aristas $e \in \mathcal{E}(G)$ que participan en una caminata bi-infinita en G y sea $\mathcal{V}(H) = \{\mathsf{i}(e) | e \in \mathcal{E}(H)\}$ el conjunto de vértices visitados durante alguna caminata. Entonces H es una subgráfica de G y por definición cada caminata bi-infinita en G es una caminata en H. Luego, H es esencial y $\mathsf{X}_G = \mathsf{X}_H$. Para probar que H es único, primero observemos que la definición de H la hace la mayor subgráfica esencial de G pues solo retiramos vértices v que no participaban en una caminata bi-infinita y esto sólo puede ocurrir si son aislados, pues en caso de no ser aislados tomamos un camino l al que pertenezcan el vertice $q = l_q$ de salida de la arista que entra a v y otro m_r que tiene a r aquel vértice al que llega la arista que sale de v y pasamos por el vértice aislado concatenando $l_{(-\infty,q]} v m_{[r,\infty)}$. Luego si H' es subgráfica esencial de G y $H' \neq H$ existe una arista de H que no está en H'. Como cada arista de H ocurre en X_H entonces esta arista que quitamos sí afecta la construcción del shift haciendo que falten justamente los puntos para los que su caminata pasa por esa arista, esto prueba $\mathsf{X}_{H'} \neq \mathsf{X}_H$, estableciendo la unicidad de H □

Proposición 1.13. *Las las gráficas esenciales G y H son isomorfas si y sólo si $\mathsf{X}_G \cong \mathsf{X}_H$.*

Demostración. Sean G, H gráficas isomorfas con $\partial\Phi : \mathcal{V}(G) \to \mathcal{V}(H)$ y $\Phi : \mathcal{E}(G) \to \mathcal{E}(H)$ tales que $\mathsf{i}(\Phi(e)) = \partial\Phi(\mathsf{i}(e))$ y $\mathsf{t}(\Phi(e)) = \partial\Phi(\mathsf{t}(e))$ para todos los vértices $e \in \mathcal{E}(G)$ y ambas biyectivas. Sea $\mathcal{A} = \mathcal{E}(G)$ el alfabeto de X_G y $\mathfrak{A} = \mathcal{E}(H)$ el alfabeto de X_H. Entonces podemos generar la conjugación con el 1-código de bloques ψ inducido por $\Psi : \mathcal{A} \to \mathfrak{A}$ tal que $\Psi(v) = \Phi(v)$ y al ser Φ biyectiva tenemos naturalmente que su inversa existe y ademas es un 1-código de bloques dado por $\Phi^{-1}(\psi(v)) = v$ donde $\psi(v)$ es un 1-bloque. Si tenemos dos shifts de tipo finito $\mathsf{X}_G, \mathsf{X}_H$ conjugados por $\theta : \mathsf{X}_G \to \mathsf{X}_H$ podemos suponer que θ es de 1-bloques y que su inversa también. Esto induce dos funciones sobre $G = \mathsf{G}(\mathsf{X}_G)$ y $H = \mathsf{G}(\mathsf{X}_H)$ como sigue: : $laprimera\mathcal{E}(G) \to \mathcal{E}(H)$ tal que $\theta_{\mathcal{E}}(e) = \theta(e)$ y la segunda, para la que recordemos que, como no hay vértices aislados, cada uno tiene una arista de entrada i y una de salida o de tal forma que podemos definir la función de vértices a partir de sus aristas incidentes, $\theta_{\mathcal{V}} : \mathcal{V}(G) \to \mathcal{V}(H)$ tal que $\theta_{\mathcal{V}}(v_{io}) = s_{\theta(i)\theta(o)}$ donde el primer subíndice representa la arista que entra y el segundo la que sale. Estas funciones heredan su biyectividad de θ y $\theta_{\mathcal{V}}(\mathsf{t}(v_{io}) = \theta(o) = t(\theta_{\mathcal{E}}(o))$ por lo tanto G y H son isomorfas. □

De ahora en adelante sólo se presta atención a gráficas esenciales pues ya tenemos una forma de reducir cualquier otra a una de ellas y vemos que son las que efectivamente generan espacios shift. Continuamos con una definición fundamental tanto para la sección de autómatas como para lo que sigue de este capítulo.

Definición 1.25. Un camino $\pi = e_1 e_2 ... e_m$ en una gráfica G es una sucesión finita de aristas e_i de G de tal forma que $\mathsf{t}(e_i) = \mathsf{i}(e_{i+1})$ para $1 \leq i \leq m-1$. La longitud del camino $|\pi| = m$ es el número de aristas que aparecen en él. El camino $\pi = e_1 e_2 ... e_m$ comienza en el vértice $i(\pi) = i(e_1)$ y termina en el vértice $t(\pi) = t(e_m)$ y π es un camino desde $i(\pi)$ hasta $t(\pi)$. Un

ciclo es un camino que comienza y termina en el mismo vértice. Un ciclo simple es aquel que no se intersecta a si mismo, es decir, $\mathsf{i}(e_1)\mathsf{i}(e_2)...\mathsf{i}(e_m)$ son distintos. Para cada vértice $I \in G$ existe también el camino vacío ϵ_I de longitud cero que comienza y termina en I.

Como nuestras gráficas son esenciales ahora, cada uno de los caminos no vacíos se corresponde con un bloque del shift X_G La matriz de adyacencia A nos da información sobre los caminos en la gráfica. Sea $\mathcal{E}_I^J$ la colección de caminos de longitud 1 que comienzan en I y terminan en J y tiene tamaño A_{IJ}. En particular A_{II} es el número de loops en el vértice I. El total de loops en una gráfica es entonces la traza $tr(A)$. Así podemos empezar a navegar estos oceanos con el álgebra lineal.

Proposición 1.14. *Sea G una gráfica con matriz de adyacencias A y sea $m \geq 0$.*
(1) El número de caminos de longitud m desde I hasta J es $(A^m)_{IJ}$,
la IJ-ésima entrada de A^m.
(2) El número de ciclos de longitud m en G es tr(A), la traza de A^m y ésto equivale al número de puntos en X_G de periodo m.

Demostración. (1) Para $m = 0$ es cierto pues $A^0 = Id$ y el único 0-camino posible es ϵ_I para toda I. Supongamos que es verdadero para $m = n$ y de allí que A^n son todos los posibles caminos de longitud n desde I hasta J. Nos fijamos en todos los vértices K de la gráfica (también podría ser J). Entonces la cantidad de aristas que hay desde cualquier K hasta J está dada por $\mathcal{E}_K^J = A_{JK}$ y además si tengo un camino de tamaño n hasta K entonces tengo un camino de tamaño $n+1$ hasta J. Dado que los caminos de cada vértice hasta K de tamaño n están dados por nuestra hipótesis de inducción por A_{IK}^n basta multiplicarlos por la cantidad de opciones que tengo para dar el último paso A_{KJ} de tal forma que $|n+1-$ caminos de I a $J| = (A_{IK}^n)(A_{KJ)} = A_{IJ}^{n+1}$ dada justamente por la regla de multiplicación matricial. Restaltamos que K también podría ser J y de todas formas agregamos el camino ϵ para alargar el camino una unidad aunque siga uniendo los mismos vértices.
(2) Gracias a la prueba anterior sabemos que los n caminos de un vértice K a él mismo estan dados por , sin embargo, hay que recordar que estos incluyen no solo ciclos simples sino todas las posibles combinaciones entre caminos triviales (sucesiones del camino ϵ) y caminos"reales". Entonces todos los posibles ciclos de tamaño n son $\sum_{K|K\in\mathcal{V}(G)} A_{KK}^n$ $= tr(A^n)$. Estos son los únicos puntos periódicos de X_G que podemos generar en el shift concatenando el camino indefinidamente hacia ambos lados. □

Como podemos ver, la información de la matriz sobre los puntos periódicos del shift es muy relevante y fácil de obtener. Para terminar presentamos la noción de irreducibilidad de los shifts. Este concepto es fundamental para poder aplicar la teoría de valores propios a las matrices de adyacencia.

Definición 1.26. Una gráfica G es *fuertemente conexa* si para cualquier par ordenado de vértices I, J existe un camino desde I hasta J. Una matriz es *irreducible* si para cada par de índices I, J existe una $n \geq 0$ tal que $A^n_{IJ} \geq 1$.

Podemos constatar que irreducibilidad en la matriz A_G de adyacencias es equivalente a conexidad fuerte en la gráfica G. Por tal motivo usaremos indistintamente el concepto de irreducibilidad y conexidad fuerte para las gráficas. El siguiente teorema completa la relación entre las últimas definiciones.

Teorema 1.8. *Una gráfica esencial es irreducible si y sólo si su shift por aristas es irreducible.*

Demostración. Sea G una gráfica irreducible y $\pi, \tau \in \mathcal{B}(\mathsf{X}_G)$. Supongamos que π termina en el vértice I y τ en el J entonces por la irreducibilidad de G hay un camino ω desde I hasta J y de allí que $\pi\omega\tau \in \mathcal{B}(\mathsf{X}_G)$. Ahora supongamos que G es esencial y es irreducible. Por ser G esencial dados I, J vértices hay dos aristas e, f tales que e comienza en I y f termina en J. Por irreducibilidad de X_G existe un bloque ω tal que $e\omega f \in \mathcal{B}(\mathsf{X}_G)$ entonces $e\omega f$ es un camino en G desde I hasta J y G es irreducible. □

Los espacios shift de aristas nos pueden parecer un tipo especial de shifts de tipo finito de memoria 1 pues para saber a qué vértice puedo ir sólo es necesario conocer en cuál me encuentro. Sin embargo, en esta sección probamos que podemos recodificar a cualquier shift de tipo finito a una versión por bloques de un shift de aristas.

Ejemplo 1.19. Sea $\mathcal{A} = \{0, 1\}$, $\mathcal{F} = \{11\}$ de tal forma que $\mathsf{X}_\mathcal{F}$ es el shift áureo del ejemplo 1.2. Afirmamos que no existe una gráfica G tal que $\mathsf{X}_\mathcal{F} = \mathsf{X}_G$ pues en caso de existir podríamos asumir que esta gráfica es esencial. Entonces G tendría exactamente dos aristas llamadas 0 y 1. Las únicas posibilidades son que G tenga un único vértice por lo que X_G se convertiría en el shift completo como hemos visto arriba o que G tuviera dos aristas lo cual forzaría a nuestro shift a tener únicamente dos puntos $(01)^\infty$ y $(10)^\infty$. En ningún caso $\mathsf{X}_\mathcal{F} = \mathsf{X}_G$.

Pero de todas maneras tenemos un teorema que nos aclara la relación que existe entre shifts de tipo finito y shifts por aristas. Recordemos que X es de M-pasos cuando $\mathcal{F}$ consta únicamente de $(M + 1)$-bloques.

Teorema 1.9. *Si X es un STF de M-pasos, entonces existe una gráfica G tal que $X^{[M+1]} = X_G$.*

Demostración. Primero hay que observar que un shift de tipo finito de memoria $M = 0$ es el shift completo y podemos tomar a G de tal forma que tenga un único vértice con un

auto-loop por cada letra del alfabeto que aparece en X. Así pues asumimos que $M \geq 1$. Definimos el conjunto de vértices de G como $\mathcal{V} = \mathcal{B}_M(X)$ que consta de todos los M-bloques que aparecen en X. Definimos el conjunto de aristas $\mathcal{E}$ como sigue: Supongamos que $I = a_1a_2...a_M$ y $J = b_1b_2...b_M$ son dos vértices de G. Si $a_2a_3...a_M = b_1b_2...b_{M-1}$ y si $a_1a_2...a_Mb_M = a_1b_1...b_M \in \mathcal{B}(X)$, entonces dibujamos exactamente una arista en G desde I hasta J llamada $a_1a_2...a_Mb_M = a_1b_1...b_M$. En otro caso no dibujamos arista alguna desde I hasta J. Por construcción tenemos que una caminata bi-infinita sobre nuestra nueva G es precisamente una sucesión de $(M+1)$-bloques en $\mathcal{B}_{M+1}(X)$ que se traslapan progresivamente y de allí $\mathsf{X}_G = X^{[M+1]}$. □

Ejemplo 1.20. Ahora observamos que nuestro ejemplo 1.19 no era codificable como un shift de aristas pero su configuración por 2-bloques sí es transformable en gráfica usando el procedimiento de la prueba anterior, como se puede ver en la figura 1.3 a continuación.

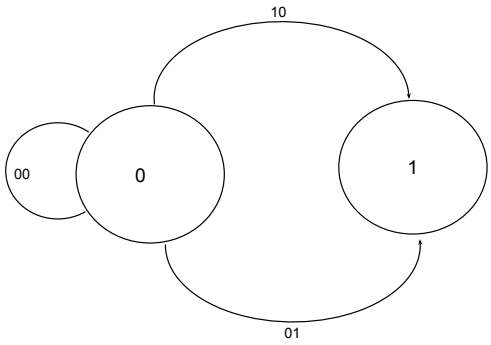


Figura 1.3: Gráfica del shift de la presentación por 2 bloques del shift áureo

A continuación definimos un análogo a la presentación por bloques y los códigos de potencias para las gráficas generadoras de shifts. De esta forma podemos agrupar muchos símbolos en una sola arista, lo que resulta muy conveniente, por ejemplo, cuando estamos trabajando con shifts generados a partir de M-bloques.

Definición 1.27. Sea G una gráfica. Para $N \geq 2$ defiinimos la *N-ésima potencia por aristas de* G como la gráfica que tiene como conjunto de vértices a la colección de todos los caminos de longitud $N-1$ que aparecen en G y a su conjunto de aristas teniendo exactamente una arista desde $e_1e_2...e_{N-1}$ hasta $f_1f_2...f_{N-1}$ siempre que $e_2e_3...e_{N-1} = f_1f_2...f_{N-2}$ (o $t(e_1) = i(f_1)$ cuando $N = 2$) y ninguna en otro caso. La arista es llamada $e_1e_2...e_{N-1}f_{N-1} = e_1f_1f_2...f_{N-1}$. Para $N = 1$ hacemos $G^{[1]} = G$.

Ejemplo 1.21. La figura 1.5 muestra las gráficas de potencias 2 y 3 por aristas para la gráfica G. Basta con seguir las reglas de construcción indicadas en el teorema 1.9 para poder obtener

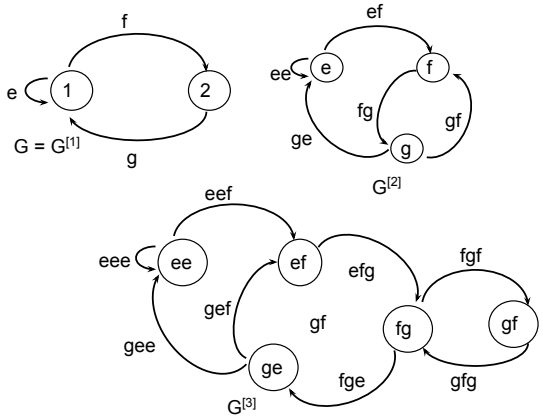


Figura 1.4: Gráficas de orden mayor para una G cualquiera.

la de cualquier orden sin tener que pasar por las otras, aunque es bueno tener en cuenta que el número de vértices puede ser parecido a $|\mathcal{A}|^q$ donde q es el orden de la presentación por bloques.

Hay una estrecha relación entre el código de potencias por bloques de un shift y la N-ésima potencia de su gráfica (siempre y cuando tenga una) que se enuncia a continuación:

Proposición 1.15. *Sea G una gráfica. Entonces* $(\mathsf{X}_G)^{[N]} \cong \mathsf{X}_{G^{[N]}}$.

Demostración. Los símbolos para $(\mathsf{X}_G)^{[N]}$ son los N-bloques de X_G, los cuales son caminos de longitud N en G. Éstos son también los símbolos de $\mathsf{X}_{G^{[N]}}$ y además por definicíon de ambas gráficas, una sucesión bi-infinita de estos símbolos sólo pertenece al shift si se traslapa progresivamente. □

Una observación muy importante es que cuando $N \geq 2$, la matriz de adyacencia de $G^{[N]}$ tiene únicamente ceros y unos. Esto simplifica mucho la lectura de la matriz. Sin embargo, al no preservarse la propiedad de tener sólo símbolos binarios bajo operaciones, no nos conviene profundizar en ello. Ahora definiremos la configuración por bloques de la gráfica, la cual es un análogo a la configuración por bloques de los espacios shift.

Definición 1.28. Sea G una gráfica. Si $N \geq 1$, definimos la N-ésima potencia G^N de G como aquella gráfica que tiene $\mathcal{V}(G^N) = \mathcal{V}(G)$ y con precisamente una arista desde I hasta J por cada camino en G de longitud N desde I hasta J.

Entonces $G^1 = G$ pero conforme aumentamos el tamaño de N la cantidad de aristas por vértice aumenta idénticamente que la cantidad de bloques permitidos que tienen al símbolo de la arista original.

Ejemplo 1.22. Las siguientes gráficas son las potencias de G. Los vértices permanecen intactos mientras se agregan aristas. Al contrario de lo que pasa con el aumento del orden, aumentar la potencia hace crecer la matriz en el valor de sus entradas pero no aumenta el tamaño.

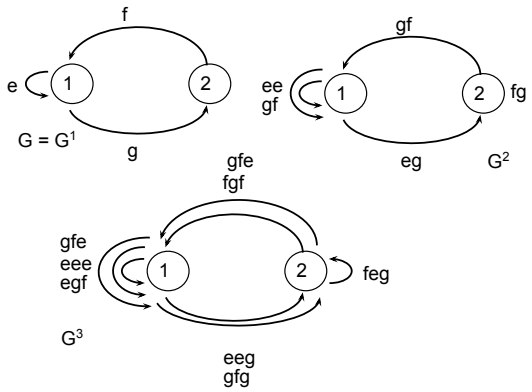


Figura 1.5: Potencias 2 y 3 para la gráfica G.

La última proposición de esta sección nos confirma que las matrices, los códigos de bloques y las potencias de gráficas de los shifts están muy relacionados.

Proposición 1.16. *Sea G la gráfica con matriz de adyacencia A_G. Entonces la matriz de adyacencia de la N-ésima potencia G^N es la matriz A_G a su N-ésima potencia, o en símbolos, $\mathsf{A}_{G^N} = (\mathsf{A}_G)^N$. Además $\mathsf{X}_{G^N} = (\mathsf{X}_G)^N$.*

Demostración. Por definición, G^N y G tienen el mismo conjunto de vértices. Por la proposición 1.14, para los vértices I y J en G hay exactamente $(\mathsf{A}_G^N)_{IJ}$ caminos de longitud N desde I hasta J y este es, justamente, el número de aristas en G^N desde I hasta J. Luego tenemos $\mathsf{A}_{G^N} = (\mathsf{A}_G)^N$. Como X_{G^N} y $(\mathsf{X}_G)^N$ tienen el mismo alfabeto, consistente en los N caminos sobre G, y las mismas sucesiones, formadas por bloques permitidos que aparecen en G, deben ser también el mismo espacio. □

1.9. Shifts sóficos

En esta sección generaremos una nueva clase de espacios shift a partir de gráficas. Como ya vimos en la sección anterior, no todos los espacios shift son de tipo finito y por lo tanto, queda mucho por hacer. Ahora vamos a deshacernos de la condición de que las aristas de nuestra gráfica generadora estén etiquetadas con símbolos diferentes del alfabeto. Nuestros shift sóficos serán precisamente la sucesión de etiquetas del las aristas por las que pasan las caminatas bi-infinitas sobre estas gráficas. La clase de los shift sóficos es mayor que la de los STFs y además tiene a todos los factores de cada STF. En la sección de autómatas derivaremos los lenguajes regulares a partir de bloques de un shift sófico. Son utilizados para transmitir y guardar información de manera efectiva.

A continuación presentamos los objetos matemáticos básicos para trabajar con esta clase de espacios.

Definición 1.29. Una *gráfica etiquetada* $\mathcal{G}$ es un par (G, L), donde G es una gráfica con conjunto de aristas $\mathcal{E}$ y un *etiquetamiento* $\mathsf{L} : \mathcal{E} \to \mathcal{A}$ que asigna a cada arista e de G una etiqueta $\mathsf{L}(e)$ del alfabeto finito $\mathcal{A}$. La *gráfica subyacente* de $\mathcal{G}$ es G. Una gráfica etiquetada es *irreducible* si su gráfica subyacente es irreducible.

Algunas veces hablaremos indistintamente de las gráficas etiquetadas y su gráfica subyacente sin dar lugar a confusión. Nuestro etiquetamiento podría ser inyectivo, pero también podría asignar el mismo símbolo del alfabeto a todas las aristas lo que generaría un shift de un punto fijo. Es bueno notar que todos los shifts por aristas son la clase de shifts sóficos en los que el etiquetamiento es inyectivo. En la mayoría de los casos tratados a continuación a varias aristas les corresponderá el mismo símbolo del alfabeto. Aquellas letras que no pertenezcan a la imágen del etiquetamiento simplemente no aparecerán en el espacio shift por lo que tendemos a usar $\mathcal{A} = \{\mathsf{L}(e) | e \in \mathcal{E}\}$. Comunmente las etiquetas serán las primeras letras del alfabeto mientras que las aristas son letras posteriores a la d en orden alfabético.

Para poder escribir una matriz de adyacencia ya no es conveniente escribir cuántas aristas hay desde el vértice I hasta el J pues nos interesa también conocer las etiquetas de cada uno de ellos. Para ello, ahora escribiremos en cada entrada de la matriz A_{IJ} las etiquetas de cada una de las aristas que unen a I con J separados por un símbolo de suma en caso de ser más de uno, como hacemos con los semigrupos libres finitamente generados en álgebra. Cuando no haya arista alguna uniendo los vértices correspondientes escribiremos el símbolo $\emptyset$. Un ejemplo son las siguientes matrices correspondientes a las gráficas anteriores, $\mathsf{A}_{\mathcal{G}}$ es la matriz de adyacencias de (a) mientras que $\mathsf{A}_{\mathcal{H}}$ es la de (b).

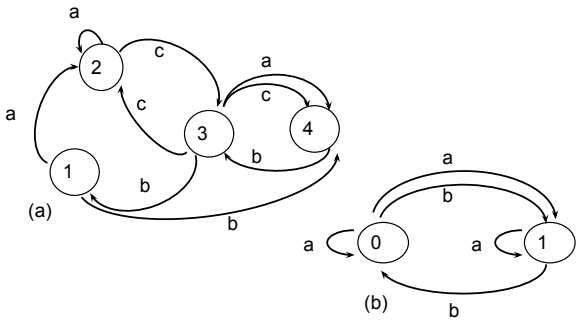


Figura 1.6: Dos gráficas etiquetadas que son diferentes presentaciones de un mismo shift sófico.

$$\mathsf{A}_{\mathcal{G}} = \begin{bmatrix} \emptyset & a & \emptyset & b \\ \emptyset & a & c & \emptyset \\ b & c & \emptyset & a+c \\ \emptyset & \emptyset & b & \emptyset \end{bmatrix} \text{ y } \mathsf{A}_{\mathcal{H}} = \begin{bmatrix} a & a+b \\ b & a \end{bmatrix}$$

Si volvemos a la sección 1.6 veremos que se define un homomorfismo de gráficas. Ahora bien, si queremos hablar de homomorfismo en gráficas etiquetadas pedimos la condición de que la función Φ preserve el etiquetamiento, es decir, que se cumpla que $\mathsf{L}_H(\Phi(e)) = \mathsf{L}_G(e)$ para toda arista $e \in \mathcal{E}(G)$ donde L_M es el etiquetamiento respectivo a la gráfica M. También podemos hablar de el *etiquetamiento de un camino* π $\pi = e_1e_2...e_n$ en G como la concatenación de las etiquetas de las aristas correspondientes $\mathsf{L}(\pi) = \mathsf{L}(e_1)\mathsf{L}(e_2)...\mathsf{L}(e_n)$ el cual es un N-bloque sobre $\mathcal{A}$ y podemos extender este concepto a caminatas bi-infinitas de tal forma que la *etiqueta de la sucesión bi-infinita* $\xi = ...e_{-1}e_0e_1...$ es $\mathsf{L}_\infty(\xi) = \mathsf{L}(e_{-1})\mathsf{L}(e_0)\mathsf{L}(e_1)... \in \mathcal{A}^{\mathbb{Z}}$.

El conjunto de las etiquetas de todas las caminatas bi-infinitas es denotado por $\mathsf{X}_{\mathcal{G}} = \{x \in \mathcal{A}^{\mathbb{Z}} \mid x = \mathsf{L}_\infty(\xi) \text{ para algún } \xi \in \mathsf{X}_G\} = \{\mathsf{L}_\infty(\xi) \text{ para algún } \xi \in \mathsf{X}_G\} = \mathsf{L}_\infty(\mathsf{X}_G)$.

Así pues X_G es siempre un subconjunto del shift completo sobre $\mathcal{A}$. Por ejemplo en las figuras anteriores podemos identificar al shift completo sobre $\{a, b\}$ y sobre $\{a, b, c\}$. Además si $\mathcal{G}$ es isomorfa a $\mathcal{H}$ entonces $\mathsf{X}_{\mathcal{G}} = \mathsf{X}_{\mathcal{H}}$

Definición 1.30. Un subconjunto X de un shift completo es un *shift sófico* si $X = \mathsf{X}_{\mathcal{G}}$ para alguna gráfica etiquetada $\mathcal{G}$. Una *presentación* de un shift sófico X es una gráfica etiquetada $\mathcal{G}$ para la cual $X = \mathsf{X}_{\mathcal{G}}$. La función shift sobre $\mathsf{X}_{\mathcal{G}}$ se escribirá $\sigma_{\mathcal{G}}$.

Es importante aclarar que para un mismo shift sófico podemos tener varias presentaciones. Por ejemplo, si escogemos en cualquier gráfica una arista $e \in \mathcal{E}(G)$ y agregamos una arista e' uniendo los mismos vértices tal que $\mathsf{L}(e) = \mathsf{L}(e')$ entonces tenemos una presentación diferente del mismo shift. Si X es un shift sófico presentado por $\mathcal{G} = (G, \mathsf{L})$ y w es un bloque de $\mathcal{B}(X)$ diremos que un camino π en G es una presentación de w siempre que $\mathsf{L}(\pi) = w$. Un bloque también puede tener diferentes presentaciones como en el ejemplo anterior, donde cualquier bloque que pase por la arista $a = \mathsf{L}(e) = \mathsf{L}(e')$ puede ser presentado de dos formas. Si $x \in \mathsf{X}_{\mathcal{G}}$ decimos que la sucesión bi-infinita ξ en $\mathsf{X}_{\mathcal{G}}$ es una presentación de x si $\mathsf{L}(\xi) = x$. De la misma forma un punto x del shift puede tener varias presentaciones.

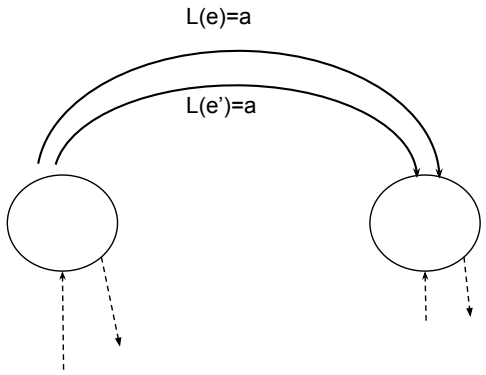


Figura 1.7: Creando una presentación alterna del mismo shift.

Todos los shifts sóficos son espacios shifts y además todo shift de tipo finito es sófico. De allí que hemos construido una categoría mayor a la de los STFs. Se prueba a continuación.

Teorema 1.10. *Todos los shifts sóficos son espacios shift.*

Demostración. Sea X un shift sófico sobre el alfabeto $\mathcal{A}$ y $\mathcal{G} = (G, \mathsf{L})$ una presentación de X. La función etiquetadora $\mathsf{L} : \mathcal{E} \to \mathcal{A}$ asocia a cada arista de G un elemento del alfabeto. Pero ya sabemos que X_G es un shift por aristas que ya probamos que es espacio shift y además $\mathsf{L}_\infty : \mathsf{X}_G \to \mathsf{X}_{\mathcal{G}}$ es un código de 1-bloques pues sólo renombra aristas por lo que no tiene ni memoria ni anticipación. De allí que $\mathsf{X}_{\mathcal{G}}$ es la imagen de un espacio shift bajo un código de bloques y por el teorema 1.1 X es también espacio shift. □

Notemos que todos los shifts por aristas son sóficos porque podemos tomar al conjunto de aristas como un alfabeto y proponer $\mathsf{L} : \mathcal{E} \to \mathcal{E}$ como la función identidad.

Teorema 1.11. *Todo shift de tipo finito (STF) es sófico.*

Demostración. Sea X shift de tipo finito. Por la proposición 1.10 este mismo X es de M-pasos para alguna $M \geq 0$. La prueba del teorema 1.9 construye una gráfica G tal que $X^{[M+1]} = \mathsf{X}_G$ en la cual los vértices son los M-bloques permitidos en X y hay una arista e desde $a_1a_2...a_M$ hasta $b_1b_2...b_M$ si y sólo si estos se traslapan en un bloque de tamaño M, i.e $a_2a_3...a_M = b_1b_2...b_{M-1}$ y $a_1...a_Mb_M(= a_1b_1...b_M)$ pertenece a $\mathcal{B}(X)$. En este caso e es llamado $a_1a_2...a_Mb_M$ y en esta prueba etiquetaremos a e con $\mathsf{L}(e) = a_1$. Esto da lugar a una gráfica $\mathcal{G} = (G, \mathsf{L})$ que demostraremos que es una presentación de X. Sea $\beta_{M+1} : X \to X^{[M+1]} = \mathsf{X}_G$ el código de bloques dado por $\beta_{M+1}(x)_{[i]} = x_{[i,i+M]}$. Como $\mathsf{L}(x_{[i,i+M]}) = x_i$, podemos ver que para cada punto de $x \in X$, $\mathsf{L}_\infty(\beta_{M+1}(x)) = x$ por lo que $X \subseteq \mathsf{X}_\mathcal{G}$. Además cada punto $\xi \in \mathsf{X}_G = X^{[M+1]}$ tiene la forma $\xi = \beta_{M+1}(x)$ para alguna $x \in X$ por lo que $\mathsf{L}_\infty(\xi) = \mathsf{L}_\infty(\beta_{M+1}(x)) = x \in X$ por lo que $\mathsf{X}_G = \mathsf{L}_\infty(\mathsf{X}_G) \subseteq X$ y por lo tanto $X = X_\mathcal{G}$. □

No todos los espacios shift tienen tipo finito. Ya demostramos que el shift par no es de tipo finito en el ejemplo 1.17. Podría decirse que mientras los shifts de tipo finito tienen una memoria de longitud finita fija, los shifts sóficos necesitan una cantidad de memoria finita pero esta puede variar. Por ejemplo, el shift áureo necesita sólo llevar la cuenta de la paridad en las filas de ceros, y estas, aunque pueden ser muy largas, son siempre finitas.

Podemos dar una caracterización de los shifts de tipo finito a partir de los sóficos

Proposición 1.17. *Un shift sófico es un shift de tipo finito si y solo si tiene una presentación* (G, L) *tal que* L_∞ *es una conjugación.*

Demostración. Si X es un shift sófico presentado por (G, L) y L_∞ es una conjugación entonces X es conjugado al shift de tipo finito X_G y por consiguiente por el teorema 1.1 que nos asegura que la imagen de un STF bajo conjugación es un shift de tipo finito ella misma. Además, si X es un shift de tipo finito, es de M-pasos para alguna M entera y podemos utilizar la construcción del teorema anterior para generar una presentación de X. □

El universo de los espacios shift es mayor al de los shift sóficos. En la teoría de los autómatas se llega al mismo resultado con las gramáticas de contexto libre. He aquí un ejemplo:

Ejemplo 1.23. Sea X el shift de contexto libre en donde $\mathcal{A}$=a, b, c y ab^mc^na está permitido como bloque únicamente cuando $m = n$. Probamos que X no es sófico. Si lo fuera, tendríamos $\mathcal{G} = (G, \mathsf{L})$ una presentación. Sea r el número de estados de G. Como $w = ab^{r+1}c^{r+1}a$

está permitido, hay un camino π en G que presenta a w. Sea τ el subcamino de π que presenta a b^{r+1}. Como G tiene únicamente r estados entonces al menos dos estados de τ deben ser el mismo. Luego podemos hacer la subdivisión $\tau = \tau_1\tau_2\tau_3$ donde τ_2 es un ciclo (si el ciclo ocurre ya sea en el primer o último estado de τ entonces τ_1 o $\tau 3$ es un camino vacío). Luego $\tau' = \tau_1\tau_2\tau_2\tau_3$ es un camino en G. Pero si reemplazamos τ por τ' en π obtendríamos un camino π' en G con $\mathsf{L}(\pi') = ab^{r+1+s}cr + 1a$ donde s es la longitud de τ_2. Pero este bloque no está permitido y por lo tanto X no puede ser sófico.

Este ejemplo utiliza el lema del bombeo de la teoría de los autómatas que dice que un camino suficientemente grande siempre tiene un subcamino que puede ser repetido cualquier número de veces. La condición que define el espacio shift de contexto libre está motivada por los lenguajes de contexto libre. Es importante mencionar que las definiciones que presentamos para gráficas se extienden a las gráficas etiquetadas naturalmente. La figura siguiente muestra un diagrama del universo de los espacios shift bajo las clasificaciones que hemos establecido durante este capítulo.

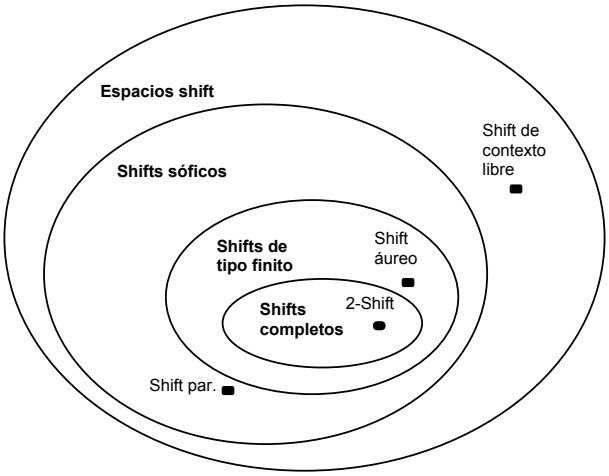


Figura 1.8: El universo de los espacios shift.

Capítulo 2

Conjugaciones

Este trabajo es un esfuerzo por comprender mejor algunos casos especiales de uno de los problemas centrales de la dinámica simbólica. La pregunta generadora es: ¿Dados dos espacios shift, existe una conjugacion entre ellos?. Hasta ahora se sabe que el problema es decidible para shifts derechos completos pero no tenemos respuesta para los shifts completos formados por sucesiones bidireccionales. Podemos acotar la pregunta al universo de los shifts de tipo finito para los cuales ya sabemos que tenemos una gráfica que los genera como su shift por aristas y una matriz de adyacencia. El problema se enunciaría para un caso particular: Dadas las matrices de incidencia

$$A = \begin{bmatrix} 1 & 4 \\ 3 & 2 \end{bmatrix} \text{ y } B = \begin{bmatrix} 2 & 1 \\ 5 & 1 \end{bmatrix} \tag{2.1}$$

¿Serán los shifts X_A y X_B conjugados?.

Primero definiremos dos operaciones sobre gráficas que resultarán muy útiles para reducir el problema, pues toda conjugación se puede descomponer en una serie de funciones inducidas por dichas operaciones. Éstas nos llevarán a formular una relación de equivalencia fuerte para shifts que implicará la equivalencia por conjugación. Sin embargo no se ha descubierto todavía un algoritmo para decidir si dos matrices son fuertemente equivalentes ni

siquiera para matrices de 2×2. No conocemos ningún invariante bajo conjugación fuerte pero podemos debilitar nuestra definición a una conjugación débil que se creyó equivalente hasta 1999 para la cual hay dos invariantes que nos pueden ayudar a identificar, por lo menos, cuándo dos shifts no son conjugados (la conjugación fuerte implica a la débil por lo que dos shifts que no son débilmente conjugados no pueden serlo fuertemente). Existen muchas buenas aproximaciones a este problema pero, al final del trabajo, nosotros resolveremos el problema para el universo de los shifts de árbol donde resulta ser decidible y derivaremos automáticamente la prueba de decidibilidad para los shifts de sucesiones de símbolos unidireccionales.

2.1. Separaciones y amalgamaciones

La separación de estados es un procedimiento para construir nuevas gráficas a partir de una ya existente. Comenzando por una partición de las aristas, cada vértice es separado en cierta cantidad de vértices derivados (recordemos que usamos indistintamente el concepto de <vértice> y <estado>). Aunque la gráfica resultante puede parecer muy distinta de la original, los shifts por aristas de ambas gráficas resultan ser conjugados. Es más, se prueba que toda conjugación puede descomponerse a partir de este tipo de operaciones y esto les da una importacia crucial en el desarrollo de la teoría. Comencemos describiendo la separación de un solo estado. Sea G una gráfica con conjunto de estados $\mathcal{V}$ y conjunto de aritas $\mathcal{E}$. Tomemos un vértice I y asumamos para fines prácticos que no hay aristas que sean loops en I. Recordemos que $\mathcal{E}_I$ es el conjunto de aristas en $\mathcal{E}$ que comienzan en I. Partiremos a nuestro $\mathcal{E}_I$ en dos conjuntos ajenos $\mathcal{E}_I^1$ y $\mathcal{E}_I^2$ donde los supraíndices sólo enumeran los elementos de la partición y no son exponentes. Construimos una nueva gráfica H ayudándonos de esta partición como sigue: Los vértices de H serán los mismos que los de G excepto que el vértice I será reemplazado por dos nuevos vértices llamados I^1 e I^2. En otras palabras $\mathcal{W} = \mathcal{V}(H) = (\mathcal{V}(G) - \{I\} \cup \{I^1, I^2\})$. Para cada $e \in \mathcal{E}_I^i, i = 1, 2$ ponemos una arista en H desde I^i hasta $t(e)$ teniendo el mismo nombre e ($t(e)$ no es I porque acordamos que no hay auto-loops por lo que pertenece a $\mathcal{W}$). Para cada $f \in \mathcal{E}$ comenzando en J y terminando en I colocamos dos aristas llamadas f^1 y f^2 en H, donde f^1 va desde J hasta I^1 y f^2 va desde J hasta I^2. Todos los estados iniciales y finales de las demás aristas en G permanecen en $\mathcal{W}$ y los copiamos directamente en H, completando la construcción. Así pues estamos dividiendo las aristas salientes del vértice y copiando las entrantes. En la figura 2.1 mostramos cómo la separación de estados se realiza en una gráfica pequeña.

Ejemplo 2.1. Sea G la gráfica de la figura 2.2. Definimos la partición $\mathcal{E}_I^1 = \{a\}$ y $\mathcal{E}_I^2 = \{b, c\}$ de $\mathcal{E}_I$ y mostramos la figura resultante en la sección derecha del dibujo.

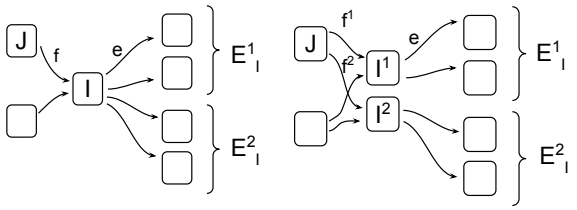


Figura 2.1: Separación en el vértice I con particiones $\mathcal{E}_I^1$ y $\mathcal{E}_I^2$.

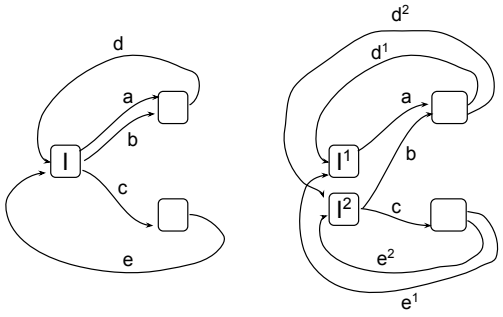


Figura 2.2: Separación en el vértice I con particiones $\mathcal{E}_I^1 = \{a\}$ y $\mathcal{E}_I^2 = \{b, c\}$.

Supongamos que H está formada a partir de G separando el estado I como describimos arriba. Ahora construiremos una conjugación entre los dos shifts por aristas X_G y X_H. Recordemos que $\mathcal{E}^I$ es el conjunto de aristas en G que terminan en I. Definimos un 2-código de bloques $\phi : \mathcal{B}_2(\mathsf{X}_G) \to \mathcal{B}_1(\mathsf{X}_H)$ de forma que

$$\phi(fe) = \begin{cases} f & \text{si } f \notin \mathcal{E}^I \\ f^1 & \text{si } f \in \mathcal{E}^I \text{ y } e \in \mathcal{E}_I^1 \\ f^2 & \text{si } f \in \mathcal{E}^I \text{ y } e \in \mathcal{E}_I^2 \end{cases}$$

Entonces ϕ tiene una anticipación de un símbolo y agrega superíndices dependiendo de

lo que ve enfrente. De esta forma es capaz de separar la gráfica en aristas con la misma letra pero diferenciadas en superíndices mandando G a H y así induce un código de bloques $\Phi = \Phi_\infty^{[0,1]} : \mathsf{X}_G \to \mathsf{X}_H$ de memoria 0 y anticipación 1.

Para la función de regreso, construimos el código de 1-bloques $\psi : \mathcal{B}_1(\mathsf{X}_H) \to \mathcal{B}_1(\mathsf{X}_G)$ como $\psi(f^i) = f$ cuando $f \in \mathcal{E}^I$ y $\psi(e) = e$ cuando $e \notin \mathcal{E}^I$. En otras palabras, ψ solamente borra superíndices. Como las aristas que agregamos están duplicadas y nombradas con la misma letra bajo distintos superíndices, borrar los superíndices sólo nos lleva a quedarnos con aristas duplicadas pero deja a ambas con el mismo nombre lo que hace que en términos prácticos las dos aristas valgan sólo por una y de esta forma nuestra función ψ nos induce un 1-código de bloques $\Psi = \Psi_\infty : \mathsf{X}_H \to \mathsf{X}_G$.

Como agregar y borrar superíndices no tiene efecto resulta que $\psi(\phi(x)) = x$ para toda $x \in \mathcal{B}_1(\mathsf{X}_H)$ De igual manera como los superíndices están determinados de manera biunívoca por el etiquetamiento de las particiones $\mathcal{E}_I^1$ y $\mathcal{E}_I^2$ tenemos que $\phi(\psi(y)) = y$ para toda $y \in \mathcal{B}_1(\mathsf{X})_H)$. Queda entonces demostrado que ϕ es una conjugación de X_G a X_H

Ejemplo 2.2. Las acciones de ϕ y ψ en algún punto $x \in \mathsf{X}_G$ y $y \in \mathsf{X}_H$ de la figura 2.2 se muestra abajo

$$\begin{array}{ccccccc} x & = & \ldots & dcebda & d & cebdadcd & \ldots \\ & & \phi \downarrow & & & & \uparrow \psi \\ y & = & \ldots & d^2ce^2bd^1a & d^2 & ce^2bd^1ad^2c & \ldots \end{array}$$

no podemos seguir computando más allá del símbolo c de y pues necesitaríamos el conocer el símbolo consecutivo a d en x para poder indexar a la misma d. Esto es justo lo que muestra la restricción de la anticipación 1.

El procedimiento de separación de vértices se extiende en dos sentidos: se pueden realizar separaciones con particiones arbitrariamente largas en lugar de tener sólo dos elementos y el particionamiento puede ocurrir en todos los vértices en lugar de uno. Este procedimiento, de paso, abarcará los casos en los que haya aristas en loop:

Definición 2.1. Sea G una gráfica con el conjunto de vértices $\mathcal{V}$ y conjunto de aristas $\mathcal{E}$. Para cada estado $I \in \mathcal{V}$, partimos $\mathcal{E}_I$ en conjuntos ajenos dos a dos $\mathcal{E}_I^1, \mathcal{E}_I^2, ..., \mathcal{E}_I^{m(I)}$, donde $m(I) \geq 1$. Sea $\mathcal{P}$ el resultado de dicha partición de $\mathcal{E}$ y sea $\mathcal{P}_I$ la partición $\mathcal{P}$ restringida a las aristas $\mathcal{E}_I$. La *gráfica de estados separados por aristas salientes* $G^{[\mathcal{P}]}$ *formada desde* G *usando* $\mathcal{P}$ tiene estados $I^1, I^2, ...I^{m(I)}$ donde I varía sobre los vértices en $\mathcal{V}$ y tiene como aristas a e^j donde e es cualquier arista de $\mathcal{E}$ y $1 \leq j \leq m(t(e))$ que se colocan como sigue: si $e \in \mathcal{E}$ va

desde I hasta K, entonces $e \in \mathcal{E}_I^i$ para alguna i y definimos el vértice inicial y el terminal de e^j en $G^{[\mathcal{P}]}$ como $\mathsf{i}(e^j) = I^i$ y $\mathsf{t}(e^j) = K^j$, esto es, e^j va desde I^i hasta K^j. Una *separación elemental de G en el estado I* ocurre cuando $m(I) = 2$ y $m(K) = 1$ para cada $K \neq I$.

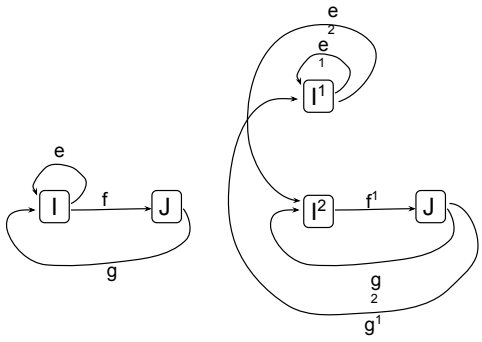


Figura 2.3: Extensión de las separaciones a gráficas con loops.

En la definición anterior estamos asumiendo que ninguno de los elementos de la partición es vacío. Bajo esta premisa, cualquier gráfica formada de separar una gráfica esencial es esencial y la separación de una gráfica irreducible es irreducible. Es importante pensar que un elemento vacío de la partición generaría un vértice al que no podríamos acceder pues no se construye arista alguna hacia él en una separación por aristas salientes o no hay manera de salir de él en una por aristas entrantes.

Proposición 2.1. *Sea H una separación de una gráfica irreducible (esencial) G usando una partición cuyos elementos son no vacíos. Entonces H es irreducible (esencial).*

Demostración. Sea G irreducible y H su separación por aristas salientes. Sea pues (I^j, J^k) un par de vértices cualquiera en H. Por ser irreducible todo par de vértices (I, J) en G tiene un camino $\tau = \tau_1\tau_2...\tau_n$ desde I hasta J. Escogemos los superíndices cambiando cada τ_k respectivo por τ_k^l donde $\tau_k \in \mathcal{E}^l_{\mathsf{i}(k-1)}$. □

Ejemplo 2.3. Para la gráfica G mostrada en la figura 2.3, usamos la partición $\mathcal{E}_I^1 = \{e\}$ $\mathcal{E}_I^2 = \{f\}$ y podemos ver que el loop se convierte en una arista por cada miembro de la partición de I y va desde un I^j en el que está hasta todos los otros I^l incluyendo I^j.

La construcción de $G^{[\mathcal{P}]}$ usa una partición de las aristas salientes de los vértices pero podemos generar una operación análoga por medio de las aristas entrantes.

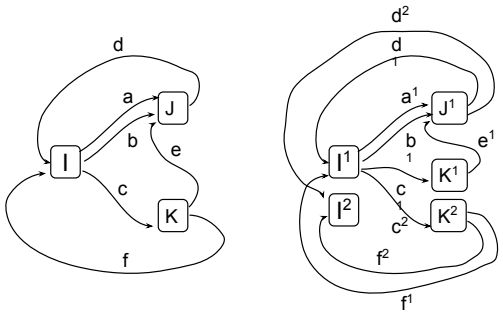


Figura 2.4: Una separación general con más de un vértice separado.

Definición 2.2. Sea G una gráfica con el conjunto de vértices $\mathcal{V}$ y conjunto de aristas $\mathcal{E}$. Para cada estado $J \in \mathcal{V}$, partimos $\mathcal{E}_I$ en conjuntos ajenos dos a dos $\mathcal{E}_J^1, \mathcal{E}_J^2, ..., \mathcal{E}_J^{m(I)}$, donde $m(J) \geq 1$. Sea $\mathcal{P}$ el resultado de dicha partición de $\mathcal{E}$ La *gráfica de estados separados por aristas entrantes* $G_{[\mathcal{P}]}$ *formada desde* G *usando* $\mathcal{P}$ tiene estados $J^1, J^2, ...J^{m(J)}$ donde J varía sobre los vértices en $\mathcal{V}$ y las aristas e^i donde e es cualquier arista de $\mathcal{E}$ y $1 \leq j \leq m(i(e))$. Si $e \in \mathcal{E}$ va desde I hasta J, entonces $e \in \mathcal{E}_j^J$ para alguna j y definimos el vértice inicial y el terminal de e^i en $G_{[\mathcal{P}]}$ como $i(e^i) = I^i$ y $t(e^i) = J^j$, esto es, e^i va desde I^i hasta J^j.

Ejemplo 2.4. En la siguiente figura mostramos una separación, por aristas entrantes, de la gráfica de la izquierda a partir de la partición $\mathcal{E}_1^I = \{e\}, \mathcal{E}_2^I = \{g\}, \mathcal{E}_1^J = \{f, h\}$ lo cual tiene como resultado a $G_{[\mathcal{P}]}$ mostrada del lado derecho de la figura.

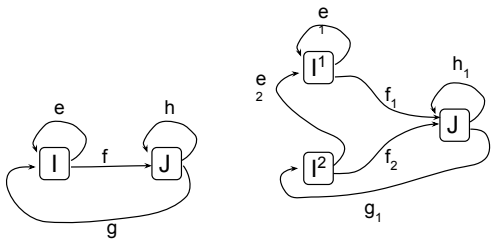


Figura 2.5: Una separación por aristas entrantes.

De ahora en adelante, convenientemente, llamaremos a cualquier gráfica isomorfa a la separación por aristas salientes $G^{[\mathcal{P}]}$ una *separación exterior de* G y de igual manera, *separación interior de* G para la de aristas entrantes. Ahora definimos la operación inversa a la separación.

Definición 2.3. Una gráfica H es una *separación* de una gráfica G y G es una *amalgamación* de H, si H es una gráfica isomorfa a alguna separación exterior $G^{[\mathcal{P}]}$ o a una separación interior $G^{[\mathcal{P}]}$ para alguna partición $\mathcal{P}$.

A continuación se presenta el teorema clave para que abordemos desde el punto de las separaciones el problema de la conjugación entre dos espacios:

Teorema 2.1. *Si una gráfica H es una separación de una gráfica G entonces los shifts por aristas X_G y X_H son conjugados.*

Demostración. Lo probaremos para separaciones interiores aunque los otros casos son parecidos. Asumamos que $H = G_{[\mathcal{P}]}$. Definimos un código de 1-bloques $\psi : \mathcal{B}_1(\mathsf{X}_H) \to \mathcal{B}_1(\mathsf{X}_G)$ por $\psi(e_j) = e$. Podemos observar que si $e_j f_k \in \mathcal{B}_2(\mathsf{X}_H)$, entonces $ef \in \mathcal{B}_2(\mathsf{X}_G)$. Luego la imagen bajo ψ de cualquier camino en H es un camino en G. Luego $\psi = \Psi :_\infty$ y entonces $\Psi(\mathsf{X}_H) \subseteq \mathsf{X}_G$ por lo que $\Psi : \mathsf{X}_H \to \mathsf{X}_G$ es un 1-código de bloques. Ahora definiremos un 2-código de bloques $\phi : \mathcal{B}_2(\mathsf{X}_G) \to \mathcal{B}_1(\mathsf{X}_H)$. Si $ef \in \mathcal{B}_2(\mathsf{X}_G)$, entonces e ocurre en un único $\mathcal{E}_j^J$ y definimos $\phi(fe) = f_j$. Como antes, la imagen bajo ϕ de un camino en G es un camino en H porque estamos añadiendo subíndices derivados de la partición. Sea $\phi = \Phi_\infty^{[1,0]}$ de memoria 1. Luego $\Phi(\mathsf{X}_G) \subseteq \mathsf{X}_H$. Si $x = ...e_{-1}e_0e_1... \in \mathsf{X}_G$, entonces $\Phi(x)$ tiene la forma $\phi(x) = ...e_{(-1)j-1}.e_{(0)j_0}e_{(1)j_1}...$, donde los subíndices entre paréntesis representan los índices originales de la sucesión y los otros la etiqueta del conjunto de la partición al que pertenecen de forma que $\Psi(\Phi(x)) = x$. De igual forma si $y = ...e_{(-1)j-1}.e_{(0)j_0}e_{(1)j_1}..., \in \mathsf{X}_H$ entonces $\Psi(y) = ...e_{-1}e_0e_1....$ Como $e_{(i)j_i}e_{(i+1)j_{i+1}}$ es un 2-bloque en X_H, se sigue que e_{i+1} pertenece a $\mathcal{E}_{j_i}^{t(e_i)}$. Luego $\phi(e_ie_{i+1}) = e_{(i)j_i}$ para toda i y concluímos que $\Phi(\Psi(y)) = y$. En conclusión, X_G y X_H son conjugados. □

Esta prueba muestra que si hay una separación interior para obtener H a partir de G, entonces hay un 1-código de bloques de X_H a X_G llamado el *código de amalgamación interior*. Este código tiene una inversa que es un 2-código de bloques con memoria 1 y anticipación 0 que se llama el *código de separación interior*. Si, en cambio, la separación es exterior, obtenemos un *2-código de separación exterior* que tiene memoria 0 y anticipación 1. Al final de la sección probaremos que toda conjugación de shifts de tipo finito es una composición de operaciones de este tipo.

Ejemplo 2.5. Tomemos la gráfica ilustrada en la figura 2.6 bajo la partición $\mathcal{P}$ con $\mathcal{E}_I^1 = \{a\}, \mathcal{E}_I^2 = \{b, c\}, \mathcal{E}_J^1 = \{d\}, \mathcal{E}_K^1 = \{e\}, \mathcal{E}_K^2 = \{f\}$ para obtener la gráfica separada $G^{[\mathcal{P}]}$ que se muestra del lado derecho.

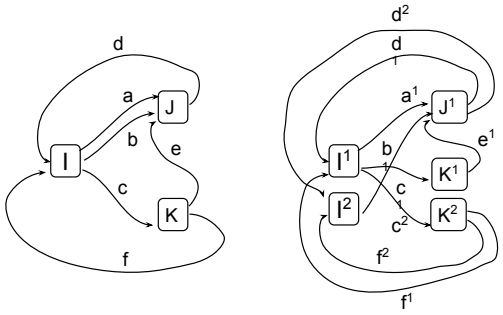


Figura 2.6: Separación externa.

Vamos a dar una definición alternativa de código de separación que nos servirá para hacer una comparación más adelante. En este momento no hay que prestar mucha atención si parece ser más complicada que la que acabamos de dar. Lo importante es que podamos ver que son equivalentes.

Sea G una gráfica y sea X_G el shift que se genera a partir de dicha gráfica con su alfabeto $\mathcal{A}$. Podemos tomar el conjunto $\mathcal{B}_2(\mathsf{X}_G)$ que tiene a todos los bloques de tamaño 2 que aparecen en nuestro shift. Ahora vamos a llamar $\mathcal{B}_2^e(\mathsf{X}_G) = \{x_1x_2 \in \mathcal{B}_2(\mathsf{X}_G) | x_1 = e\}$ el conjunto que tiene a todos los bloques de tamaño 2 que comienzan con el símbolo e. Luego formamos una partición de este conjunto $\mathcal{P}_e$ a cuyos elementos llamaremos $\mathcal{P}_e^i$ donde $1 \leq i \leq m(e)$ siendo $m(e)$ el número de partes de $\mathcal{P}_e$. Juntando todas estas particiones para cada símbolo obtenemos $\mathcal{P} = \cup_{e\in\mathcal{A}}\mathcal{P}_e$ partición de $\mathcal{B}_2(\mathsf{X}_G)$. Vamos a construir un nuevo alfabeto $\mathcal{A}' = \cup_{e\in\mathcal{A}}\{e_1, e_2, ..., e_{m(e)}\}$ y generar el shift completo sobre él: $\mathcal{A}'^{\mathbb{Z}}$. Sea $\phi : \mathcal{B}(\mathsf{X}_G) \to \mathcal{A}'^{\mathbb{Z}}$ el código de 2-bloques tal que $\phi(ab) = a^i$ donde i es aquella tal que $ab \in \mathcal{P}_e^i$. Entonces podemos inducir un código de bloques $\phi = \Phi_\infty^{[0,1]}$ el cual, curiosamente, es justo el código de separación para cierta partición de los vértices. Vamos a verlo: como $\Phi(\mathsf{X}_G)$ es un espacio shift de tipo finito por ser imagen bajo un código de bloques de un STF entonces podemos encontrarle una gráfica H que lo genere. Podemos construir H. Para cada $e^i \in \mathcal{A}'$ construimos un vértice $\mathsf{i}(e)_i$ (donde i nos da el vértice inicial en G de la arista e) y dibujamos una arista desde $\mathsf{i}(a)_j$ hasta $\mathsf{i}(b)_k$ siempre que $a^jb^k \in \Phi\mathsf{X}_G$. Esta gráfica es isomorfa a la que obtenemos haciendo nuestra separación con la siguiente partición: Siempre que $\mathcal{P}_a^l = \{ax_1, ..., ax_t\}$ formamos $\mathcal{E}_{\mathsf{i}(a)}^l = \{x_1, ..., x_t\}$.

Nótese entonces que para definir separaciones y amalgamaciones de gráficas es equivalente usar cualquier definición, ya sea la informal del párrafo anterior o la formal que dimos antes.

Ejemplo 2.6. Consideremos la figura 2.3 y la separación que fue realizada en ella usando $\mathcal{E}_I^1 = \{e\}$ $\mathcal{E}_I^2 = \{f\}$ $\mathcal{E}_J^1 = \{g\}$. Ahora realizamos la partición de $\mathcal{B}_2(\mathsf{X}_G)$ como sigue:

$$\mathcal{B}_2^f = \{fg\} = \mathcal{P}_f$$

$$\mathcal{B}_2 = \{gf, ge\} = \mathcal{P}_g$$

$$\mathcal{B}_2^e = \{ee, ef\} \text{ con } \mathcal{P}_e^1 = \{ee\} \text{ y } \mathcal{P}_e^2 = \{ef\}$$

lo que nos lleva a $fg \mapsto f$, $gf \mapsto g$, $ge \mapsto g$, $ee \mapsto e^1$ y $ef \mapsto e^2$. Puede verificarse fácilmente que obtenemos la misma gráfica con ambos códigos de separación y por lo tanto, el mismo espacio shift.

Ahora veremos cómo cambia la matriz de adyacencias bajo las operaciones de separación y amalgamación. Consideremos el ejemplo al que pertenece la figura 2.6. Sean $\mathcal{V} = \{I, J, K\}$ y $\mathcal{W} = \{I^1, I^2, J^1, K^1, K^2\}$ el conjunto de estados para G y para $H = G^{[\mathcal{P}]}$. Podemos representar como los estados de $\mathcal{W}$ se derivan de los estados de $\mathcal{V}$ por medio de la matriz a continuación:

$$D = \begin{array}{c} \\ I \\ J \\ K \end{array} \begin{array}{c} \begin{array}{ccccc} I^1 & I^2 & J^1 & K^1 & K^2 \end{array} \\ \begin{pmatrix} 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 \end{pmatrix} \end{array}.$$

Aquí D tiene índices sobre $\mathcal{V} \times \mathcal{W}$ y denotaremos las entradas por sus índices en paréntesis en lugar de sus subíndices por lo que, por ejemplo, $D(I, I^1)$ representa la (I, I^1) entrada de D.

El número de aristas en cada elemento de la partición que terminan en un estado de $\mathcal{V}$ determinado estará dado en la siguiente matriz:

$$E = \begin{array}{c} \\ I^1 \\ I^2 \\ J^1 \\ K^1 \\ K^2 \end{array} \begin{array}{c} \begin{array}{ccc} I & J & K \end{array} \\ \begin{pmatrix} 0 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{pmatrix} \end{array}.$$

en donde, por ejemplo, $E(I^1, J)$ es el número de aristas en $\mathcal{E}_I^1$ que terminan en J, en este caso, 1.

Sean $A = \mathsf{A}_G$ y $B = \mathsf{A}_H$ las matrices de adyacencia que ya conocemos de G y de H. Un cálculo directo nos muestra que la entrada (I, J) de la matriz producto DE es la suma sobre los conjuntos de la partición $\mathcal{E}_I^i$ del número de aristas en $\mathcal{E}_I^i$ que terminan en J. Como cada arista en $\mathcal{E}_I$ que termina en J ocurre exactamente en uno de los $\mathcal{E}_I^i$, se sigue que $(DE)(I, J)$ es el número total de aristas en G desde I hasta J, lo cual es por definición $\mathsf{a}(I, J)$. Luego

$$DE = A = \begin{array}{c} \\ I \\ J \\ K \end{array} \begin{array}{c} \begin{array}{ccc} I & J & K \end{array} \\ \begin{pmatrix} 0 & 2 & 1 \\ 1 & 0 & 0 \\ 1 & 1 & 0 \end{pmatrix} \end{array}$$

Si consideramos el producto de D y E en el orden contrario,

$$ED = \begin{array}{c} \\ I^1 \\ I^2 \\ J^1 \\ K^1 \\ K^2 \end{array} \begin{array}{c} \begin{array}{ccccc} I^1 & I^2 & J^1 & K^1 & K^2 \end{array} \\ \begin{pmatrix} 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \end{pmatrix} \end{array}$$

podemos ver que mientras E nos dice cuántas aristas van a dar al vértice K desde todos los vértices originados por la separación, D nos indica qué vértice se originó a partir de K, y por cada uno de los originados tendremos una arista nueva. Así, la matriz producto nos dice la cantidad de aristas que acabaron en cada vértice derivado de K que vienen de vértices derivados. Es decir, $ED = B$, la matriz de adyacencia de la gráfica separada. Probaremos esta relación en general.

Definición 2.4. Sea G una gráfica y $H = G^{[\mathcal{P}]}$ la separación exterior de G utilizando $\mathcal{P}$. Sea $\mathcal{V} = \mathcal{V}(G)$ y $\mathcal{W} = \mathcal{V}(H)$. La *matriz de división D para $\mathcal{P}$* es la matriz de $\mathcal{V} \times \mathcal{W}$ definida por

$$D(I, J^k) = \begin{cases} 1 & \text{si } I = J \\ 0 & \text{otro caso} \end{cases}$$

llamada la *matriz de aristas E para $\mathcal{P}$* es la matriz de $\mathcal{W} \times \mathcal{V}$ definida por $E(I^k, J) = |\mathcal{E}_I^k \cap \mathcal{E}^J|$.

Las matrices de división y de aristas pueden ser usadas para computar las matrices de adyacencia involucradas en una separación externa de estados.

Teorema 2.2. *Sea G una gráfica y $H = G^{[\mathcal{P}]}$ la separación externa formada desde G usando la partición $\mathcal{P}$. Si D y E son las matrices de división y de aristas para $\mathcal{P}$ respectivamente entonces $DE = \mathsf{A}_G$ y $ED = \mathsf{A}_H$.*

Demostración. Usando la notación de la definición tenemos que

$$(DE)(I,J) = \sum_{i=1}^{m(I)} D(I,I^i)E(I^i,J) = \sum_{i=1}^{m(I)} E(I^i,J) =$$

$$\sum_{i=1}^{m(I)} |\mathcal{E}_I^i \cap \mathcal{E}^J| = |(\cup_{i=1}^{m(I)} \mathcal{E}_I^i) \cap \mathcal{E}^J| =$$

$$|\mathcal{E}_I \cap \mathcal{E}^J| = \mathsf{A}|_{G(I,J)}.$$

Luego $DE = \mathsf{A}_G$. Para probar la otra igualdad basta notar que

$$(ED)(I^i,J^j) = E(I^i,J)D(J,J^j) = E(I^i,J) =$$

$$|\mathcal{E}_I^i \cap \mathcal{E}^J| = \mathsf{A}_H(I^i,J^j).$$

□

Supongamos que H es una separación exterior de una gráfica G, i.e, H es isomorfa pero tal vez no exactamente igual al la separación dada por $H' = G^{[\mathcal{P}]}$. Entonces $\mathsf{A}_H = P\mathsf{A}_H P^{-1}$ para alguna matríz de permutación P. Luego de las ecuaciones $\mathsf{A}_G = DE$ y $\mathsf{A}'_H = ED$ correspondientes a la separación vemos que $\mathsf{A}_G = D'E'$ y $\mathsf{A}_H = E'D'$ donde $D' = DP^{-1}$ y $E = PE$, resultando un conjunto de matrices similar al de nuestro teorema 2.2. Curiosamente D' tiene la propiedad de ser una *matriz de división abstracta* lo cual significa que ésta tiene solo ceros y unos, cada fila de D' tiene al menos una entrada con un número 1 y cada columna tiene únicamente una entrada 1. Terminamos esta introducción a las separaciones y amalgamaciones con el teorema central refinado lo máximo posible.

Teorema 2.3. *Sean G y H gráficas esenciales. Entonces H es una separación exterior de G si y sólo si existe una matriz de división D y una matriz rectangular E con entradas enteras no negativas tales que $\mathsf{A}_G = DE$ y $\mathsf{A}_H = ED$.*

Demostración. Para el regreso podemos obtener la partición de los vértices $\mathcal{P}$ de manera que $H = G^{[\mathcal{P}]}$ como sigue: $m(I) =$máx$\{i \,|I^i$ es etiqueta de la columna de $E\}$ y para cada $1 \leq i \leq m(I)$ el elemento $e \in \mathcal{E}_G$ pertenece a $\mathcal{E}_I^i$ si y solo si $E(I^i, t(e))$. A partir de esta $\mathcal{P}$ podemos separar a G en H. El regreso es justamente resultado de la discusión anterior, pues dadas las gráficas ya sabemos que procedimiento seguir para obtener las gráficas D y E. □

2.2. El teorema de descomposición

Conociendo los códigos de amalgamación y separación podemos probar el teorema central del trabajo, que afirma que todas las conjugaciones entre shifts por aristas pueden descomponerse con conjugaciones básicas como factores. Recordemos que para una gráfica G definimos a H como una separación exterior de G siempre que H es isomorfa a una gráfica de separación $G^{[\mathcal{P}]}$, para alguna $\mathcal{P}$ partición de $\mathcal{E}(G)$ las aristas de G. Esto mismo define a G como una amalgamación exterior de H. Ya tratamos el tema de los códigos de amalgamación y separación donde $\psi_{GH} : \mathsf{X}_G \to \mathsf{X}_H$ de memoria 0 y anticipación 1 es el código de separación natural y su inversa $\alpha_{HG} : \mathsf{X}_H \to \mathsf{X}_G$ es un código de 1-bloques de amalgamación. Podemos hablar de *códigos de separación* sin importar si son exteriores o interiores para fines prácticos, al igual que los de amalgamación.

Sean G y H gráficas esenciales. Entonces un código de 1-bloques $\phi : \mathsf{X}_G \to \mathsf{X}_H$, que tiene inversa que también es código de 1-bloques, tiene la forma $\phi = \Phi_\infty$ donde Φ es la extensión de un isomorfismo de gráficas de G a H. Como todo lo que hace este código es renombrar las aristas de G le llamaremos *código de renombramiento*. Notemos que un código de renombramiento cumple con ser simultaneamente separación y amalgamación tanto interior y exterior, pues éstas están definidas bajo clases de isomorfismo. Nuestra meta ahora es mostrar que toda conjugación entre shifts por aristas es descomponible en una sucesión de separaciones y amalgamaciones. Veamos primero un ejemplo.

Ejemplo 2.7. Sea G una gráfica y $\phi = \sigma_G : \mathsf{X}_G \to \mathsf{X}_G$. Estamos entonces considerando la función shift como una conjugación de X en él mismo. Representaremos a ϕ como un código de separación exterior seguido por un renombramiento y seguido de una amalgamación. Sea H la separación completa exterior de G dada por el código $\psi_{GH} = (\Psi_{GH}^{[0,1]})_\infty$ donde $\Psi_{GH}(ef) = e^f$. Esta separación es isomorfa al código de presentación por bloques pues a cada arista le añade su sucesor como superíndice. Sea K la separación completa interior de G usando el código de separación interior $\psi_{GK} = (\Psi_{GK}^{[-1,0]})_\infty$, donde $\Psi(fg) = g_f$. El código de amalgamación interior $\alpha_{KG} = \psi_{GK}^{-1}$ está dado por el código de 1-bloques $gf \mapsto g$. Finalmente es fácil observar que $\Theta(f^g) = g_f$ es un isomorfismo de gráficas de H a K, que induce un código de renombramiento $\theta : \mathsf{X}_H \to \mathsf{X}_K$. El siguiente diagrama muestra lo que le ocurre a un punto en X_G cuando le aplicamos ψ_{GH}, luego θ y después α^{KG}.

$$\begin{array}{ccc} ...cde.fghi... & \xrightarrow{\psi_{GH}} & ...c^d d^e e^f . f^g g^h h^i... \\ {\scriptstyle \phi=\sigma_G}\downarrow & & \downarrow{\scriptstyle \theta} \\ ...def.ghi... & \xleftarrow{\alpha_{KG}} & ...d_c e_d f_e . g_f h_g i_h... \end{array}$$

Este diagrama nos muestra claramente que $\phi = \alpha_{KG} \circ \theta \circ \psi_{GH}$ es la descomposición deseada de nuestra función shift.

Vamos ahora a presentar el teorema central de este capítulo y que, en su presentación para espacios shift, resulta ser la médula de este trabajo.

Teorema 2.4. EL TEOREMA DE DESCOMPOSICION. *Cada conjugación que va de un shift por aristas a otro es la composición de códigos de separación y amalgamación.*

Antes de dar una prueba formal, haremos una breve reseña de los pasos. Sea $\phi : \mathsf{X}_G \to \mathsf{X}_H$ una conjugación. La proposición 1.7 nos muestra que podemos recodificar ϕ de tal forma que sea un código de 1-bloques $\widehat{\phi}$ que tiene como dominio una presentación por bloques mayor de X_G a X_H. Esta recodificación consiste en reescribir ϕ como la composición de $\widehat{\phi}$ con una sucesión de códigos de separación obtenidos de separaciones completas. Si $\widehat{\phi}^{-1}$ fuera también un código de 1-bloques, entonces habríamos acabado pues $\widehat{\phi}$ sería un código de renombramiento. Así pues, necesitamos una forma de reducir la memoria y anticipación de la inversa de un código de 1-bloques usando códigos de separación y amalgamación mientras preservamos la correspondencia del código de 1-bloques. El siguiente lema nos ayuda en eso:

Lema 2.1. *Sea $\phi : \mathsf{X}_G \to \mathsf{X}_H$ una conjugación de 1-bloques cuya inversa tiene memoria $m \geq 0$ y anticipación $n \geq 1$. Entonces hay una separación exterior $\widehat{G}$ de G y $\widehat{H}$ de H inducidas por $\psi_{G\widehat{G}}$ y $\alpha_{\widehat{H}H}$ respectivamente tales que el diagrama siguiente conmuta*

$$\begin{array}{ccc} \mathsf{X}_G & \xrightarrow{\psi_{\widehat{G}}} & \mathsf{X}_{G\widehat{G}} \\ {\scriptstyle\phi}\downarrow & & \downarrow{\scriptstyle\widehat{\phi}} \\ \mathsf{X}_H & \xleftarrow{\alpha_{\widehat{H}H}} & \mathsf{X}_{\widehat{H}} \end{array}$$

donde $\widehat{\phi}$ es una conjugación de 1-bloques cuya inversa tiene memoria m y anticipación $n-1$.

Demostración. Sea $\widehat{H}$ la separación exterior completa de H, tal que las aristas de $\widehat{H}$ tienen la forma h^k, donde $h, k \in \mathcal{E}(H)$ y k es consecutiva a h en H. Sea $\phi = \Phi_\infty$, donde $\Phi : \mathcal{E}(G) \to \mathcal{E}(H)$. Partimos las aristas salientes de G en una nueva partición $\mathcal{P}$ de acuerdo con sus Φ-imágenes, de tal forma que para cada $I \in \mathcal{V}(G)$ y $h \in \mathcal{E}(H)$ definimos $\mathcal{E}_I^h = \{g \in \mathcal{E}_I \mid \Phi(g) = h\}$ un conjunto que clasifica las aristas que salen de I y cuya imágen es la arista $h \in H$. Sea $\widehat{G} = G^{[\mathcal{P}]}$ la gráfica resultante de dicha separación. Finalmente definamos $\widehat{\Phi} : \mathcal{E}(\widehat{G}) \to \mathcal{E}(\widehat{H})$

por $\widehat{\Phi}(g^h) = \Phi(g)^h$. Entonces $\widehat{\Phi}$ induce un código de 1-bloques $\widehat{\phi} = \widehat{\Phi}_\infty : \mathsf{X}_{\widehat{G}} \to \mathsf{X}_{\widehat{H}}$ como se muestra en el diagrama:

$$\begin{array}{ccc} \ldots g_{-3}g_{-2}g_{-1}.g_0g_1g_2g_3\ldots & \xrightarrow{\psi_{G\widehat{G}}} & \ldots g_{-3}^{h_{-2}}g_{-2}^{h_{-1}}g_{-1}^{h_{-0}}.g_0^{h_1}g_1^{h_2}g_2^{h_3}\ldots \\ \downarrow{\scriptstyle \phi=\Phi_\infty} & & \downarrow{\scriptstyle \widehat{\phi}=\widehat{\Phi}_\infty} \\ \ldots h_{-3}h_{-2}h_{-1}.h_0h_1h_2h_3\ldots & \xleftarrow{\alpha_{\widehat{H}H}} & \ldots h_{-3}^{h_{-2}}h_{-2}^{h_{-1}}h_{-1}^{h_{-0}}.h_0^{h_1}h_1^{h_2}h_2^{h_3}\ldots \end{array}$$

Para mostrar que $\widehat{\phi}^{-1}$ tiene la memoria y anticipación requerida hay que comprobar que siempre que $\widehat{y} \in \mathsf{X}_{\widehat{H}}$, las coordenadas $-m$ a $n-1$ de $\widehat{y}$ determinan la coordenada 0 de $\widehat{x} = \widehat{\phi}^{-1}(\widehat{y})$. Para ésto, escribimos $y = \alpha_{\widehat{H}H}(\widehat{y}), x = \phi^{-1}(y)$ y observamos que $\widehat{x}_0 = x_0^{y_1}$. Como $\widehat{y}_{[-m,n-1]}$ determina $y_{[-m,n]}$ y como $y_{[-m,n]}$ a su vez determina x_0, se sigue que $\widehat{y}_{[-m,n-1]}$ determina $\widehat{x}_0$. □

Lo que estamos haciendo en el procedimiento anterior es, en cierta forma, guardar la anticipación en los exponentes de los símbolos para no tener que utilizar tantos símbols. La aplicacion iterada de este lema prueba el teorema de descomposición.

Demostración. PRUEBA DEL TEOREMA DE DESCOMPOSICIÓN

Como ya mencionamos arriba por la proposicion 1.7 podemos siempre recodificar y asumir que cualquier conjugación ϕ es sobre 1-bloques. Sea ϕ^{-1} de memoria m y anticipación n. Al alargar la ventana siempre que sea necesario podemos suponer que $m \geq 0$ y que $n \geq 0$. Si $m = n = 0$, entonces ϕ es un código de 1-bloques con una inversa de 1-bloques lo que lo hace un renombramiento y nos ahorra todo el trabajo. Así supongamos que $n \geq 1$. Por el lema 2.1, existe una sucesión de códigos de separación ψ_j y de amalgamación α_j y conjugaciones de 1-bloques $\widehat{\phi_j}$ donde $1 \leq j \leq n$ para los cuales $\widehat{\phi_j}^{-1}$ tiene memoria m y anticipación $n-j$ como se muestra en el diagrama de abajo:

$$\begin{array}{ccccccccc} \mathsf{X}_G & \xrightarrow{\psi_1} & \mathsf{X}_{G_1} & \xrightarrow{\psi_2} & \mathsf{X}_{G_2} & \longrightarrow & \ldots & \xrightarrow[n]{\psi} & \mathsf{X}_{G_n} \\ \downarrow{\scriptstyle \phi} & & \downarrow{\scriptstyle \widehat{\phi_1}} & & \downarrow{\scriptstyle \widehat{\phi}} & & & & \downarrow{\scriptstyle \widehat{\phi_n}} \\ \mathsf{X}_H & \xleftarrow{\alpha_1} & \mathsf{X}_{H_1} & \xleftarrow{\alpha_2} & \mathsf{X}_{H_2} & \longleftarrow & \ldots & \xleftarrow{\alpha_n} & \mathsf{X}_{H_n} \end{array}$$

La funcion inversa de $\widehat{\phi_n}$ tiene memoria m y anticipación 0. El lema 2.1 puede usarse para reducir la memoria aplicándolo a la gráfica traspuesta. Así pues, hay una sucesión subsecuente de códigos de separación interna ψ_{n+k}, de amalgamación α_{n+k} y conjugaciones de 1-bloques $\widehat{\phi}_{n+k}$ para $1 \leq k \leq m$ tales que $\widehat{\phi}_{n+k}^{-1}$ tiene memoria $m-k$ y anticipación 0 como se muestra a continuación:

$$\begin{array}{ccccccccc}
\mathsf{X}_{G_n} & \xrightarrow{\psi_{n+1}} & \mathsf{X}_{G_{n+1}} & \xrightarrow{\psi_{n+2}} & \mathsf{X}_{G_{n+2}} & \longrightarrow \cdots & \xrightarrow[n+m]{\psi} & \mathsf{X}_{G_{n+m}} \\
\downarrow{\scriptstyle \widehat{\phi_n}} & & \downarrow{\scriptstyle \widehat{\phi_{n+1}}} & & \downarrow{\scriptstyle \widehat{\phi}} & & & \downarrow{\scriptstyle \widehat{\phi_{n+m}}} \\
\mathsf{X}_{H_n} & \xleftarrow{\alpha_{n+1}} & \mathsf{X}_{H_{n+1}} & \xleftarrow{\alpha_{n+2}} & \mathsf{X}_{H_{n+2}} & \longleftarrow \cdots & \xleftarrow{\alpha_{n+m}} & \mathsf{X}_{H_{n+m}}
\end{array}$$

En particular $\widehat{\phi_{n+m}}$ tiene memoria 0 y anticipación 0 por lo que es un renombramiento. Si juntamos ambos diagramas en forma natural obtenemos el siguiente

$$\begin{array}{ccccccccccc}
\mathsf{X}_G & \xrightarrow{\psi_1} & \mathsf{X}_{G_1} & \xrightarrow{\psi_2} & \mathsf{X}_{G_2} & \longrightarrow \cdots & \xrightarrow[n]{\psi} & \mathsf{X}_{G_n} & \xrightarrow{\psi_{n+1}} & \cdots & \xrightarrow[n+m]{\psi} & \mathsf{X}_{G_{n+m}} \\
\downarrow{\scriptstyle \phi} & & \downarrow{\scriptstyle \widehat{\phi_1}} & & \downarrow{\scriptstyle \widehat{\phi}} & & & \downarrow{\scriptstyle \widehat{\phi_n}} & & & & \downarrow{\scriptstyle \widehat{\phi_{n+m}}} \\
\mathsf{X}_H & \xleftarrow{\alpha_1} & \mathsf{X}_{H_1} & \xleftarrow{\alpha_2} & \mathsf{X}_{H_2} & \longleftarrow \cdots & \xleftarrow{\alpha_n} & \mathsf{X}_{H_n} & \xleftarrow{\alpha_{n+1}} & \mathsf{X}_{H_{n+1}} \cdots & \xleftarrow{\alpha_{n+m}} & \mathsf{X}_{H_{n+m}}
\end{array}$$

Recorriendo este diagrama a lo largo de la parte superior, después bajando por la última arista del lado derecho y luego en dirección de las flechas inferiores obtenemos una demostración de cómo ϕ es una composición de conjugaciones del tipo requerido. □

Vamos a ver en un ejemplo cómo funcionan estas descomposiciones:

Ejemplo 2.8. Consideremos la gráfica dibujada en la figura 2.7 (a). Etiquetamos sus aristas $a, b, ..., f$. Los símbolos entre paréntesis (0) y (1) sobre las aristas indican un código de bloques $\phi = \Phi_\infty : \mathsf{X}_G \to \mathsf{X}_{[2]}$ (el contradominio es el shift completo sobre el alfabeto binario), donde $\Phi(a) = 0, \Phi(b) = 0$ y así sucesivamente.

Resulta que ϕ es una conjugación de 1-bloques y que $\phi^{-1} = \theta = \Theta_\infty$ tiene memoria 1 y anticipación 1. El código de 3 bloques Θ está escrito abajo:

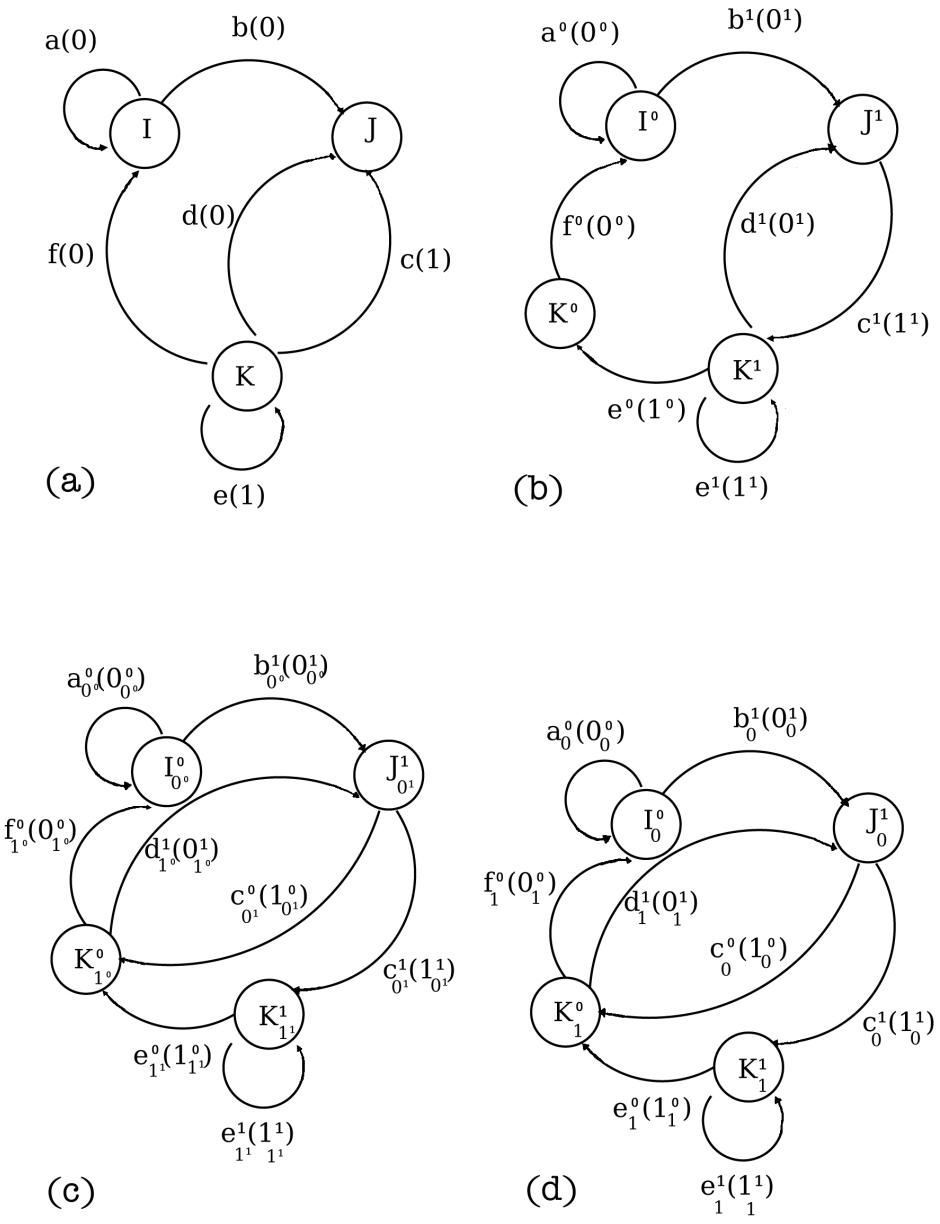


Figura 2.7: Estas son las gráficas de la descomposición del shift generado por (a).

$$\begin{array}{lcccccccc} \Theta: & 000 & 001 & 010 & 011 & 100 & 101 & 110 & 111 \\ & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow \\ & a & b & c & c & f & d & e & e \end{array}$$

Vamos a aplicar el lema 2.1 para obtener una conjugación de 1-bloques con memoria 1 y anticipación 0. Hacemos la separación externa de G de acuerdo con la Φ-imágen de las aristas, resultando en la gráfica G_1 mostrada en la figura 2.7 (b). Para I hay un único estado nuevo I^0 pues todas las aristas que abandonan I tienen Φ-imágen 0. Lo mismo pasa con nuestro estado J que da origen a J^1 pues todas las aristas que salen de él son enviadas al 1 por Φ. En cambio el estado K se divide en K^0 y K^1. El etiquetamiento en G_1 sigue la prueba del lema 2.1 de tal forma que $\widehat{\Phi}(a^0) = \Phi(a)^0 = 0^0$ y así en adelante y H_1 es la gráfica de segundo orden por aristas del shift completo $G^{[2]}$. Sea $\widehat{\phi} = \widehat{\Phi}_\infty$. Luego $\widehat{\phi}^{-1}$ tiene memoria 1 y anticipación 0 pues $\widehat{\phi}^{-1} = \widehat{\Theta}_\infty$ donde $\widehat{\Theta}$ es el código de dos bloques de abajo:

$$\begin{array}{lcccccccc} \widehat{\Theta}: & 0^00^0 & 0^00^1 & 0^11^0 & 0^11^1 & 1^00^0 & 1^00^1 & 1^11^0 & 1^11^1 \\ & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow \\ & a^0 & b^1 & c^0 & c^1 & f^0 & d^1 & e^0 & e^1 \end{array}$$

En esta etapa, la prueba del teorema de descomposición dice que habría que separar internamente G_1 de acuerdo con las $\widehat{\Phi}$-imágenes de todas las aristas entrantes, produciendo una gráfica nueva G_2. De todas maneras, notemos que en la separación que nos lleva de la figura 2.7 (c) a la figura 2.7 (d), para todos los estados de G_1, las $\widehat{\Phi}$-imágenes de todas las aristas entrantes al estado son las mismas. Así pues, la gráfica de separación interior G_2 es isomorfa a G_1, pero sus vértices tienen nuevas etiquetas que reflejan la operación de la separación interna. La complejidad en la notación se puede reducir observando aquellas partes de las etiquetas que son redundantes. Las etiquetas de vértice tienen la forma $L^s_{r_s}$ lo que se puede abreviar a simplemente L^s_r y en el caso de las aristas que tienen la forma $k^t_{r_s}$ donde $\Phi(k) = s$, entonces esta se puede reemplazar por r^s_t. La figura (d) siguiente muestra la simplificación de las etiquetas que resulta mucho más útil en rondas posteriores en donde los subíndices son cada vez mas pequeños. Identificando los símbolos dentro de los paréntesis r^s_t con el 3-bloque trs obtenemos un isomorfismo de gráficas Φ_2 de G_2 en H_2, la gráfica de tercer orden del shift completo. La descomposición se muestra a continuación:

$$\begin{array}{ccccc} \mathsf{X}_G & \xrightarrow{\psi_1} & \mathsf{X}_{G_1} & \xrightarrow{\psi_2} & \mathsf{X}_{G_2} \\ \downarrow{\scriptstyle \phi} & & & & \downarrow{\scriptstyle \widehat{\phi}} \\ \mathsf{X}_H & \xleftarrow{\alpha_1} & \mathsf{X}_{H_1} & \xleftarrow{\alpha_2} & \mathsf{X}_{H_2} \end{array}$$

Aquí $\psi_1^{-1} \circ \psi_2^{-1} \circ \widehat{\phi}_2^{-1}$ es inducida por el código de bloques Θ definido al principio de este ejemplo por lo que Θ puede ser leído directamente de la figura figura 2.7 (d) (leyendo los símbolos entre paréntesis r_t^s como trs).

Es muy probable que el procedimiento que se da en la prueba del teorema de descomposición no sea el más simple para descomponer una conjugación en separaciones y amalgamaciones. Continuamos con un resultado automático del teorema si recordamos que para cada shift por aristas X hay una única gráfica esencial G tal que $X = \mathsf{X}_G$

Corolario 2.1. *Sean G y H gráficas esenciales. Los shifts por aristas X_G y X_H son conjugados si y sólo si G es obtenida de H a partir de una sucesión de amalgamaciones interiores, amalgamaciones exteriores, separaciones interiores y separaciones exteriores.*

Es importante ver que no hay una única descomposición y normalmente existen muchas formas de llegar al mismo lugar. Es posible extender el resultado a todos los espacios shift a partir de un truco que usa las presentaciones por bloques.

2.3. Equivalencia fuerte de shifts

En esta sección traducimos el teorema de descomposición para shifts por aristas a condiciones sobre la matriz de adyacencia, lo que nos conduce a una definición de equivalencia fuerte bajo conjugación para espacios shift. Sea H una separación o amalgamación de una gráfica G y $A = \mathsf{A}_G, B = \mathsf{A}_H$. Recordemos que existen entonces dos matrices de entradas enteras D y E de tal forma que $A = DE$ y $ED = B$. Esto incluye el caso en el que H es isomorfa a G, pues tendríamos una matriz de permutación P tal que $A = P^{-1}BP$ y entonces $D = P^{-1}$ y $E = BP$ de donde obtenemos las ecuaciones $A = DE$ y $ED = B$. Consecuente con el teorema de descomposición encontramos que si A y B son matrices esenciales y $\mathsf{X}_A \cong \mathsf{X}_B$ entonces existe una sucesión $(D_0, E_0), (D_1, E_1), ..., (D_l, E_l)$ de pares de matrices de entradas enteras no negativas tales que

$$\begin{aligned} A = D_0E_0, &\qquad E_0D_0 = A_1 \\ A_1 = D_1E_1, &\qquad E_1D_1 = A_2 \\ A_2 = D_2E_2, &\qquad E_2D_2 = A_3 \\ &\vdots \\ A_l = D_lE_l, &\qquad E_lD_l = B. \end{aligned}$$

Definición 2.5. Sean A y B matrices no negativas enteras. Una *equivalencia elemental* de A a B es un par (R, S) de matrices rectangulares no negativas que satisfacen $A = RS$ y $B = SR$. En este caso escribiremos $(R, S) : A \approx B$ Una *equivalencia fuerte de espacios shift de rezago* l es una sucesión de equivalencias elementales $(R_1, S_1) : A = A_0 \approx A_1, (R_2, S_2) : A_1 \approx A_2, ..., (R_l, S_l) : A_{l-1} \approx A_l = B$. En este caso escribiremos $A \approx B$ (rezago l) y diremos que A es un shift fuertemente equivalente a B ($A \approx B$) si existe una equivalencia fuerte de cualquier rezago desde A hasta B.

Notemos que las matrices A y B en esta definición no necesitan tener el mismo tamaño. El uso de la palabra "fuerte" es para distinguir la relación de otro concepto llamado equivalencia de shifts.

Ejemplo 2.9. Sean

$$A = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 1 & 1 \\ 1 & 0 & 0 \end{bmatrix}, B = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}, C = \begin{bmatrix} 2 \end{bmatrix}$$

En la figura 2.8 dibujamos las gráficas que corresponden a las matrices, la primera de un vértice a A, la segunda de dos vértices a B y la tercera a C.

Podemos ver que si

$$R = \begin{bmatrix} 1 & 1 \\ 0 & 1 \\ 1 & 0 \end{bmatrix}, S = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 1 \end{bmatrix},$$

entonces $(R, S) : A \approx B$ y de igual manera tomando

$$T = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, U = \begin{bmatrix} 1 & 1 \end{bmatrix},$$

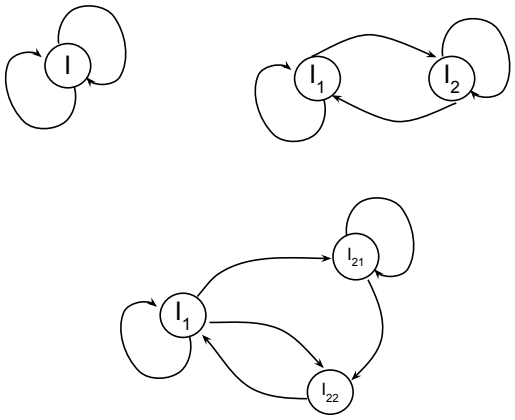


Figura 2.8: Los shifts fuertemente equivalentes tras aplicar la separación.

obtenemos que $(T, U) : B \approx C$. Luego A es fuertemente equivalente a C con rezago 2 o en símbolos $A \approx C$ (rezago 2).

Observemos, sin embargo, que no puede haber una equivalencia elemental entre A y C. Supongamos que tenemos $(V, W) : A \approx C$. Sabemos que

$$rango(VW) \leq \text{ mín}\{rango(V), rango(W)\}$$

pero además forzosamente W debe ser de 1×3 porque al multiplicar por ella del lado derecho debemos obtener una matriz de 3×3 y del lado izquierdo una de 1×1, así que $A = VW$ forzosamente tendría $rango(A) \leq 1$, contradiciendo que nuestra matriz A tiene sus dos primeras filas linealmente independientes y, por lo tanto, rango mayor o igual a 2.

De este ejemplo obtenemos un dato importante. La relación de equivalencia elemental no es una relación de equivalencia, pues no es transitiva. En cambio la equivalencia fuerte de shifts es la relación de equivalencia que resulta de hacer a la equivalencia elemental transitiva.

Proposición 2.2. *La equivalencia fuerte de shifts es una relación de equivalencia en las matrices enteras no negativas.*

Demostración. Primero veamos que $(A, I) : A \approx A$ por lo que la reflexividad se cumple. También si $(R, S) : A \approx B$ entonces justamente $(S, R) : B \approx A$. Luego cualquier cadena

de equivalencias elementales puede operarse en el sentido contrario de donde obtenemos la simetría. También una cadena de equivalencias de A a B puede seguirse de una de B a C pues en ese caso el final de la primera cadena tiene como nodo terminal a B que es justamente el nodo inicial de la segunda, por lo que obtenemos también la transitividad. □

Al inicio de la sección comentamos que si $\mathsf{X}_A \cong \mathsf{X}_B$ entonces $A \approx B$. En lo que sigue de esta sección probaremos el regreso, la equivalencia fuerte en matrices nos lleva a conjugación de los espacios shift que generan.

Para nuestra prueba primero mostraremos cómo una equivalencia elemental $(R, S) : A \approx B$ puede ser usada para producir una conjugación $\gamma_{R,S} : \mathsf{X}_A \to \mathsf{X}_B$. Primero construiremos una gráfica $\mathsf{G}_{R,S}$ como sigue. Comenzamos con la unión disjunta de G_A y G_B. Las aristas de G_A se llamarán las *A-aristas* y las de G_B serán las *B-aristas*. Para cada estado I en G_A y cada estado J en G_B, añadimos R_{IJ} (la entrada IJ de la matriz R) aristas desde I hasta J (llamadas las *R-aristas*) y S_{IJ} aristas (llamadas las *S-aristas*) completando la construcción de $\mathsf{G}_{R,S}$. Llamaremos *RS-camino* a una R-aristas seguida de una S-arista formando un camino en $\mathsf{G}_{R,S}$ y simétricamente definimos un *SR-camino*.

Sean I e I' vértices en G_A. Como $A = RS$ hay una correspondencia biunívoca entre las $A-aristas$ desde I hasta I' y los RS-caminos desde I hasta I'. Denotaremos una biyección específica por $a \leftrightarrow \mathsf{r}(a)\mathsf{s}(a)$ donde $\mathsf{r}(a)$ es una R-arista y $\mathsf{s}(a)$ es una S-arista. De igual forma podemos escribir $a = \mathsf{a}(rs)$ siendo s una S-arista y r una R-arista y así dependiendo del RS-camino escogemos la A-arista a que le corresponde. De igual manera dado que $SR = B$, para cada par de vértices J, J' en G_B tendremos una biyección entre las aristas que van de J a J' y los SR-caminos desde J hasta J'. Denotamos una función biyectiva específica como $b \leftrightarrow \mathsf{s}(b)\mathsf{r}(b)$, o por $b = \mathsf{b}(sr)$.

Ejemplo 2.10. Sea

$$A = \begin{bmatrix} 3 & 2 \\ 2 & 0 \end{bmatrix}, B = \begin{bmatrix} 1 & 3 \\ 2 & 2 \end{bmatrix}, R = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} \ \ y \ \ S = \begin{bmatrix} 1 & 2 \\ 2 & 0 \end{bmatrix}$$

Entonces $(R, S) : A \approx B$. Vale la pena ver que las matrices R y S no son obtenidas con el método que desarrollamos sobre las separaciones de vértices pues estas gráficas tienen la misma cantidad. De allí que ya deben haber pasado por un conjunto de separaciones y amalgamaciones que no utilizan estas matrices R y S. Dibujamos la gráfica $\mathsf{G}_{R,S}$ que podemos ver al final del capítulo en la figura 2.9. En este ejemplo lo único que haremos será dar una biyeccion de las que presentamos en los dos párrafos anteriores. Una posible elección de

biyección entre las A-aristas y los R, S-caminos es la siguiente:

$$\begin{array}{ccccccc} a_1 & a_2 & a_3 & a_4 & a_5 & a_6 & a_7 \\ \updownarrow & \updownarrow & \updownarrow & \updownarrow & \updownarrow & \updownarrow & \updownarrow \\ r_1s_1 & r_2s_4 & r_2s_5 & r_1s_2 & r_1s_3 & r_3s_4 & r_3s_5 \end{array}$$

En nuestra notación de biyección, tenemos $\mathsf{r}(a_1) = r_1, \mathsf{s}(a_1) = s_1, \mathsf{r}(a_2) = r_2, \mathsf{s}(a_2) = s_4$ y así podemos representar todas las relaciones del diagrama, o en dirección contraria $\mathsf{a}(r_1s_1) = a_1, \mathsf{a}(r_2s_4) = a_2$, etc. Podríamos encontrar otras biyecciones diferentes, por ejemplo, todas las que van desde el primer vértice en él mismo pueden permutar valores, lo que en el diagrama de representación se vería como una permutación de los tres primeros símbolos debajo de las flechas dobles. De igual manera, una selección de biyección entre B-aristas y S, R-caminos es:

$$\begin{array}{cccccccc} b_1 & b_2 & b_3 & b_4 & b_5 & b_6 & b_7 & b_8 \\ \updownarrow & \updownarrow & \updownarrow & \updownarrow & \updownarrow & \updownarrow & \updownarrow & \updownarrow \\ s_1r_1 & s_1r_2 & s_2r_3 & s_3r_3 & s_4r_1 & s_5r_1 & s_4r_2 & s_5r_2 \end{array}$$

Bajo esta elección nuestra biyección tiene valores $\mathsf{s}(b_1) = s_1, \mathsf{r}(b_1) = r_1, \mathsf{s}(b_2) = s_1, \mathsf{r}(b_2) = r_2$ y así subsecuentemente mientras que $\mathsf{b}(s_1r_1) = b_1, \mathsf{b}(s_1r_2) = b_2$ etc. Otra vez habría muchas opciones para escoger la biyección.

Ahora podemos definir un código de 2-bloques $\Gamma_{R,S} : \mathcal{B}_2(\mathsf{X}_A) \to \mathcal{B}_1(\mathsf{X}_B)$ usando las biyecciones escogidas haciendo

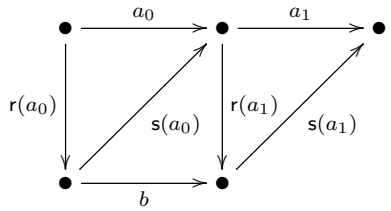
$$\Gamma_{R,S}(a_0a_1) = \mathsf{b}(\mathsf{s}(a_0)\mathsf{r}(a_1)). \tag{2.2}$$

Lo que en realidad estamos pensando es que el bloque a_0a_1 que podría pensarse como un A-camino de longitud 2 en realidad es equivalente a dos RS-caminos concatenados $\mathsf{r}(a_0)\mathsf{s}(a_0)\mathsf{r}(a_1)\mathsf{s}(a_1)$. La parte de enmedio de este camino, $\mathsf{s}(a_0)r_(a_1)$, es un SR-camino que corresponde a una B-arista $b = \mathsf{b}(\mathsf{s}(a_0)\mathsf{r}(a_1))$, el valor de $\Gamma_{R,S}$ en el bloque a_0a_1. La figura en 2.4 nos da una forma de visualizar esta situación. Con una elección fija de ambas biyecciones podemos definir $\gamma_{R,S} = (\Gamma_{R,S})_\infty^{[0,1]}$. De manera parecida podemos definir el 2-código que va

en sentido opuesto como sigue: $\Gamma_{S,R} : \mathcal{B}_2(\mathsf{X}_B) \to \mathcal{B}_1(\mathsf{X}_A)$ dada por las aristas centrales de $s_0r_1s_1r_2$ tal que

$$\Gamma_{S,R}(b_0b_1) = \mathsf{a}(\mathsf{r}(b_1)\mathsf{s}(b_1)). \tag{2.3}$$

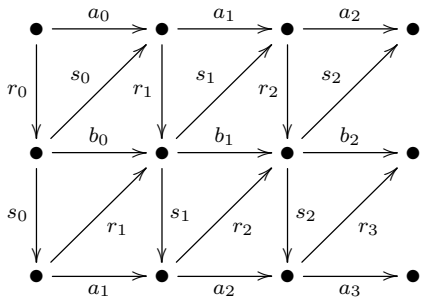
En el diagrama 2.4 podemos corroborar que nuestro razonamiento anterior tiene sentido, pues en este caso las dos aristas del centro de la ecuación son equivalentes al trayecto que realiza la arista de G_A asociada como imagen. Claro que siempre que tengamos matrices únicamente de ceros y unos estas biyecciónes serán únicas por no poder hacer permutaciones de aristas que unen los mismos vértices.



(2.4)

Proposición 2.3. *Si las mismas biyecciónes son usadas para definir* $\gamma_{R,S} : \mathsf{X}_A \to \mathsf{X}_B$ *y* $\gamma_{S,R} : \mathsf{X}_B \to \mathsf{X}_A$ *entonces* $\gamma_{S,R} \circ \gamma_{R,S} = \sigma_A$ *y* $\gamma_{R,S} \circ \gamma_{S,R} = \sigma_B$ *En particular, los códigos de bloques* $\gamma_{R,S}$ *y* $\gamma_{S,R}$ *son conjugaciones.*

Demostración. Las definiciones muestran que dados $a_0a_1a_2 \in \mathcal{B}(\mathsf{X}_A)$ entonces $\Gamma_{R,S}(a_1a_2) = \mathsf{b}(\mathsf{s}(a_1)r(a_2))$ sigue a $\Gamma_{R,S}(a_0a_1) = b(s(a_0)r(a_1))$ pues $r(a_1)$ termina justo en el mismo vértice de $\mathcal{V}(\mathsf{G}_B)$ en el que empezará $s(a_1)$ derivando en $\Gamma_{R,S}(a_0a_1)\Gamma_{R,S}(a_1a_2) \in \mathcal{B}_2(\mathsf{X}_B)$ de donde $\gamma_{R,S}(\mathsf{X}_A) \subseteq \mathsf{X}_B$. De manera análoga ocurre que $\gamma_{S,R}(\mathsf{X}_B) \subseteq \mathsf{X}_A$. El diagrama muestra porqué $\gamma_{S,R} \circ \gamma_{R,S} = \sigma_A$.



También podemos encontrar este argumento en términos de las funciones biyectivas como sigue:

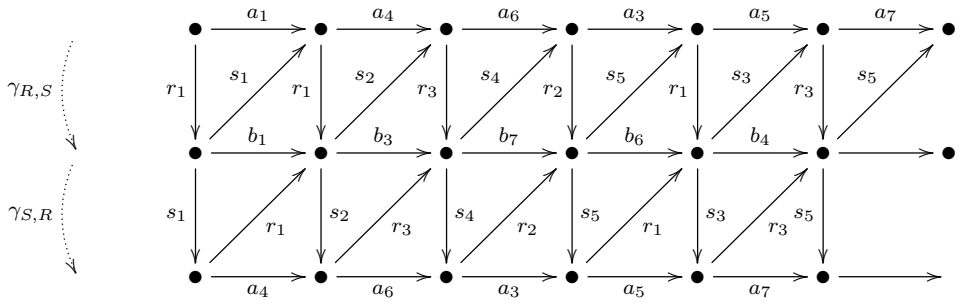
$\Gamma_{S,R} \circ \Gamma_{R,S}(a_0a_1a_2) = \Gamma_{S,R}(\Gamma_{R,S}(a_0a_1)\Gamma_{R,S}(a_1a_2)) = \Gamma_{S,R}(\mathsf{b}(\mathsf{s}(a_0)\mathsf{r}(a_1))\mathsf{b}(\mathsf{s}(a_1)\mathsf{r}(a_2))) = \Gamma_{S,R}(b_0b_1) = \mathsf{a}(\mathsf{r}(b_1)\mathsf{s}(b_1) = \mathsf{a}(r_1s_1) = a_1$

De igual manera podemos ver el resultado para la composición en el orden inverso.

$\Gamma_{R,S} \circ \Gamma_{S,R}(b_0b_1b_2) = \Gamma_{R,S}(\Gamma_{S,R}(b_0b_1)\Gamma_{S,R}(b_1b_2)) = \Gamma_{R,S}(\mathsf{a}(\mathsf{s}(b_1)\mathsf{r}(b_1))\mathsf{b}(\mathsf{s}(b_2)\mathsf{r}(b_2))) = \Gamma_{R,S}(a_1a_2) = \mathsf{b}(\mathsf{s}(a_1)\mathsf{r}(a_2) = \mathsf{b}(s_1r_2) = b_1$

Como σ_A y σ_B son conjugaciones, se sigue que $\gamma_{R,S}$ y $\gamma_{S,R}$ deben ser también conjugaciones pues a partir de la la ecuación $\gamma_{S,R} \circ \gamma_{R,S} = \sigma_A$ obtenemos que $\gamma_{S,R} \circ \gamma_{R,S} \circ \sigma_A^{-1} = \sigma_A \circ \sigma_A^{-1} = 1_A$. □

Ejemplo 2.11. Para las matrices A, B, R y S presentadas en el ejemplo 2.10 y nuestra elección de biyecciones ahí, dibujamos el diagrama de $\gamma_{R,S}$ y de $\gamma_{S,R}$ donde cada flecha representa una arista y cada punto un vértice.



Notemos que una elección diferente en las biyecciones hubiera modificado el diagrama y las conjugaciones.

Ahora pasamos al teorema central demostrado por Williams en el artículo [14], uno de los pilares de la teoría de la clasificación de shifts en la dinámica simbólica.

Teorema 2.5. *TEOREMA DE CLASIFICACION. Los shifts por aristas X_A y X_B son conjugados si y sólo si las matrices A y B son equivalentes respecto a la relación de equivalencia fuerte de shifts.*

Demostración. Nuestra discusión del teorema de descomposición en el comienzo de este capítulo muestra que si A y B son matrices esenciales y $\mathsf{X}_A \cong \mathsf{X}_B$, entonces $A \approx B$. Esto ocurre aún si las matrices no son esenciales. Luego, si $A \approx B$ por medio de una sucesión

(R_j, S_j) de equivalencias elementales, la composición de las conjugaciones γ_{R_j,S_j} que hemos construído arriba es una conjugación desde X_A a X_B □

Hasta este punto hemos terminado nuestra revisión del material de la dinámica simbólica que necesitamos para entrar en los shifts de árbol. En el capítulo 5 utilizaremos un autómata para verificar que es decidible la conjugación entre dos shifts de árbol, lo cual no es posible hacer con los shifts que hemos presentado aquí formados por sucesiones bi-infinitas como explican [6] y [7] y por esta razón no son incluídos en la presentación estándar de la dinámica simbólica. De todas formas, como más tarde sí los necesitaremos, les dedicamos el capítulo siguiente.

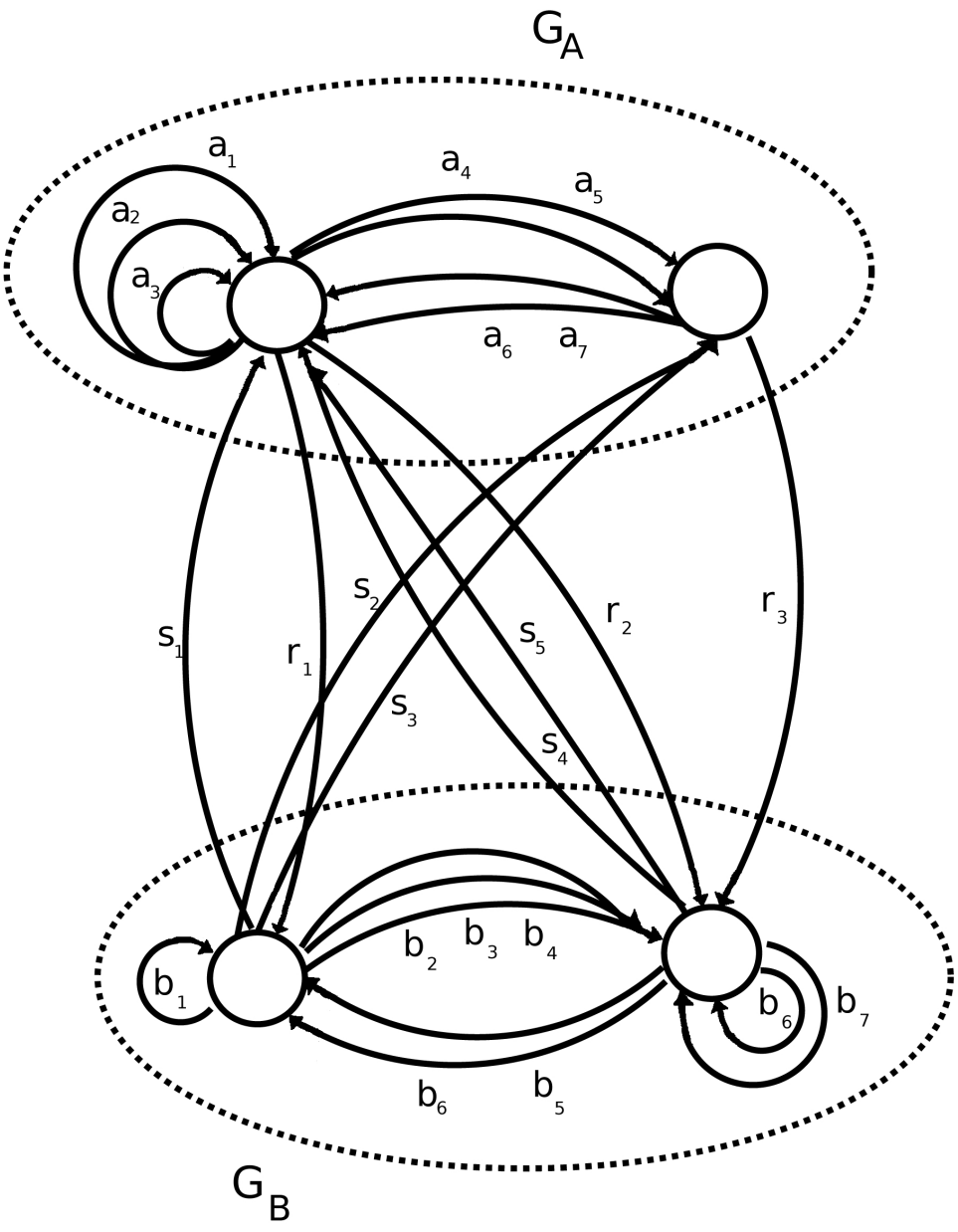


Figura 2.9: Gráfica para construir una conjugacion desde una equivalencia elemental $\mathsf{G}_{R,S}$

Capítulo 3

Autómatas

Los autómatas resultan ser una herramienta muy importante para poder reconocer la pertenencia de ciertos objetos a un conjunto definido por ciertas características. En la teoría de los lenguajes reconocen los lenguajes regulares, que son utilizados para disenãr todos los softwares que utilizamos, como en los procesadores de texto, en los que reconoceremos el proceso de búsqueda de palabras como la acción de evaluar en un autómata cada una de las cadenas de letras separadas por dos espacios. Los shifts sóficos son un caso particular de lenguaje regular. En esta sección no profundizaremos mucho en toda la teoría que se ha forjado para la teoría de la computación sino que presentaremos los dos tipos de autómatas principales y un caso especial de autómata utilizado para hacer cálculos en árboles que nos permitirá probar el teorema de descomposición en el próximo capítulo al reconocer la equivalencia fuerte de shifts entre dos shift de árbol diferentes.

Antes de comenzar con las definiciones daremos un ejemplo de un autómata que todos podemos reconocer fácilmente.

Ejemplo 3.1. En un partido de Tenis se disputan entre dos jugadores de tres a cinco sets de juegos. Para ganar un juego hay que acumular tres partidas ganadas en las cuales siempre saca el mismo jugador y que pueden ganarse de dos formas:

- haciendo salir la pelota de la cancha del jugador oponente antes de que el pueda responder tras el primer bote o;

- haciendo que la pelota bote dos veces en la cancha del oponente.

Cuando se han acumulado tres partidas, se puede ganar únicamente por diferencia de dos partidas por arriba del rival. Hay una infinidad de combinaciones de anotación para que un jugador gane. Para reconocer si alguno ha ganado una sucesión de partidas que lo haga ganar un juego utilizamos el esquema siguiente:

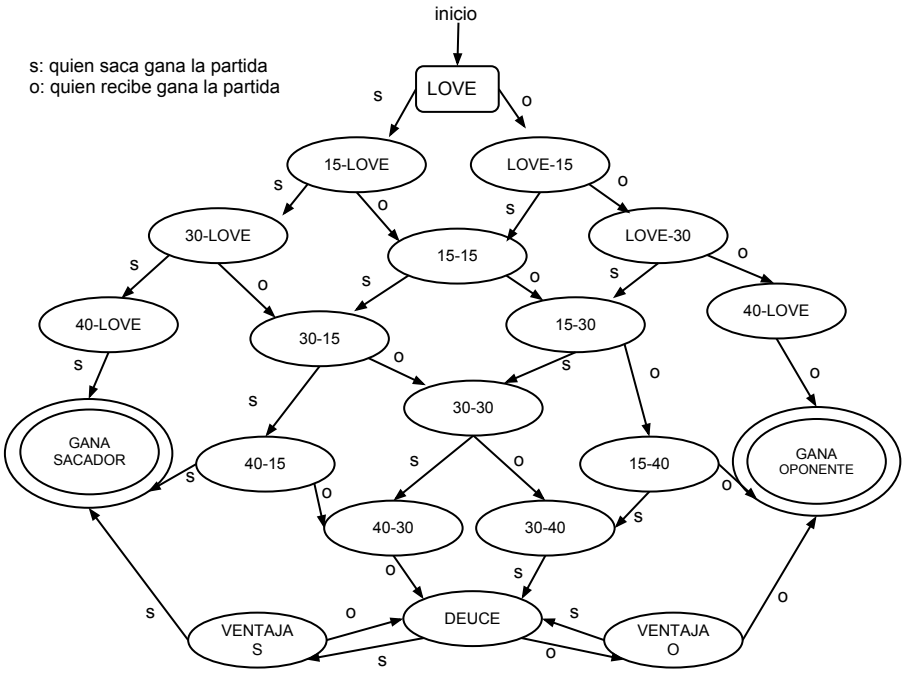


Figura 3.1: Esquema para la evaluación de partidas de tenis.

Si computamos la secuencia de resultados en una sucesión finita donde escribimos s cuando el jugador que está sacando gana la partida y o cuando lo hace su oponente obtenemos una cadena de símbolos que podemos evaluar en el diagrama como sigue: Comenzamos en el vértice LOVE (que significa cero en el argot deportivo) y nos movemos al vértice terminal de la flecha marcada con el símbolo que tiene la sucesión en su primera posición. De igual forma recursivamente si estamos en un vértice después de N símbolos nos moveremos al vértice terminal de la flecha marcada con el símbolo $N+1$ de la sucesión. Así pues, la cadena $w = sooosso$ nos llevaría al vértice marcado con "VENTAJA O". Llamaremos a esta relación unaria evaluación E y la realizaremos sobre todos los posibles bloques del espacio shift $\{s, o\}^{\mathbb{N}}$.

De allí que si queremos saber si una cadena w representa una victoria del jugador que realiza el saque, nos basta con confirmar si $w \in \mathsf{E}_s = \{t \in \mathcal{B}(\{s,o\}^{\mathbb{N}}) | \mathsf{E}(t) = \text{“GANA SACADOR”}\}$.

Así pues, nuestra gráfica a partir de la relación evaluación ha definido un subconjunto de la potencia nuestro espacio shift que nos sirve para saber si debemos marcarle un punto al jugador que saca o no. En el diagrama los vértices con doble borde son llamados terminales porque en el momento de que una sucesión llega a ellos queda evaluada y no nos importa seguir leyéndola, pues la partida se ha acabado.

3.1. DFA: Autómatas deteterministas finitos

El ejemplo presentado en la introducción del capítulo es determinista porque una vez que nos encontramos en un vértice siempre que nos den un resultado de partida sabemos hacia dónde ir y hay una única forma de hacerlo. Además es finito pues aunque puede aceptar cadenas infinitas en su lenguaje, tiene una cantidad finita de posibles estados o vértices.

Comenzaremos a dar las definiciones fundamentales de la teoría de autómatas que no se hayan abordado aún pero también es importante resaltar que en la teoría de la computación hay una terminología diferente para los objetos que ya hemos tratado. Por ejemplo, los bloques sobre un alfabeto son llamados cadenas y ahora los vértices de una gráfica podrán tener funciones especiales.

Definición 3.1. Sea X un espacio shift . Un *Autómata Determinista Finito (DFA) sobre* X denotado por $\mathfrak{A}$ consta de:
1. Un conjunto finito de estados Q.
2. Un alfabeto $\mathcal{A} = \mathcal{B}_1(X)$ de entradas o comandos.
3. Una función de transición δ.
4. Un estado inicial $q_0 \in Q$.
5. Un conjunto de estados finales o de aceptación $F \subseteq Q$.

Como la función de transición $\delta : Q \times \mathcal{B}_1(X) \to Q$ podría no estar definida para algunos puntos, algunas veces agregamos un estado muerto $m \in Q$ de tal manera que todos aquellos (q,a) para los que no hemos definido δ lo tienen como imagen y él bajo cualquier transición llega a si mismo.

Anteriormente evaluamos esta función δ (como función evaluación) sobre bloques de nuestro espacio shift. Para ello necesitamos formalizar algunas cosas. Sea $b \in \mathcal{B}(X)$, entonces

definimos recursivamente la función δ *extendida* como $\delta^* : Q \times \mathcal{B}(X) \to Q$ de tal forma que, dado $q \in Q$, consideramos $\delta^*(q, \epsilon) = q$ en la base de la recursión y dado $b = \epsilon\, b_1 ... b_i ... b_n \in \mathcal{B}_n(X)$ (hemos concatenado al principio un símbolo vacío sólo para ser más específicos), definimos $\delta^*(q, b_i) = \delta(\delta^*(q, b_{i-1}), b_i)$. Así podemos aplicar la función de evaluación a bloques. Adaptamos ahora la definición de lenguaje para autómatas:

Definición 3.2. Sea $\mathfrak{A}$ un DFA sobre X, entonces el *lenguaje de* $\mathfrak{A}$ denotado por $\mathcal{L}(\mathfrak{A})$ es el conjunto $\mathcal{L}(\mathfrak{A}) = \{b \in \mathcal{B}(X) | \delta^*(q_0, b) \in F\}$.

También podemos asociar una gráfica a cada autómata como sigue:

Definición 3.3. Dado $\mathfrak{A}$ un DFA sea $\mathsf{G}_{\mathfrak{A}}$ la grafica que tiene $\mathsf{V}(\mathsf{G}_{\mathfrak{A}}) = Q$ y para la que dibujamos una arista $a \in \mathcal{E}(\mathsf{G}_{\mathfrak{A}})$ desde el estado p hasta el estado q siempre que $\delta(p, a) = q$. A los estados que sean finales los marcaremos con un doble borde y al estado inicial lo marcaremos con una flecha llamada "inicio".

Dada la gráfica de un autómata podemos recuperar toda la información para reconstruir los conjuntos y viceversa, al igual que con los espacios shift. Las funciones de transición también pueden ser expresadas en una tabla como mostramos abajo. Podemos ya dar un ejemplo interesante de cómo puede funcionar un autómata dentro de un programa de procesamiento de texto:

Ejemplo 3.2. El siguiente autómata $\mathfrak{A}$ tiene como lenguaje a todas las palabras que terminan con "endo". Sea $\mathcal{A}$ el alfabeto tradicional del español (UTF-8). Podemos formar el conjunto $\mathcal{L}(\mathfrak{A}) = \{t \in \mathcal{B}(\mathcal{A}^{\mathbb{N}}) | \delta^*(t) =$"VIO endo"$\}$ que tiene a todas los bloques que terminan en "endo".

Describir nuestra función δ en la mayoría de los casos resulta tedioso a partir de notación funcional, pero se puede hacer una buena esquematización en una tabla como mostramos en el ejemplo siguiente:

Ejemplo 3.3. Sea $\mathfrak{A}$ el autómata que acepta en su lenguaje $\mathcal{L}(\mathfrak{A})$ a todos los bloques del shift completo $\{0, 1\}^{\mathbb{N}}$ que no tienen ceros consecutivos que queda descrito por la gráfica:

Podríamos en este caso dar todas las correspondencias entre puntos de $Q \times \mathcal{A}$ de la forma estándar de tal manera que para calcular $\delta^*(A, 1011)$ haríamos

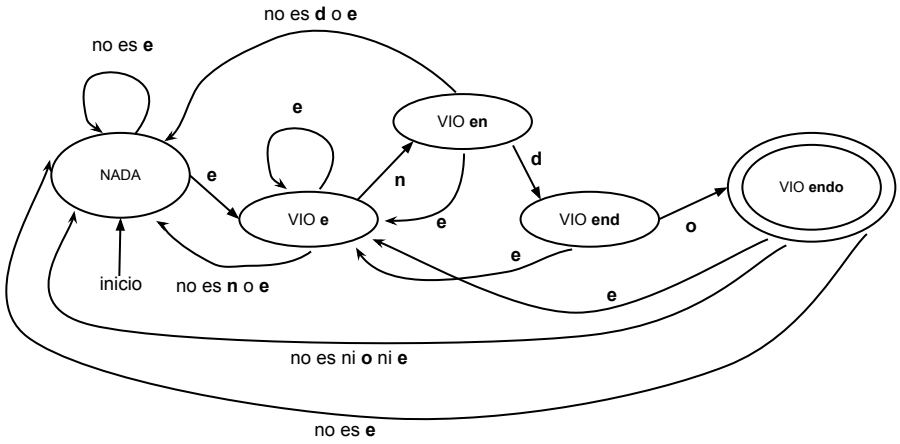


Figura 3.2: Gráfica del autómata $\mathfrak{A}$ para reconocer terminaciones "endo"

$$\begin{gathered}\delta^*(A,\epsilon) = A \\ \delta^*(A,1) = \delta(\delta^*(A,\epsilon),1) = \delta(A,1) = A \\ \delta^*(A,10) = \delta(\delta^*(A,1),0) = \delta(A,0) = B \\ \delta^*(A,101) = \delta(\delta^*(A,10),1) = \delta(B,1) = A \\ \delta^*(A,1011) = \delta(\delta^*(A,101),1) = \delta(A,1) = A\end{gathered}$$

pero también podemos condensar toda la información en una simple tabla como sigue:

Q $\setminus \mathcal{A}$	0	1
$\rightarrow$ *A	B	A
* B	C	A
C	C	C

de donde podemos computar fácilmente la transición del vértice en la columna izquierda bajo la entrada de la columna superior. El vértice que tiene una flecha a su izquierda está señalado como el inicial q_0 y además cada estado final está marcado también por una estrella.

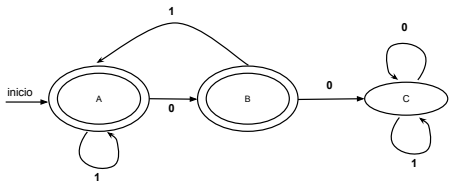


Figura 3.3: Gráfica del autómata $\mathfrak{A}$ que acepta bloques que no tienen dos ceros consecutivos.

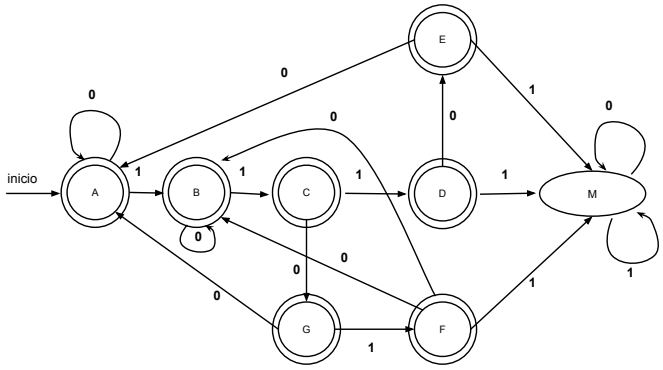


Figura 3.4: Gráfica del autómata $\mathfrak{A}$ que acepta todos los bloques cuyos sub-bloques de tamaño cinco tienen por lo menos dos cifras cero.

Ejemplo 3.4. Otro autómata determinista que no es tan fácil de construir es el que se asegura de que cada bloque del espacio shift tenga en todos sus sub-bloques de tamaño cinco por lo menos dos símbolos cero. Tiene a todos sus estados como finales excepto el muerto, llamado M, y cuando se llega a él sabemos que el bloque no ha cumplido la condición en alguno de sus subbloques y no debe ser aceptado. Lo hemos dibujado en la figura 3.4

Por los autómatas podemos evaluar propiedades numéricas fijándonos en los residuos. El siguiente es un ejemplo fabuloso:

Ejemplo 3.5. Vamos a construir un autómata que tenga en su lenguaje a todos los bloques del shift binario completo que son divisibles por cinco. Para ello hay que recordar que dado un número en su notación decimal n, nos podemos fijar en su expresión binaria w. Sabemos que que concatenando un cero al final de su expresión binaria lo multiplicamos por dos, es decir $w0 = 2n$ y que cuando concatenamos un uno al final de la expresión binaria obtenemos $w1 = 2n + 1$. De allí que si tenemos un número n cualquiera en su expresión binaria w tal que $n = w \equiv k(\mod 5)$, donde k es el residuo al dividir por 5 en notación decimal, entonces $w0 \equiv 2k \pmod 5$ y también $w1 \equiv 2k + 1 \pmod 5$. Usando este razonamiento es fácil diseñar el autómata para el cual, dado $w_0w_1...w_l$ subbloque de w, el estado r_i representa que el residuo al dividir por cinco a $w_0w_1...w_l$ es i y en el que dicho estado r_i tiene una arista etiquetada por 0 que va a $r_{2i \pmod 5}$ y una arista etiquetada por 1 que termina en $r_{2i+1 \pmod 5}$. El autómata funciona de forma recursiva usando la mecánica siguiente: si w tenía residuo k al dividirlo por 5, entonces $w0$ tiene residuo $2k$ al dividirlo por 5 y de igual forma $w1$ tiene residuo $2k+1$ al dividirlo por 5. Sólo aceptaremos a los bloques cuya evaluación termine en el vértice r_0 representante del residuo nulo. Por ejemplo, el número 11001 visitaría los vértices $r_0r_1r_3r_1r_2r_0$ (todos los números visitan r_0 porque es el vértice de inicio del autómata) y por lo tanto, sería aceptado.

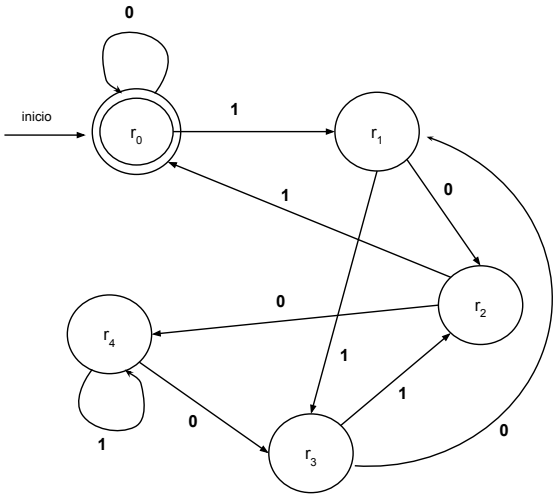


Figura 3.5: Gráfica del autómata $\mathfrak{A}$ que acepta números binarios múltiplos de cinco.

Podemos construir un autómata que realice la misma evaluación de bloques para cada número natural n usando la periodicidad de sus residuos. Podemos observar que el autómata que construyamos tendrá tantos vértices como el número n al que estamos encontrando los múltiplos.

3.2. Autómatas finitos no-deterministas NFA

Hay problemas para los que el planteamiento de encontrarse en un único estado se vuelve un obstáculo. Por ejemplo, en un juego de ajedrez es importante tomar varios caminos y evaluar el mejor. Si construyéramos un autómata que juega ajedrez lo más fácil sería permitirle moverse hacia todos los estados posibles y después evaluar en cuál obtuvo la mejor posición. Para poder lograr esto simplemente podemos cambiar δ para que, en lugar de ser una función que asigna un único estado bajo una entrada, pueda asignarnos más.

Definición 3.4. Sea un espacio shift X. Un *Autómata No-determinista Finito (NFA) sobre* X consta de:
1. Un conjunto finito de estados Q.
2. Un alfabeto $\mathcal{A} = \mathcal{B}_1(X)$ de entradas o comandos.
3. Una función de conjuntos de transición δ.
4. Un estado inicial $q_0 \in Q$.
5. Un conjunto de estados finales o de aceptación $F \subseteq Q$.

Ahora nuestra función de transición cambia su contradominio, de forma que $\delta : Q \times \mathcal{B}_1(X) \to \wp(Q)$ (donde $\wp$ es el conjunto potencia) puede asignar a una misma entrada varios estados, lo cual nos obliga a reformular tanto la extensión de la función de transición δ como la formación del lenguaje.

Definición 3.5. Sea $\mathfrak{A}$ un NFA y sea δ su función de transición. Entonces $\delta^* : Q \times \mathcal{B}(X) \to \wp(Q)$ de tal forma que:

- Base: $\delta^*(q, \epsilon) = \delta(q, \epsilon)$
- Recursión: dado un bloque $b = xb_n$ donde x son los primeros $n-1$ símbolos de b, b_n es el último símbolo y donde además sabemos que $\delta^*(q, x) = \{p_1, p_2, ..., p_j\}$ un conjunto de estados, definimos
$$\delta^*(q, b) = \bigcup_{i=1}^{j} \delta(p_i, b_n)$$
- Además el lenguaje del autómata $\mathfrak{A}$ es:
$$\mathcal{L}(\mathfrak{A}) = \{b \in \mathcal{B}(X) \mid \delta^*(q_0, b) \cap F \neq \emptyset\}.$$

Dicho coloquialmente, lo que hace ahora nuestra función δ^* es formar el conjunto de todos los estados a los que podemos ir desde cada uno de los estados a los que hemos llegado anteriormente. Un punto muy importante es que puede ocurrir que $\delta^*(q, b_i) = \emptyset$ y de allí que si b_i no es el último símbolo de la cadena, hay una indeterminación. Lo que haremos cuando ocurra ésto es marcar ese vértice como atascado, olvidarlo y seguir computando todos los otros, sobre los que sí es posible continuar el camino. Daremos un ejemplo para aclarar cualquier duda sobre ese detalle:

Ejemplo 3.6. El siguiente NFA acepta todos los bloques del shift binario completo que no tengan como penúltimo símbolo al número 1, es decir, los números impares. Podemos notar, a diferencia de los ejemplos con DFAs, que las tablas contendrán conjuntos debajo de las entradas.

Q \ $\mathcal{A}$	0	1
→p	{p,q}	{p}
q	{r,s}	{t}
r	{p,r}	{t}
*s	∅	∅
*t	∅	∅

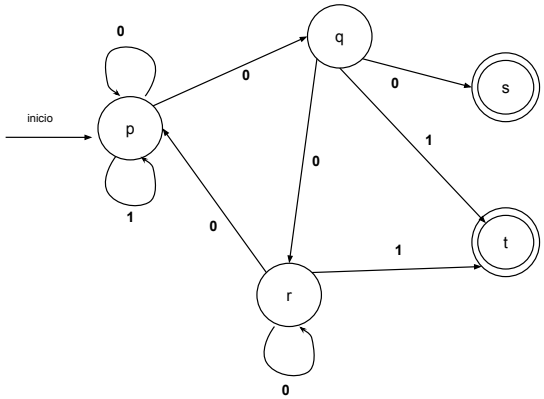


Figura 3.6: Gráfica y tabla del autómata que acepta números binarios cuya penúltima cifra es cero.

Vamos a evaluar entonces el bloque 011010.

$$\delta^*(0) = \{p, q\}$$
$$\delta^*(01) = \cup\{\delta(p, 1)\delta(q, 1)\} = \{p, t\}$$
$$\delta^*(011) = \{p\}$$
$$\delta^*(0110) = \{p, q\}$$
$$\delta^*(01101) = \{p, t\}$$
$$\delta^*(011010) = \{p, q\}$$

Como ninguno de los estados que pertenecen al conjunto $\{p, q\}$ es un estado final concluímos que nuestro bloque no es aceptado.

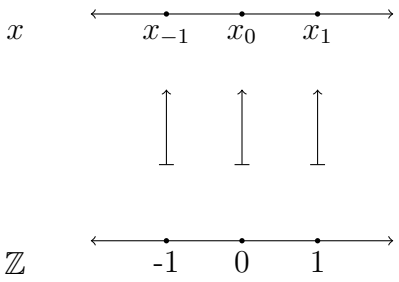
Capítulo 4

Árboles

Ahora presentamos el concepto de shift de árbol de tipo finito que fue introducido en el año 2009 por los investigadores Marie-Pierre Béal y Nathalie Aubrun de la Université Paris-Est, CNRS en el artículo [2], del cual tambien probamos y explicamos digeridamente los resultados más relevantes.

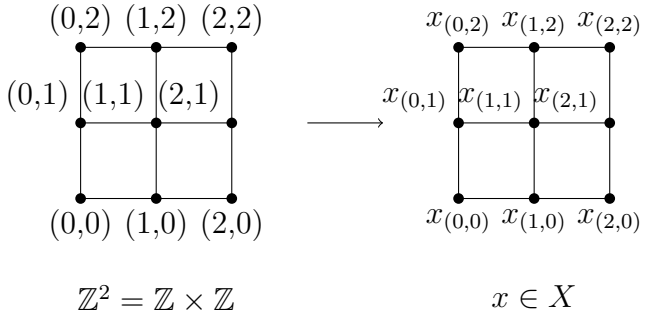
4.1. Árboles y sus shifts

En un ejercicio de abstracción ahora queremos tomar una estructura diferente a las sucesiones para presentar las letras de nuestro alfabeto. Primero analicemos lo que hemos hecho hasta ahora con los espacios de sucesiones: Hemos tomado a los números enteros con su orden y dicho que un punto de un espacio sería una función dada $x : \mathbb{Z} \to \mathcal{A}$ de tal manera que a cada $i \in \mathbb{Z}$ se le asocia una letra x_i y de allí que $x = ...x_{-1}x_0x_1....$ Hemos aprovechado el orden de $\mathbb{Z}$ para comparar las letras utilizadas y para nombrarlas.



Trasladando el orden de los enteros a las sucesiones.

Podríamos ahora tomar la latiz $\mathbb{Z}^2$ y acomodar en cada uno de sus vértices una letra de nuestro alfabeto $\mathcal{A}$, que nombraremos $x_{(i,j)}$ por estar anclada en el vértice (i, j), y definir un punto de nuestro espacio shift como una latiz con letras del alfabeto fijas en cada uno de sus vértices.



Después es posible dar al conjunto $\mathcal{A}^{\mathbb{Z}^2} = \{x \,|\, x : \mathbb{Z}^2 \to \mathcal{A}\}$ de todas las posibles funciones de la latiz en el alfabeto la topología discreta y considerar subconjuntos. Esto se llama un *shift de dimensión 2* o *bidimensional*. La generalización puede continuar a latices multidimensionales $\mathbb{Z}^n$ que tienen una letra en cada vértice y podemos buscar después otros espacios. Estas generalizaciones han empezado a estudiarse en [4] y [17]. Podemos darnos cuenta en una ojeada al título de [15], de que aún hay muchos problemas abiertos en estas exploraciones. Uno de los problemas es que no hay una única forma de definir los bloques en estos espacios shift porque en varias dimensiones se pueden tomar diferentes tipos de celdas. Aún hay mucho por estudiar de los espacios shift de dimensión mayor.

Sin embargo, lo que queremos resaltar es que podemos "armar" espacios shift en estructuras subyacentes discretas i.e: estamos tomando funciones de diferentes objetos discretos

de la matemática en el alfabeto y una vez hecho esto, estudiamos el comportamiento de los conceptos que hemos construído. A lo largo de este capítulo aplicaremos la metodología a los espacios shift de árboles.

En este capítulo queremos proponer una estructura subyacente que podría encontrarse entre los numeros naturales $\mathbb{N}$ y su latiz de dimensión 2, $\mathbb{N}^2$. Recordemos que cuando la estructura subyacente es $\mathbb{N}$ obtenemos como espacio las sucesiones derechas que tienen un primer elemento, de las cuales hablamos en el primer capítulo y a las que naturalmente comprende la teoria de los capítulos anteriores, si consideramos todos los vértices de las gráficas generadoras del shift como un posible inicio de una trayectoria que se computa únicamente hacia el lado derecho. De igual manera $\mathbb{N}^2 = \mathbb{N} \times \mathbb{N}$ consiste en todos los puntos con dos coordenadas enteras positivas del plano cartesiano. Para pasar a una teoría sobre ellas hay varios caminos que no vamos a explorar ahora.

Los N-árboles consisten de un orden jerárquico de todas las posibles cadenas que pueden escribirse con ciertos símbolos (es exactamente un orden para un lenguaje de los que definimos en la sección 1.3). Vamos a recordar un concepto de la construcción de los números naturales que nos va a ser muy útil:

Definición 4.1. Sea $n \in \mathbb{N}$. El *segmento inferior de n* denotado por $\underline{\mathsf{S}_n}$ es: $\underline{\mathsf{S}_n} = \{0, 1, ..., n-1\}$ es decir, el conjunto que tiene a los números menores a él.

Por ejemplo, $\underline{\mathsf{S}_3} = \{0, 1, 2\}$ tiene tres símbolos diferentes. Si queremos tomar todas las posibles cadenas que se pueden formar en $\underline{\mathsf{S}_3}$, (no estamos usando la palabra bloque porque la tenemos reservada para otro concepto) podemos usar la notación del capítulo 1 de tal manera que $\underline{\mathsf{S}_3}^* = \{\epsilon, 0, 1, 2, 00, 01, 02, 10, 11, 12, 20, 21, 22, 000, ...\}$. Podemos ver que es natural ordenarlas como si fueran los números naturales en su notación trescimal. En este momento nos interesa no solo sus el tamaño de las cadenas, sino construir una jerarquía de tal manera que una sea menor que otra si la menor puede extenderse por la derecha para convertirse en la mayor.

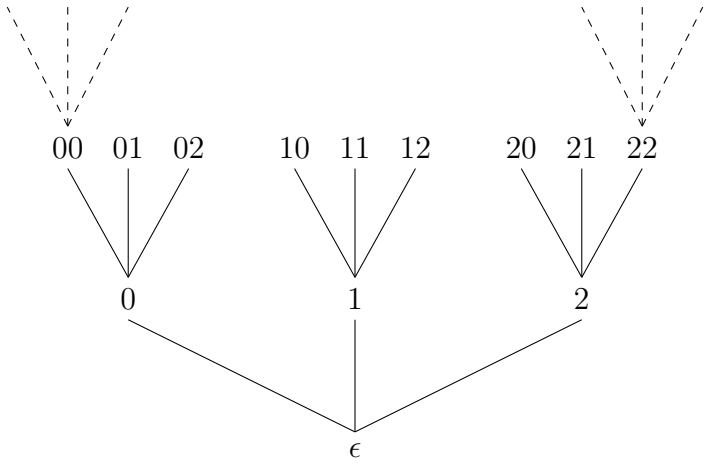
Definición 4.2. Sea $\mathcal{O}$ el orden parcial para $\underline{\mathsf{S}_n}^*$ de tal manera que dado $a, b \in \underline{\mathsf{S}_n}^*$ dos cadenas, diremos que $a \leq_{\mathcal{O}} b$ si y sólo si existe $c \in \underline{\mathsf{S}_n}^*$ tal que $ac = b$ donde si $a = a_1...a_k$ y $b = b_1...b_l$ $a_i, b_j \in \underline{\mathsf{S}_n}$ la operación ab es simplemente concatenar los símbolos $ab = a_1...a_k b_1...b_l$.

Siempre que nos refiramos a un orden dentro de este cuarto capítulo será el de la definición que acabamos de dar, y de igual forma, las cadenas serán siempre las que definimos en el párrafo anterior.

Dado $s \in \underline{\mathsf{S}_n}^*$ podemos formar el conjunto $\mathcal{P}_1(s) = \{p \in \underline{\mathsf{S}_n}^* \mid sa = p \text{ para algún } a \in \underline{\mathsf{S}_n}\}$ el cual es llamado el *conjunto de padres* de s. Cada s tiene justamente n padres y en general

definimos la *ascendencia de grado k de s* como $\mathcal{P}_k = \{p \in \underline{\mathsf{S}_n}^* \mid sq = p \text{ para algún } p \in \underline{\mathsf{S}_n}^* \text{ tal que } |p| = k\}$ el cual tiene n^k elementos. Naturalmente s es *hijo* de p si p es emphpadre de s y además r es *k-abuelo* de s cuando $r \in \mathcal{P}_k(s)$ con $k \geq 2$.

Este orden no nos permite comparar dos cadenas del mismo tamaño a menos que sean la misma porque $a \leq a$ con $a\epsilon = a$ y sin embargo si $a \neq b$ y $|a| = |b|$, cualquier concatenación las hace diferir en tamaño y por lo tanto las vuelve distintas, pero sí organiza las cadenas de diferente tamaño en lo que llamaremos *esqueleto de n-árbol*, como vemos a continuación para $\underline{\mathsf{S}_3}^*$, en donde a cada $s \in \underline{\mathsf{S}_n}^*$ lo unimos por una línea con todos sus padres e indirectamente con todos sus abuelos.



Podríamos pensar a la representación gráfica de $(\underline{\mathsf{S}_n}^*, \mathcal{O})$ como el esqueleto de un n-árbol cuyo tronco ϵ se ramifica n veces en cada nudo [1]. Cada elemento $s \in \underline{\mathsf{S}_n}^*$ será llamado *nudo* del árbol y llamaremos *ramas* a las líneas que unen los nudos o a cualquier camino que podemos recorrer desde un nudo a siguiendo ramas en nuestro árbol y llegando al nudo b.

Definición 4.3. Dado $(\underline{\mathsf{S}_n}^*, \mathcal{O})$ un esqueleto de n-árbol, llamaremos *rama desde a hasta b* a cualquier arreglo $r_{ab} = (a, s_1, s_2, ..., s_k) \in (\underline{\mathsf{S}_n}^*)^k$ de tal forma que $s_1 \leq_{\mathcal{O}} s_2 \leq_{\mathcal{O}} ... \leq_{\mathcal{O}} s_k$ y además $|s_i| - |s_j| = i - j \quad \forall i \geq j$.

Es bueno resaltar que las ramas siempre son una sucesión de nudos de diferente tamaño y en la que dos nudos consecutivos siempre son hijo y padre, en ese orden.

[1] En botática, un nudo en un árbol es aquel lugar donde ha nacido un brote en una rama. Todas las ramificaciones de los árboles son originadas por nudos. Por eso que hemos escogido este nombre, para hacer mas sugestiva nuestra definición.

Proposición 4.1. *Dados $a, b \in \underline{\mathsf{S}_n}^*$ tales que $a \leq_{\mathcal{O}} b$ existe una única rama desde a hasta b*

Demostración. Podemos construir una rama ya que como $a \leq_{\mathcal{O}} b$ entonces $ac = b$ con $c = c_1...c_q, c \in \underline{\mathsf{S}_n}^*$ con $c_i \in \underline{\mathsf{S}_n}$ y $b = b_1...b_d$ donde $b_i \in \underline{\mathsf{S}_n}$ de donde obtenemos que $r_{a,b} = (a, ac_1, ac_1c_2, ..., ac_1...c_q = ac = b)$. Si suponemos que hay una rama distinta entonces existe $m \in \underline{\mathsf{S}_n}^*$ tal que $am = b$ con $m = m_1...m_l$. Primero $q = l$ porque $|ac| = |a|+|c| = |b| = |a| + |m| = |am|$ y restando $|a|$ del segundo y cuarto término obtenemos $|c| = |m|$ entonces las ramas se ven

$$r_{a,b} = (a, ac_1, ..., b = ac = ac_1...c_q)$$
$$r'_{a,b} = (a, am_1, ..., b = am = am_1...m_q),$$

de donde $c_q = b_d = m_q$ y recorriendo los términos de derecha a izquierda $c_i = m_i$. □

En el árbol de $\underline{\mathsf{S}_3}^*$ podemos seguir la rama desde ϵ hasta 1010 que se describiría como $r_{\epsilon,1010} = (\epsilon, 1, 10, 101, 1010)$. Al ser únicas, ahora todas las ramas que comiencen en ϵ las escribiremos únicamente con el nudo en el que terminan como subíndice, en este caso r_{1010}.

Para formar un árbol al parecer lo único que nos falta son las hojas.

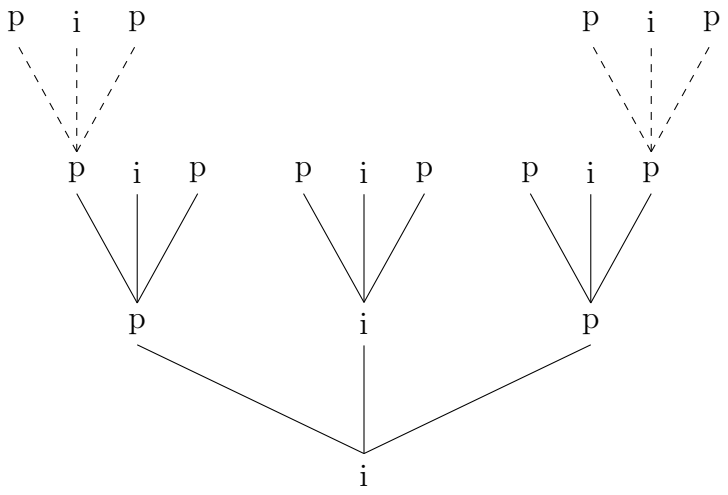
Definición 4.4. Sea $\mathcal{A}$ un alfabeto, es decir, un conjunto finito de símbolos de cualquier tipo. Sea $n \in \mathbb{N}$ fija y por ende $\underline{\mathsf{S}_n}^*$ fijo. Definimos un *n-árbol* t como una función $t : \underline{\mathsf{S}_n}^* \to \mathcal{A}$. Cuando n sea conocida nos referiremos únicamente a un *árbol*.

Teniendo un árbol t podemos generar un orden parcial nuevo $\mathcal{O}'$ para $\mathsf{A} = \{(s, t(s)) \mid s \in \underline{\mathsf{S}_n}^*\} \subseteq (\underline{\mathsf{S}_n}^*) \times t(\underline{\mathsf{S}_n}^*)$, de tal forma que $(a, t(a)) \leq_{\mathcal{O}'} (b, t(b))$ si y sólo si $a \leq_{\mathcal{O}} b$. Para dibujar nuestro árbol pondremos de nodos a nuestros puntos $(s, t(s)) \in (\underline{\mathsf{S}_n}^*) \times t(\underline{\mathsf{S}_n}^*)$ y los uniremos como hicimos antes uniendo con líneas a cada nudo con sus padres bajo el nuevo orden parcial $\mathcal{O}'$ y los etiquetamos únicamente por su segunda coordenada.

Así entonces, a nuestro esqueleto le estamos colocando una hoja (símbolo de nuestro alfabeto) en cada nudo. Si tenemos un nudo s, formalmete llamaremos a $t(s)$ *la hoja de s en el árbol t*. Por ejemplo, si definimos $\mathcal{A} = \{p, i\}$ y un árbol t, que es una función que va de nuestro esqueleto de árbol de $\underline{\mathsf{S}_3}^*$ que tenemos arriba a $\mathcal{A}$, como $t : \{0, 1, 2\}* \to$ {p,i} tal que, considerando a los nodos en su notación ternaria

$$t(s) = \begin{cases} \text{p} & \text{si } s \equiv 0 \bmod 2 \\ \text{i} & \text{si } s \equiv 1 \bmod 2 \text{ o } s = \epsilon \end{cases} \tag{4.1}$$

obtenemos el siguiente árbol:



Una vez fija n, y por ende $\underline{\mathsf{S}_n}^*$, $\mathcal{O}$ y el esqueleto del árbol $(\underline{\mathsf{S}_n}^*, \mathcal{O})$, podemos pensar en shifts. De ahora en adelante supondremos que n está dada.

Definición 4.5. Sea $\mathcal{A}$ un alfabeto fijo. Llamaremos $\mathcal{T}(\mathcal{A})$ al conjunto de todos los n-árboles sobre $\mathcal{A}$, es decir

$$\mathcal{A}^{\underline{\mathsf{S}_n}^*} = \{f \mid f : \underline{\mathsf{S}_n}^* \to \mathcal{A}\} = \mathcal{T}(\mathcal{A}).$$

Podemos definir una métrica y una topología en el conjunto $\mathcal{T}(\mathcal{A})$ como sigue: Dados dos árboles $t, t' \in \mathcal{T}(\mathcal{A})$ consideramos $d : \mathcal{T}(\mathcal{A}) \times \mathcal{T}(\mathcal{A}) \to \mathbb{R}$ tal que:

$$d(t,t') = \begin{cases} 2^{-\min D} & \text{si } D \neq \emptyset \\ 0 & \text{si } D = \emptyset \end{cases}$$

donde $D = \{|s| \mid s \in \underline{\mathsf{S}_n}^* \text{ y } t(s) \neq t'(s)\}$.

Ésto nos dice en qué nivel de ramificación dejan de tener las mismas hojas los dos árboles. Esta métrica induce una topología equivalente a la topología producto si tomamos la topología de $\mathcal{A}$ como discreta.

Proposición 4.2. *Sea $\mathcal{A}$ un alfabeto cualquiera y $\mathcal{T}(\mathcal{A})$ el conjunto de árboles sobre $\mathcal{A}$. La topología producto para $\mathcal{T}(\mathcal{A})$ cuando consideramos a $\mathcal{A}$ con la topología discreta ($\forall a \in \mathcal{A}, a$ es abierto) es equivalente a la topología inducida por la métrica d que acabamos de definir.*

Demostración. Empecemos probando que para cada abierto $A \subseteq \mathcal{T}(\mathcal{A}) = \mathcal{A}^{\underline{\mathsf{S}_n}^*} = \{f \,|\, f : \underline{\mathsf{S}_n}^* \to \mathcal{A}\}$, existe un punto (un árbol) t y una distancia ε de tal forma que $\mathcal{B}_\varepsilon(t) \subseteq A$. Sea $A \subseteq \mathcal{T}(\mathcal{A}) = \mathcal{A}^{\underline{\mathsf{S}_n}^*}$ entonces $A = \prod_{j\in\mathcal{J}} a_j \times \mathcal{A}^{\underline{\mathsf{S}_n}^* - \mathcal{J}}$ donde $\mathcal{J} \subseteq \underline{\mathsf{S}_n}^*$ es finito y $a_j \in \underline{\mathsf{S}_n}^*$ para cada $j \in \mathcal{J}$ porque como $\mathcal{A}$ tiene la topología discreta, todos sus abiertos son uniones de símbolos. Sea l=máx$\{|b| \,|\, b \in \mathcal{J}\}$. Sea entonces $\varepsilon = \frac{1}{2}^{l+1}$ y luego $\mathcal{B}_{\frac{1}{2}^{l+1}}(t)$ donde t es la función tal que $t(j) = a_j$ siempre que $j \in \mathcal{J}$ y tomando $r \in \mathcal{A}$ un símbolo cualquiera definimos $t(j) = r$ cuando $j \in \underline{\mathsf{S}_n}^* - \mathcal{J}$. De allí que

$$\mathcal{B}_\varepsilon(t) \subseteq \mathcal{B}_{\frac{1}{2}^l}(t) = \{t' : \underline{\mathsf{S}_n}^* \to \mathcal{A} |\ t(b) = t'(b)\ \forall b \ni |b| \leq l\} \subseteq$$

$$\{t' : \underline{\mathsf{S}_n}^* \to \mathcal{A} |\ t(b) = t'(b)\ \forall b \in \mathcal{J}\} = \prod_{j\in\mathcal{J}} a_j \times \mathcal{A}^{\underline{\mathsf{S}_n}^* - \mathcal{J}}.$$

Ahora tomemos $t \in \mathcal{T}(\mathcal{A})$ y $\varepsilon > 0$ y sea $\mathcal{B}_\varepsilon(t)$. Queremos encontrar un $A \subseteq \mathcal{T}(\mathcal{A}) = \mathcal{A}^{\underline{\mathsf{S}_n}^*}$ abierto tal que $A \subseteq \mathcal{B}_\varepsilon(t)$. Sea $k \in \mathbb{N}$ tal que $\frac{1}{2}^k < \varepsilon$ y sea $\mathcal{R} = \{b \in \underline{\mathsf{S}_n}^* |\, |b| \leq k\}$. Entonces

$$A = \prod_{b\in\mathcal{R}} (t(b)) \times \mathcal{A}^{\underline{\mathsf{S}_n}^* - \mathcal{R}} \subseteq \{t' \in \mathcal{T}(\mathcal{A}) |\ t(b) = t'(b)\ \forall b \ni |b| < k\} = \mathcal{B}_\varepsilon(t).$$

□

En la teoría de los shift de árbol, en vez de tener una función shift tenemos n, pues podemos movernos en n direcciones para formar un árbol nuevo. De allí que dado $i \in \underline{\mathsf{S}_n}$ tenemos $\sigma_i : \mathcal{T}(\mathcal{A}) \to \mathcal{T}(\mathcal{A})$ tal que $\sigma_i(t)$ es el árbol cuyo tronco (es decir, su inicio) se encuentra en el padre i de ϵ y generalizando $\sigma_i(s)$ asocia a s su padre que termina en i:

$$\begin{array}{ccc} t : \underline{\mathsf{S}_n}^* \to \mathcal{A} & \overset{\sigma}{\longmapsto} & \sigma_i(t) : \underline{\mathsf{S}_n}^* \to \mathcal{A} \\ t(is) \mapsto a & & \sigma_i(t(s)) \mapsto a \end{array}$$

Definición 4.6. Al conjunto $\mathcal{T}(\mathcal{A})$ con sus n shifts le llamaremos el *n-shift completo de árboles sobre $\mathcal{A}$*.

Para generalizar a nuestras funciones shift diremos que, dada una cadena $b = b_1...b_l \in \underline{\mathsf{S}_n}^*$, la funcion σ_b es aquella que coloca el tronco del árbol en el nudo b, es decir $\sigma_b = \sigma_{b_l} \circ ... \circ \sigma_{b_1}$ y de allí que la imágen de $\sigma_b(t(x))$ es justamente $t(bx)$.

Ahora que ya tenemos nuestro espacio shift vamos a definir los bloques de un punto. Para ello necesitamos un subconjunto de $\underline{\mathsf{S}_n}^*$ cerrado bajo prefijos, lo que significa que éste

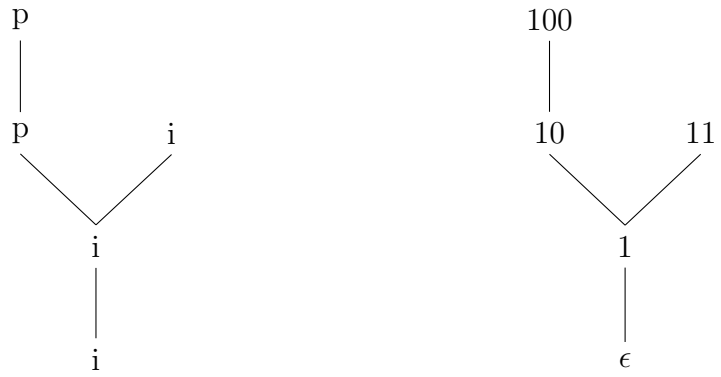
tiene que tener a todos los hijos de sus elementos. Recordemos que una rama es un arreglo $\bar{r}$. Diremos que $x \in \bar{r}$ si x es una entrada de $\bar{r}$.

Definición 4.7. Sea $(\underline{\mathsf{S}_n}^*, \mathcal{O})$ un esqueleto de árbol. Sea $S \subseteq \underline{\mathsf{S}_n}^*$ un conjunto cualquiera. S es emphcerrado bajo prefijos cuando, para cada $s \in S$, el siguiente enunciado es verdadero: $x \in r_s \Rightarrow x \in S$. Además llamamos $\bar{S}$ al conjunto $\bar{S} = \bigcup_{s \in S} \{x \in r_s\}$.

Ahora nos interesa la imagen de los árboles en los subconjuntos cerrados bajo prefijos.

Definición 4.8. Un *patrón p_S con soporte S* es una función $p : S \to \mathcal{A}$, es decir, $p = t|_S$ para algún $t \in \mathcal{T}(\mathcal{A})$, donde S es cerrado bajo prefijos. El patrón se presentará en un arreglo que tiene las imágenes de los elementos de S bajo p en el mismo orden que sus preimágenes ordenadas numéricamente, o lo que es lo mismo, el orden que obtenemos leyendo de izquierda a derecha la gráfica del árbol de S con el orden $\mathcal{O}$.

Por ejemplo, en nuestro esqueleto de árbol para $\underline{\mathsf{S}_3}^*$ un conjunto cerrado bajo prefijos puede ser $S = \{\epsilon, 1, 10, 11, 100\}$ y si usamos el árbol definido en (4.2) para generar $p_S = t|_S$ obtendríamos $p_S =$ (i, i, p, i, p). La presentación gráfica de p_S se obtiene aplicando nuestro orden $\mathcal{O}'$ a los puntos $(s, t(s))$, lo que nos daría la gráfica de abajo. Es importante recalcar la diferencia entre un esqueleto de árbol, que es el que se encuentra a la derecha, y un patrón, que es el que se encuentra a la izquierda.



Todavía necesitamos un último elemento importante para cumplir nuestra meta:

Definición 4.9. El *conjunto de cadenas sobre $\underline{\mathsf{S}_n}$ de longitud a lo más l* es el conjunto $\underline{\mathsf{S}_n}^l = \{b \in \underline{\mathsf{S}_n}^* \mid |b| \leq l\}$.

Así cuando escribimos $l = *$ estamos diciendo que l es infinito. Ahora sí estamos bien armados para ensamblar la definición de bloque:

Definición 4.10. Sea $t \in \mathcal{T}(\mathcal{A})$ un árbol. Diremos que f *es un bloque de altura* h *en* t *enraizado en* a, denotado por $f(h,a)$, cuando dado un patrón $p_S = t|_S : S \to \mathcal{A}$ donde $S = \underline{\mathsf{S}_n}^{h-1}$, se cumple que $f = \sigma_a \circ t|_S$ con $f = \sigma_a \circ t|_S : S' \to \mathcal{A}$ y $S' = \{as \,|\, s \in S\}$. Al símbolo a lo llamamos la *raíz del bloque*.

Diremos que un patrón p_S aparece en t si $\sigma_a \circ t|_S$ tiene la misma gráfica que p_S para alguna $a \in \underline{\mathsf{S}_n}^*$, donde esta gráfica es la que se deriva de ordenar las imágenes de S con el orden $\mathcal{O}'$. De igual manera podemos hablar de un bloque que aparezce en t. La representación de un bloque se deriva de las presentaciones que definimos para los partones (ya sea gráfica o en arreglo de símbolos). Consideremos $\mathcal{T}(\mathcal{A})$ el n-shift completo de árboles sobre $\mathcal{A}$. Cuando un bloque f aparezca en algún $t \in \mathcal{T}(\mathcal{A})$ diremos que f *es un bloque permitido en* $\mathcal{T}(\mathcal{A})$. El conjunto de todos los bloques permitidos en $\mathcal{T}(\mathcal{A})$ es $\mathcal{B}(\mathcal{T}(\mathcal{A}))$. Naturalmente definimos $\mathcal{B}_h(t) = \{f(h,a) \,|\, a \in \underline{\mathsf{S}_n}^*\}$ y $\mathcal{B}(t) = \cup_{h\in\mathbb{N}}\{f(h,a) | a \in \underline{\mathsf{S}_n}^*\}$.

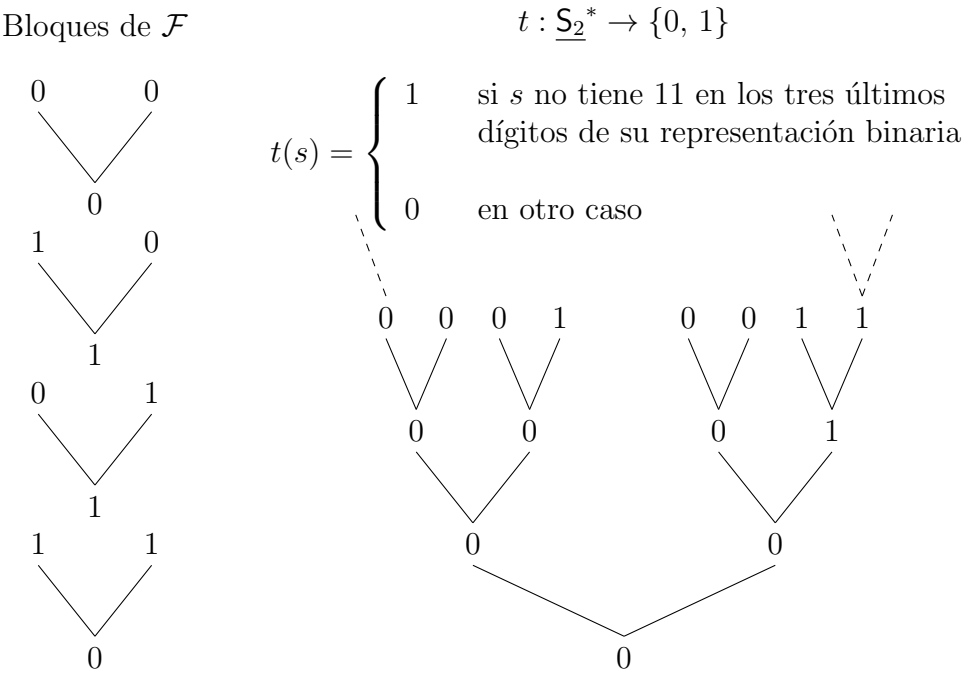
Si $f(h,a)$ es un bloque de $t \in \mathcal{T}(\mathcal{A})$ entonces $\sigma_i(f(h,a)) = f(h,ia)$ con $a, i \in \underline{\mathsf{S}_n}^*$.

Definición 4.11. Sea $\mathcal{A}$ un alfabeto. Un *shift (o subshift) de árboles* X *generado por* $\mathcal{F}$ es aquel y sólo aquel que tiene a todos los árboles que no tienen ningún bloque que aparece en $\mathcal{F}$, donde $\mathcal{F} \subseteq \mathcal{B}(\mathcal{T}(\mathcal{A}))$. Es decir $X = \mathsf{X}_\mathcal{F} = \{t \in \mathcal{T}(\mathcal{A}) | \forall f \in \mathcal{B}(X), f \notin \mathcal{F}\}$.

Como probamos en el capítulo 1 los shifts son σ-invariantes. Un *shift de árboles de tipo finito TSFT* es un shift de árbol para el cual existe un conjunto de bloques prohibidos $\mathcal{F}$ finito que lo genera.

De aquí en adelante, para poder simplificar la notación, trabajaremos únicamente con shifts con esqueleto $\underline{\mathsf{S}_2}^*$ lo que nos forzará a hablar únicamente de 2-árboles, pero todos los resultados pueden extenderse de manera sencilla a $\underline{\mathsf{S}_n}^*$.

Ejemplo 4.1. Consideremos el espacio shift de 2-árboles (es decir, $\underline{\mathsf{S}_n} = \underline{\mathsf{S}_2}$) generado a partir de un conjunto de palabras prohibidas $\mathsf{X}_\mathcal{F} = X$. Esto significa que nuestros esqueletos de árbol serán todos binarios y tendrán en cada nudo una ramificación a la izquierda y otra a la derecha. Definimos $\mathcal{A} = \{0, 1\}$ nuestro alfabeto y sea $\mathcal{F}$ el conjunto de bloques prohibidos que tiene a todos los bloques de altura 2 para los que la suma de sus símbolos es dos, es decir $\mathcal{F} = \{(s_1, s_2, s_3) \in \mathcal{B}_2(\{0, 1\})^{\underline{\mathsf{S}_2}^*} \,|\, s_1 + s_2 + s_3 = 2\} = \{\ (1, 1, 0), (1, 0, 1), (0, 1, 1)\}$ lo que significa que un punto de X tiene una suma de símbolos impar o cero en todos sus bloques de altura 2. A la izquierda de la imagen explicativa de la próxima página, presentamos los bloques prohibidos gráficamente, a la derecha se encuentra un punto de ejemplo que representa a la función descrita en la esquina superior derecha:



Hay que notar que en este caso el alfabeto tiene los mismos símbolos que $\underline{\mathsf{S}_2}^*$, sin embargo, aquellas gráficas de árbol que tienen únicamente un símbolo por nudo son árboles y las que tienen una cadena en cada nudo son esqueletos de árbol. Si perdemos de vista en algún momento esta observación llegaremos a grandes confusiones. Pueden aquí ejemplificarse casi todos los objetos definidos a lo largo de este capítulo, por ejemplo, un bloque de altura 2 es cualquiera de los prohibidos y uno de altura n es un triángulo con todo su interior en el que en cada arista hay n nudos. El árbol es la función t definida en la parte superior derecha del dibujo y la representación gráfica del árbol t es la estructura dibujada abajo a la derecha. También podemos ver los bloques en el mismo orden, ya sea como arreglo o dentro de una gráfica. En esta figura no tenemos esqueletos de árbol.

Así pues, tenemos todo para seguir adelante y presentar los códigos de bloques, último paso para tener estructurada ya una teoría de espacios shift. El concepto es una traducción del que habíabmos construído en el capítulo 1, excepto que ahora nuestros bloques son triángulos en gráficas de árbol.

Definición 4.12. Sea $n \in \mathbb{N}$ fija de tal manera que $\underline{\mathsf{S}_n}^*$ es fijo para cualquier shift que se mencione. Sean $\mathcal{A}, \mathcal{A}'$ dos alfabetos y $X \subseteq \mathcal{T}(\mathcal{A})$ un espacio shift (es decir, X está generado por un conjunto de palabras prohibidas $\mathcal{F}$) y sea $m \in \mathbb{N}$. Una función $\Phi : X \to \mathcal{T}(\mathcal{A}')$ es llamada un *código local de m-bloques* siempre que exista una función $\phi : \mathcal{B}_m(X) \to \mathcal{A}'$ de tal manera que para cualquier $s \in \underline{\mathsf{S}_n}^*$ ocurre que $\Phi(t(s)) = \phi(b)$ donde b es justamente el bloque

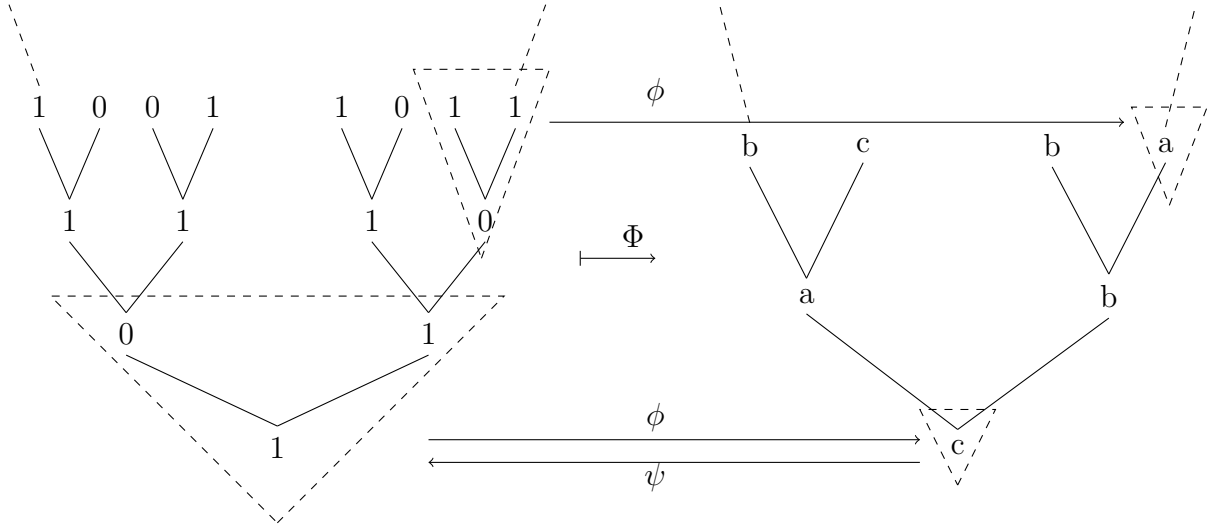
$f(m, s)$ de altura m, enraizado en s del árbol t. El menor entero m para el que se satisface la condición de existencia de ϕ es llamado la *meomoria* del código local de m-bloques Φ. Un *código de bloques* es un código local de bloques que es m-local para alguna $m \in \mathbb{N}$.

El teorema de Curtis-Lyndon-Hedlund dice que los códigos de bloques son exactamente las funciones $\Phi : X \to Y$ que son continuas y conmutan con las funciones shift, es decir $\sigma_i(\Phi(t)) = \Phi(\sigma_i(t))$ para cualquier árbol t e $i \in \underline{\mathsf{S}_n}$. La imagen de un espacio shift de árboles bajo de un código de bloques es también un espacio shift de árboles. La definición estrella del capítulo es la siguiente:

Definición 4.13. Sean X, Y espacios shift de árbol sobre $\underline{\mathsf{S}_n}^*$. Un código de bloques $\Phi : X \to Y$ que sea inyectivo y suprayectivo (biyectivo) y que tenga una inversa que también es un código de bloques es llamado una *conjugación de X en Y*. Siempre que existe una conjugación de X en Y decimos que X *es conjugado a* Y y viceversa (la relación es simétrica) y la escribimos como $X \approx Y$.

Así pues, podemos convertir un árbol en otro a partir de códigos que para cada bloque asignan una nueva letra. Ahora presentamos un ejemplo de código y después definimos a las presentaciones por bloques, que son los códigos de bloques de mayor simpleza y de mucha utilidad.

Ejemplo 4.2. Sea X el TSFT formado como $\mathsf{X}_{\mathcal{F}} = X$, con $\mathcal{F} = \{(s_1, s_2, s_3) \in \mathcal{B}(\{0,1\}^{\underline{\mathsf{S}_2}^*}) | s_1 + s_2 + s_3 \equiv 1 \bmod 2\} = \{(1,0,0), (0,1,0), (0,1,0), (1,1,1)\}$, el shift de árboles que tiene prohibidos los bloques de altura 2 para los que la suma de sus símbolos es impar. Sea Y el shift completo sobre el alfabeto $\{a, b, c\}$ donde los bloques permitidos son $\mathcal{F}^c = \{(a,a,a), (a,b,c), (a,c,b),$
$(a,c,c), (b,b,a), (b,c,a), (c,a,b), (c,a,c)\}$ (cuidado, hemos definido el complemento de $\mathcal{F}$ así que $\mathcal{F}$ tiene todos los bloques de altura 2 que no aparecen en su complemento). El 2-código de bloques $\Phi : X \to Y$ definido por $\phi : \mathcal{B}_2(X) \to \{0,1\}$ donde $\phi(0,0,0) = a$, $\phi(0,1,1) = a$, $\phi(1,1,0) = b$, $\phi(1,0,1) = c$, está representado en la figura transformando un punto $x \in X$ en un punto $y \in Y$. Importante es observar que Φ es una conjugación con inversa Ψ definida bajo ψ como $\psi(a) = 0, \psi(b) = 1, \psi(c) = 1$.



Definición 4.14. Sea X espacio shift sobre el alfabeto $\mathcal{A}$ y sea $k \in \mathbb{N}$. La *presentación en k bloques de X* es un espacio shift de árboles sobre el alfabeto $\mathcal{B}_k(X)$ definido como:

$$B_k(X) = \{t' \in \{\mathcal{B}_k(X)\}^{\underline{\mathsf{S}_n}^*} | \exists t \in X \ni \forall s \in \underline{\mathsf{S}_n}^*, t'(s) = f(k, t(s))\}$$

Naturalmente cada bloque es conjugado a cada una de sus presentaciones en bloques bajo la función $\gamma : \mathcal{T}(X) \to \mathcal{T}(\mathcal{B}_k(X))$ tal que $\gamma(a) = f(k, a)$ cuya inversa es $\gamma' : \mathcal{T}(\mathcal{B}_k(X)) \to \mathcal{T}(X)$ donde $\gamma'(b_1, ..., b_k) = b_1$.

4.2. El teorema de descomposición para espacios shift de árboles

Hemos llegado al teorema central del trabajo. Probaremos que toda conjugación entre TSFTs se puede descomponer en una sucesión de amalgamaciones y separaciones como hicimos en el capítulo 2. Para ello necesitamos definir amalgamación y separación y después probar dos lemas, el primero nos ayuda a descomponer una conjugación de memoria m en una de memoria $m-1$ compuesta con una separación y el segundo hace lo mismo con su inversa. He allí la clave. Una vez probados estos lemas los aplicaremos m veces para descomponer a la conjugación en un renombramiento y puras separaciones. Vamos a ello.

Sea X un shift de árboles sobre un alfabeto $\mathcal{A}$. Pensemos en los bloques de tamaño n de X, $\mathcal{B}_n(X)$. Para cada letra $a \in \mathcal{A}$ podemos formar el conjunto de los bloques de tamaño n que comienzan con a y llamarlo $\mathcal{B}_n^a(X)$. Ahora formemos una partición cualquiera de $\mathcal{B}_n^a(X)$

que llamaremos $\mathcal{P}_a$. Es bueno resaltar que $\bigcup\bigcup_{a\in\mathcal{A}}\mathcal{P}_a = \mathcal{B}_n(X)$ [2]de donde obtenemos que $\bigcup_{a\in\mathcal{A}}\mathcal{P}_a = \mathcal{P}$ es una partición de $\mathcal{B}_n(X)$ pues todos sus subconjuntos son no vacíos y ajenos. Queremos resaltar la partición especial donde para cada $a \in \mathcal{A}, \mathcal{P}_a$ es la partición total, es decir, cada bloque que empieza con a es el único de su clase y por lo tanto $\mathcal{P}$ tiene un conjunto con un único elemento por cada bloque de $\mathcal{B}_n(X)$. Estas particiones son las que definirán ahora nuestras separaciones en analogía con lo que hacíamos con los vértices de las gráficas de los shift de tipo finito del capítulo 2.

Definición 4.15. Sea X TSFT y sea $\mathcal{P}$ una partición de sus bloques de tamaño 2 como la que mostramos en el párrafo anterior, de forma que $\mathcal{P} = \cup_{a\in\mathcal{A}}\mathcal{P}_a \subseteq \wp(\mathcal{B}_2(X))$. Luego $\cup\mathcal{P} = \mathcal{B}_2(X),\ p_i \cap p_j = \emptyset$ siempre que $i \neq j,$ y para toda $i,\ p_i \neq \emptyset$. Sea $\Phi : X \to \mathcal{P}^{\underline{\mathsf{S}_2}^*}$ de tal forma que $\phi(a, s_1, s_2) = [(a, s_1, s_2)]_a$. Al shift $\Phi(X)$ le llamaremos $\tilde{X}$. Diremos que Φ es una *función de separación* y que $\tilde{X}$ y cualquier renombramiento de los elementos de $\tilde{X}$ es una *separación interior de* X. La función inversa $\Phi^{-1} : \mathcal{P}^{\underline{\mathsf{S}_2}^*} \to X$ es aquel código de 1-bloques tal que $\Phi^{-1}([(a, s_1, s_2)]_a) = a$, al que llamamos una *función de amalgamación* y cuando dicho código existe decimos que X es una *amalgamación* de $\tilde{X}$.

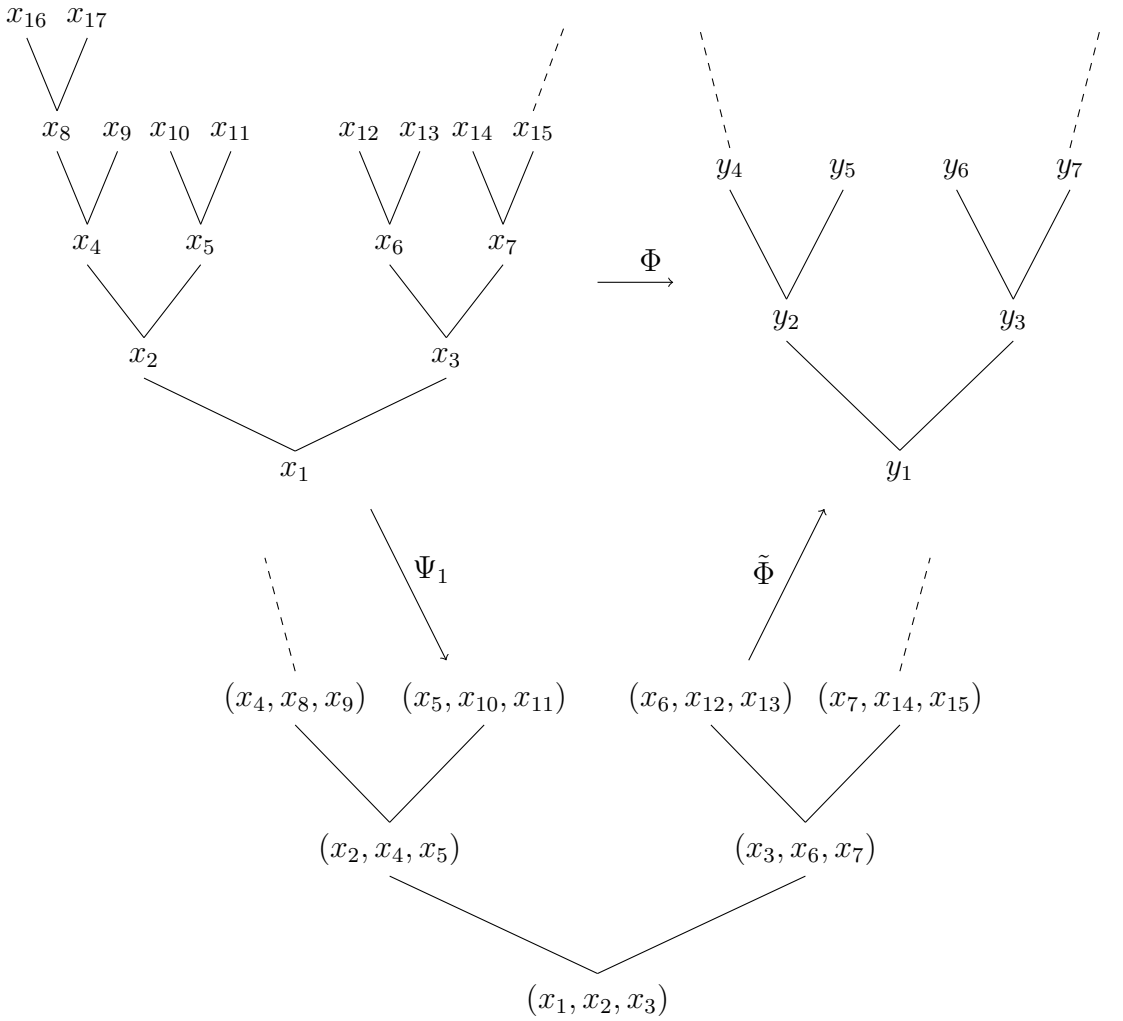
Es muy importante resaltar que cuando la partición $\mathcal{P}$ es la discreta obtenemos $\tilde{X} = B_2(X)$, nuestra presentación en 2-bloques de X. Esto nos dice que nuestra generalización es una buen intento, pues coincide con el resultado del capítulo 2. Por el momento todas nuestras separaciones son códigos de 2-bloques pero se pueden generalizar. Recordemos que a una conjugación de 1-bloques le llamamos una *función de renombramiento* pues no modifica la estructura del árbol. Vamos por el primer lema.

Lema 4.1. *Sea $\Phi : X \to Y$ una conjugación de m-bloques entre dos espacios shift X, Y con $m \geq 2$. Entonces existe una separación interior $\Psi_1 : X \to \tilde{X}$ y un $(m-1)$-código de bloques $\tilde{\Phi} : \tilde{X} \to Y$ tal que $\Phi = \tilde{\Phi} \circ \Psi_1$.*

Demostración. Supongamos que $\mathcal{A}$ es el alfabeto del shift X y $\mathcal{A}'$ el del shift Y. Sea $\phi : \mathcal{B}_m(X) \to Y$ la función que define a Φ (la definiremos más adelante). Sea $\Psi_1 : X \to \tilde{X}$ la conjugación de 2-bloques tal que $\Psi_1((s_1, s_2, s_3)) = (s_1, s_2, s_3)$ y sea $\mathcal{P}$ la partición total. La función Ψ_1 es una separación completa que nos lleva a que $\tilde{X}$ se convierta en su presentación por 2-bloques $B_2(X)$. Sean además las funciones $g, g_0, g_1 : \mathcal{B}_2(X) \to \mathcal{A}$ tales que $g((s_1, s_2, s_3)) = s_1$, $g_0((s_1, s_2, s_3)) = s_2$, $g_1((s_1, s_2, s_3)) = s_3$. Es bueno ver que la notación es intencional pues el subíndice nos refiere al símbolo que encontramos al aplicar la función shift a la imágen de g. Sea $\tilde{\Phi} : \tilde{X} \to Y$ el código de $(m-1)$-bloques definido como sigue: para cada bloque $f(m-1, (a, b, c)) \in \mathcal{B}_{m-1}(\tilde{X})$ tomamos $\tilde{\phi}(f(m-1, (a, b, c))) = \phi(l(m, a))$

[2]Unimos dos veces porque la primera unión junta todos los conjuntos que tiene cada uno de los $\mathcal{P}_a$ para a fija y la segunda corre sobre todos estos conjuntos que últimos términos lo único que están haciendo es separar a todos los bloques de tamaño n.

donde $l(m,a) \in \mathcal{B}_m(X)$ es tal que $l(m-1,a) = (g(f(m-1,(a,b,c))))$ y además $l(1,\sigma_s(a)) = g_0(f(sa))$ cuando $s \in \underline{\mathsf{S}_2}^{|a|+(m-1)}$ termina en 0 y $l(1,\sigma_s(a)) = g_1(f(sa))$ cuando s termina en 1. Es decir, reconstruímos el bloque de X hasta $m-1$ usando los primeros símbolos de la presentación por bloques y después añadimos una fila abajo usando los últimos símbolos dentro de los últimos elementos de la presentación por bloques. Tenemos que $\Phi = \tilde{\Phi} \circ \Psi_1$ como se explica en el dibujo:



□

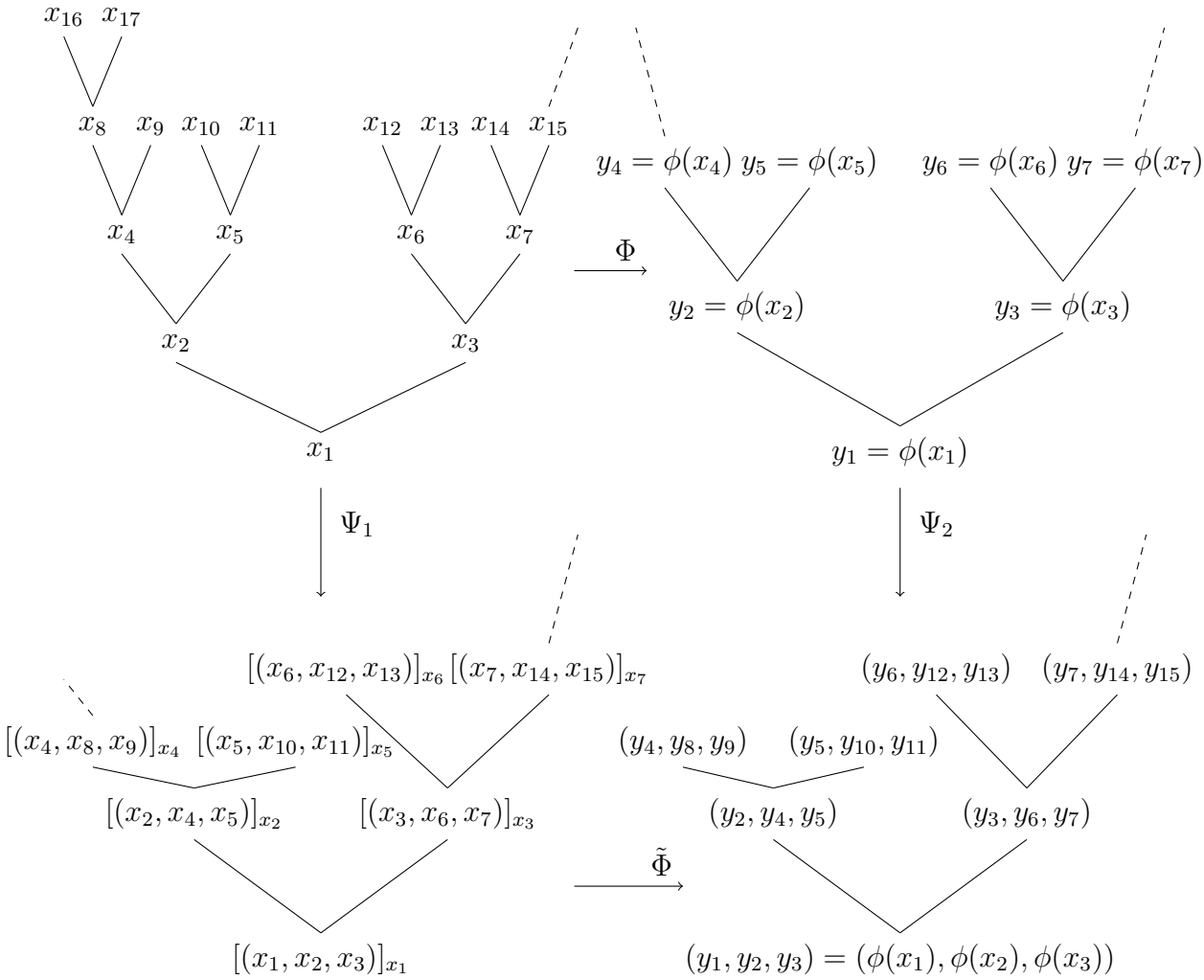
Lema 4.2. *Sea $\Phi : X \to Y$ una conjugación entre ambos espacios que es un código de 1-bloques y tal que Φ^{-1} es un código de m-bloques con $m \geq 2$. Entonces hay códigos de separación $\Psi_1 : X \to \tilde{X}$ y $\Psi_2 : Y \to \tilde{Y}$ y una conjugación de 1-bloques $\tilde{\Phi} : \tilde{X} \to \tilde{Y}$ tales que $\Phi = \Psi_2^{-1} \circ \tilde{\Phi} \circ \Psi_2$ y tal que $\tilde{\Phi}^{-1}$ es un $(m-1)$-código de bloques. Es decir, el siguiente*

diagrama conmuta:

$$\begin{array}{ccc} X & \xrightarrow{\Phi} & Y \\ {\scriptstyle\Psi_1}\downarrow & & \downarrow{\scriptstyle\Psi_2} \\ \tilde{X} & \xrightarrow[\tilde{\Phi}]{} & \tilde{Y} \end{array}$$

Demostración. Sea $\mathcal{A}$ el alfabeto del espacio shift X y sea $\mathcal{A}'$ el de Y. Para cada $a \in \mathcal{A}$ definimos la partición $\mathcal{P}_a$ de $\mathcal{B}_2^a(X)$ de tal forma que $[(a,b,c)]_a = \{(a,s_1,s_2) \in \mathcal{B}_2^a(X) \mid \phi(b) = \phi(s_1)$ y $\phi(c) = \phi(s_2)\}$ donde $\phi : \mathcal{A} \to \mathcal{A}'$ es la función que define a Φ. Formemos la siguiente partición de los bloques de tamaño 2 de X: $\mathcal{P} = \bigcup_{a\in\mathcal{A}} \mathcal{P}_a$. Sea $[(a,b,c)]_a$ la clase del bloque (a,b,c) en $\mathcal{P}$. Definimos un nuevo shift como subespacio del shift completo $\mathcal{P}^{\underline{\mathsf{S}_2}^*}$ como sigue: sea $\Psi_1 : X \to \mathcal{P}^{\underline{\mathsf{S}_2}^*}$ el 2-código de bloques de tal forma que $\Psi_1(a,b,c) = [(a,b,c)]_a$. La función está bien definida pues cada 2-bloque tiene una clase en la partición $\mathcal{P}_i$ donde $i \in \mathcal{A}$ es su primera letra. Tenemos entonces nuestro nuevo shift $\tilde{X} = \Psi_1(X)$ que tiene como puntos a árboles (binarios) con clases de equivalencia colocadas en los nudos. $\tilde{X}$ es una separación interior de X.

Ahora definamos $\tilde{Y}$ como la presentación en 2-bloques de Y. Es decir sea $\mathcal{P}_a$ la partición que a cada (a,b,c) le asigna $[(a,b,c)]_a$ la clase que lo tiene únicamente a él como elemento, lo que genera lo que habíamos llamado $B_2(Y)$. Llamamos pues $\Psi_2 : Y \to \mathcal{B}_2(Y)$ la función definida como $\Psi_2((a,b,c) = [(a,b,c)]_a)$ que genera la presentación en 2-bloques $\Psi_2(Y) = \tilde{Y}$. Consideremos entonces el código de 1-bloques $\tilde{\Phi} : \tilde{X} \to \tilde{Y}$ como $\tilde{\Phi}([(a,b,c)]_a) = (\phi(a),\phi(b),\phi(c))$. La imágen de la clase no depende del representante pues justamente en $[\,]_a$ estan únicamente bloques cuya imágen es igual bajo ϕ por la forma que definimos $\mathcal{P}_a$. Tenemos que $\Phi = \Psi_2^{-1} \circ \tilde{\Phi} \circ \Psi_1$ tras efectuar el sencillo cálculo siguiente: $\Phi(a) = \phi(a)$ y $\Psi_2^{-1} \circ \tilde{\Phi} \circ \Psi_1(a) = \Psi_2^{-1}(\tilde{\Phi}([(a,b,c)]_a)) = \Psi_2^{-1}(\phi(a),\phi(b),\phi(c)) = \phi(a)$. Ahora nos falta verificar que $\tilde{\Phi}^{-1} = \Psi_1 \circ \Phi^{-1} \circ \Psi_2^{-1}$ es un código de $(m-1)$-bloques. Es decir, debemos probar que para cualquier árbol t en $\tilde{Y}$, el bloque $f_{\tilde{Y}}(m-1,\epsilon)$ determina completamente el elemento $\tilde{\Phi}^{-1}(t(\epsilon))$, pero ésto se sigue de que nuestro bloque $f_{\tilde{Y}}(m-1,\epsilon)$ determina a todos los $\Psi_2^{-1}(t(x0))$ y a todos los $\Psi_2^{-1}(t(x1))$ una vez dado $x \in \underline{\mathsf{S}_2}^{(m-1)}$, y luego, el bloque de altura m enraizado en ϵ del árbol $\tilde{\Phi}^{-1}(t) \in \mathcal{T}(Y)$. Entonces como Φ^{-1} es un código de m-bloques, queda derterminado el elemento $t'(\epsilon) = (\Phi^{-1} \circ \Psi_2^{-1})(t(\epsilon)) \in \mathcal{A} = \mathcal{B}_0(X)$ del árbol $t' = (\Phi^{-1} \circ \Psi_2^{-1})(t)$. Pero para tener un nudo de un árbol en $\tilde{X}$ necesitamos una clase de la partición que usamos para definir Ψ_1 que es justamente elemento de $\mathcal{P}_{t'(\epsilon)}$. La buena noticia es que ésta ya está definida a partir de nuestro $(m-1)$-bloque en $\tilde{Y}$ y es $[(t'(\epsilon),\phi(t'(0)),\phi(t'(1)))]_{t'(\epsilon)}$, pues curiosamente $\phi(t'(0))$ es $\Psi_2^{-1}(f(0,0))$ y $\phi(t'(1)) = \Psi_2^{-1}(f(0,1))$ los cuales ya conocemos porque los hemos obtenido en el primer paso (debido a que $m \geq 2$), y así, nuestra $\tilde{\Phi}^{-1}$ tiene memoria $m-1$.



□

Teorema 4.1. *TEOREMA DE DESCOMPOSICIÓN PARA TSFTS. Cualquier conjugación entre dos shifts de árbol puede descomponerse en una sucesión de códigos de separación interiores y códigos de amalgamación interiores.*

Demostración. Sea $\Phi : X \to Y$ una conjugación de n-bloques entre dos shifts de árboles de tipo finito X y Y, tal que su inversa $\Phi^{-1} : Y \to X$ es un m-código de bloques con $m, n \geq 1$. Gracias a los lemas anteriores 4.2 y 4.1 esta prueba se hace bastante corta. Primero tenemos un conjunto de $n-1$ separaciones Ψ_i interiores si aplicamos el lema 4.1 recursivamente, que separan a $\tilde{X}_{i-1}$ convirtiéndolo en $\tilde{X}_i$ con $1 \leq i \leq n-1$ y $\tilde{X}_0 = X$ en los que cada una de las separaciones induce un nuevo código de $(n-i)$-bloques llamado $\tilde{\Phi}_{n-i}$ que nos lleva desde $\tilde{X}_i$ a Y. También, por el lema 4.2, tenemos un conjunto de $m-1$ separaciones

Ψ_{n-1+i} con $1 \leq i \leq m-1$ que transforman $\tilde{X}_{n-2+i}$ en $\tilde{X}_{n-1+i}$ y para las cuales hay una correspondiente Δ_i que transforma a $\tilde{Y}_{i-1}$ en $\tilde{Y}_i$ donde $\tilde{Y}_0 = Y$ de tal forma que la función $\tilde{\Phi}_{n-2+i} : \tilde{X}_{n-2+i} \to \tilde{Y}_{i-1}$ es una conjugación de 1-bloques cuya inversa $\tilde{\Phi}^{-1}_{n-2+i}$ es un $(m-i)$-código de bloques. De allí que la función $\Gamma = (Id)^{n-1} \circ \Delta_1^{-1} \circ ... \circ \Delta_{n-1}^{-1} \circ \tilde{\Phi}_{n+m-2} \circ \Psi_{n+m-2} \circ ... \circ \Psi_1$ es una sucesión de conjugaciones y amalgamaciones que hace lo mismo que Φ. Es importante resaltar que $\tilde{\Phi}_{n+m-2}$ es un renombramiento pues es un código de 1-bloques con inversa de memoria 1 y por lo tanto es considerado un caso trivial de separación interior.

$$\begin{array}{ccc}
X = \tilde{X}_0 & \xrightarrow{\tilde{\Phi}} & Y = \tilde{Y}_0 \\
\Big\downarrow{\scriptstyle \Psi_1} & & \Big\downarrow{\scriptstyle Id} \\
\vdots & & \vdots \\
\Big\downarrow{\scriptstyle \Psi_{n-1}} & & \Big\downarrow{\scriptstyle Id} \\
\tilde{X}_{n-1} & \xrightarrow{\tilde{\Phi}_{n-1}} & Y \\
\Big\downarrow{\scriptstyle \Psi_n} & & \Big\downarrow{\scriptstyle \Delta_1} \\
\tilde{X}_n & \xrightarrow{\tilde{\Phi}_n} & \tilde{Y}_1 \\
\Big\downarrow{\scriptstyle \Psi_{n+1}} & & \Big\downarrow{\scriptstyle \Delta_2} \\
\vdots & & \vdots \\
\Big\downarrow{\scriptstyle \Psi_{n+m-2}} & & \Big\downarrow{\scriptstyle \Delta_{m-1}} \\
\tilde{X}_{n+m-2} & \xrightarrow{\tilde{\Phi}_{n+m-2}} & \tilde{Y}_{m-1}
\end{array}$$

□

4.3. Autómatas de árbol

Queremos presentar en esta sección a los autómatas de árbol, que son una categoría especial dentro de los autómatas. Aunque siguen siendo autómatas, se diferencían de los que hemos presentado antes en que su función de transición toma varios vértices y un símbolo de entrada, en lugar de un vértice y un símbolo de entrada, para asociarles un nuevo vértice. Es decir, queremos un objeto que cada vez que se encuentre en un estado (que a su vez es un conjunto de condiciones) y reciba una entrada nos lleve a otro estado. En específico presentaremos a los autómatas que realizan cálculos de abajo hacia arriba en un árbol, es

decir, toman varios estados previamente calculados en $t(ai)$ donde $ai \in \underline{\mathsf{S}_n}^{(k+1)}$ y recibiendo como entrada al símbolo $a \in \mathcal{A}$ nos llevan a un nuevo estado $q_a \in \underline{\mathsf{S}_n}^{k}$.

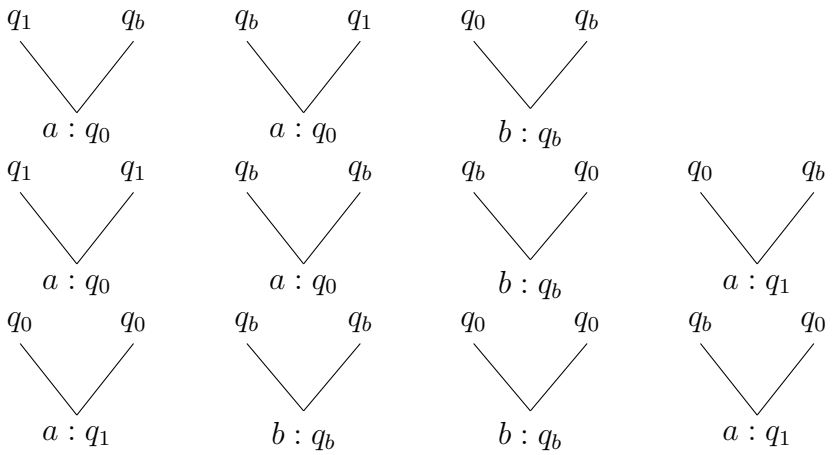
Definición 4.16. Un *autómata de árbol* es una estructura $\mathfrak{A} = (\mathcal{V}, \mathcal{A}, \Delta)$ donde $\mathcal{V}$ es un conjunto finito de vértices o estados, $\mathcal{A}$ es un conjunto de entradas (que normalmente será el alfabeto del árbol) y Δ es un conjunto de transiciones que escribiremos como $(q_1, ..., q_n), a \to q$ que se lee"la transición que sale de $(q_1, ..., q_n)$ y llega a q bajo la entrada a" y donde $q_1, ..., q_n, q \in \mathcal{V}$ y $a \in \mathcal{A}$.

En este caso no tendremos un conjunto de estados iniciales o finales, como ocurría en la definición anterior de autómata, lo que significa que todos los estados son tanto iniciales como finales, es decir, un proceso del autómata puede empezar leyendo cualquier vector de estados y cualquier símbolo de entrada. Nuestro autómata se llama *autómata determinista* siempre que el conjunto de transiciones pueda expresarse como una función, es decir, para cada $(q_1, ..., q_n) \in \mathcal{V}^n$ y para cada $a \in \mathcal{A}$ existe un único $q \in \mathcal{V}$ y por lo tanto una única transición tal que $(q_1, ..., q_n), a \to q$. Cuando se cumple esta condición definimos una función $\delta : \mathcal{V}^n \times \mathcal{A} \to \mathcal{V}$ de tal manera que $\delta((q_1, ..., q_n), a)) = q$ siempre que $(q_1, ..., q_n), a \to q \in \Delta$ y $\delta((q_1, ..., q_n), a)) = m_0 \in \mathcal{V}$ donde m_0 es un estado muerto en el que se termina el proceso del autómata cuando $(q_1, ..., q_n), a \to q \notin \Delta$. A los autómatas deterministas los escribiremos naturalmente como $\mathfrak{A} = (\mathcal{V}, \mathcal{A}, \delta)$.

Ahora consideremos un alfabeto $\mathcal{A}$ y un árbol $t \in \mathcal{T}(\mathcal{A})$. Podemos escoger un patrón p de t, es decir, un conjunto de hojas (elementos de $\mathcal{A}$) colocadas en un conjunto $\mathcal{N}$ de nudos (elementos de $\underline{\mathsf{S}_n}^*$) que es cerrado bajo prefijos, $p = t|_{\mathcal{N}} : \mathcal{N} \to \mathcal{A}$. Quisiéramos ver si este patron puede ser leído por el autómata. Para ésto, sobre el mismo esqueleto de árbol podemos colocar un patrón c (que recordemos que es una función de $\mathcal{N}$ en un alfabeto) sobre el alfabeto de vértices $\mathcal{V}$ de tal manera que $c : \mathcal{N} \to \mathcal{V}$. Si podemos encontrar un patrón c sobre el alfabeto $\mathcal{V}$ de vértices del autómata $\mathcal{A}$ de tal manera que para cada nudo $x \in \mathcal{N}$ existe una transición $(c(x0), ..., c(x(n-1)), p(x) \to c(x) \in \Delta$ diremos que dicho patrón es aceptado por el autómata $\mathcal{A}$. A dicho patrón c le llamaremos un *cómputo finito ascendente de* $\mathfrak{A}$ *en* p. De igual manera, un *cómputo de* $\mathfrak{A}$ en un árbol infinito t consiste en encontrar un árbol c sobre el mismo esqueleto de árbol que es el dominio de t, pero con alfabeto $\mathcal{V}$ de tal manera que para cada nudo $x \in \underline{\mathsf{S}_n}^*$ exista una transición $(c(x0), ..., c(x(n-1)), t(x) \to c(x) \in \Delta$. Diremos que un árbol t es aceptado por $\mathfrak{A}$ siempre que exista un cómputo de $\mathfrak{A}$ en t.

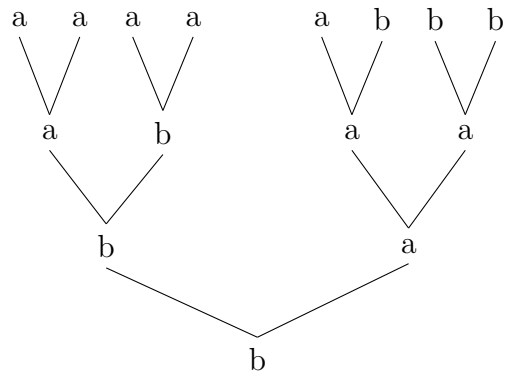
Ejemplo 4.3. Sea X el espacio shift sobre el alfabeto $\{a, b\}$ donde un bloque está prohibido cuando tenga una rama con una cantidad impar de letras a entre dos letras b. Queremos comprobar que éste es un espacio shift, pero parece que es muy difíci dar un conjunto de bloques prohibidos, pues éstos deben prohibir cualquier rama con un numero impar de letras a y éstas son infinitas, por lo que preferimos diseñar un autómata que reconozca justamente a los árboles que pertenecen a X. Vamos a tomar $\mathfrak{A} = (\mathcal{V} = \{q_b, q_0, q_1\}, \mathcal{A} = \{a, b\}, \Delta)$ donde

las transiciones de Δ son las que hemos escrito abajo con una notación muy natural. Por ejemplo, el triángulo de la esquina superior izquierda es otra escritura de $((q_1, q_b), a \to q_0)$.

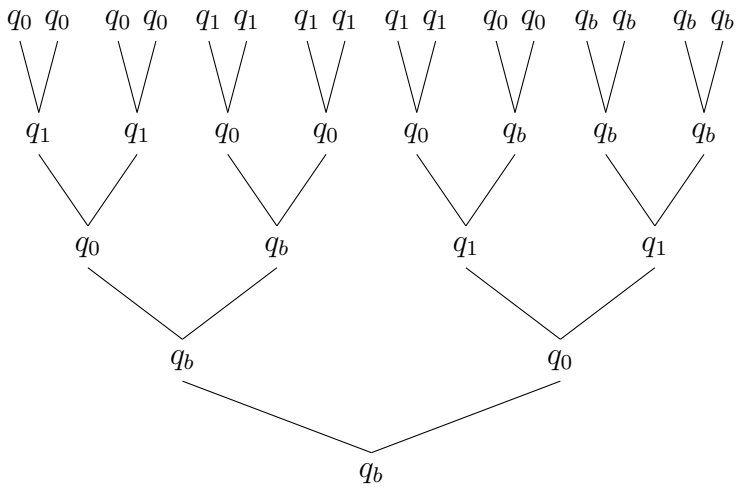


Transiciones en Δ.

Es importante analizar cómo fue construído el autómata. El estado q_b se alcanza únicamente cuando leemos una letra b. Además el estado q_0 puede alcanzarse únicamente cuando dos ramas con una cantidad par de letras a desde la última b convergen. El estado q_1 significa que en alguna de las ramas hemos contado una cantidad impar de letras a y, por lo tanto, estamos forzados a añadir una a. Es por eso que del estado q_1 nunca podemos pasar a un estado q_b. Ilustramos abajo un bloque permitido de este espacio y podemos revisar que cumple la condición de tener un cómputo en el autómata $\mathfrak{A}$ que lo convierte en un bloque aceptado.



Un bloque permitido en X.



Un cómputo en $\mathfrak{A}$ para el bloque de arriba.

Ahora vamos a pasar a la analogía de los shifts de M-pasos bajo esta nueva presentación. Recordemos que esta clase de shifts siempre se puede representar como un shift por aristas tras aplicarle una presentación por bloques. Esto nos hizo muy fáciles los cálculos y nos permitió usar las matrices de adyacencia. Una simplificación parecida tendrán los shifts de

árbol que cumplan con la siguiente propiedad:

Definición 4.17. Sea $m \in \mathbb{N}^+$. Un *autómata de árbol determinista m-local*, o también *autómata m-definido de árbol*, es un autómata determinista de árbol $\mathfrak{A} = (\mathcal{V}, \mathcal{A}, \delta)$ que cumple lo siguiente: siempre que dos árboles $t, t' \in \mathcal{T}(\mathcal{A})$ sean aceptados por $\mathfrak{A}$ con el mismo bloque $f(m, \epsilon)$, tenemos que para cualquier cómputo c de $\mathfrak{A}$ sobre t y cualquier cómputo c' de $\mathfrak{A}$ sobre t' ocurre que $c(\epsilon) = c'(\epsilon)$. Recordemos que el cómputo es una función de los nudos a los vértices del autómata, por lo que ésta última condición pide que ambos cómputos valgan lo mismo en ϵ, es decir, que todo cómputo termine en el mismo vértice. Así pues, la memoria de altura m determina el estado al que llegaremos y por eso decimos que nuestro bloque $f(m, \epsilon)$ se enfoca al estado $c(\epsilon)$. Un autómata de árbol es *local* siempre que sea m-local para alguna m.

Ahora veremos una propiedad importante de los autómatas locales de árbol: podemos caracterizar a los shifts de árbol de tipo finito (TSFT) como todos aquellos que pueden reconocer estos autómatas y viceversa.

Proposición 4.3. *Todo shift de árbol de tipo finito (TSFT) es aceptado por un autómata local de árbol y además todo shift de árbol que sea aceptado por un autómata local de árbol es de tipo finito.*

Demostración. Sea $X = \mathsf{X}_{\mathcal{F}}$ un espacio shift de árbol de tipo finito (TSFT) definido por $\mathcal{F}$, un conjunto finito de bloques prohibidos sobre el alfabeto $\mathcal{A}$. Sin pérdida de generalidad podemos asumir que $\mathcal{F}$ tiene puros bloques de tamaño m; para alguna $m \geq 2$, pues en caso contrario, escogemos el bloque más grande $f(k, x)$ de nuestro conjunto finito y agregamos todos aquellos bloques de tamaño k sobre el alfabeto de X que tengan como subbloque algún bloque de $\mathcal{F}$ y tomamos $m = k$ (ya hicimos esto en el capítulo 1).

Definimos al autómata determinista de árbol $\mathfrak{A} = (\mathcal{V}, \mathcal{A}, \delta)$ tal que $\mathcal{V} = \mathcal{B}_{m-1}(X)$. Para $f_0(m-1, \epsilon), f_1(m-1, \epsilon) \in \mathcal{V}$ y $a \in \mathcal{A}$, si el bloque $f(m, a) = (a, f_0, f_1)$ de altura m es un bloque permitido en X, entonces $\delta((f_0, f_1), a) = (a, f_0(m-2, \epsilon), f_1(m-2, \epsilon))$, donde este nuevo bloque es de altura $m-1$ y por lo tanto también pertenece a $\mathcal{V}$; pues es subbloque de uno permitido, $f(m, a)$. Cuando $f(m, a)$ no sea bloque permitido de X, el autómata entra a su estado muerto m_0 y el cómputo rechaza. Por construcción, éste autómata es $m-1$ local, pues siempre que toma un par de bloques iguales forma, utilizando el mismo símbolo de entrada, un nuevo bloque usando la información de los bloques iguales. Además, acepta al espacio shift de árbol X, pues justamente se encarga de verificar que los árboles no tengan bloques prohibidos en X; por lo tanto la primera parte de la proposición queda probada.

Ahora sea X un espacio shift de árbol y $\mathfrak{A} = (\mathcal{V}, \mathcal{A}, \delta)$ un autómata m-local determinista que acepta a X. Definimos entonces $\mathcal{F}$ como el conjunto de bloques prohibidos de altura

$m+1$ de X. Naturalmente $X \subseteq \mathsf{X}_{\mathcal{F}}$ pues X podría estar más restringido para aceptar sus árboles.

Supongamos que $t \in \mathsf{X}_{\mathcal{F}}$. Construímos un cómputo c de $\mathfrak{A}$ en t (para verificar que $\mathfrak{A}$ acepta a t) como sigue: para cualquier nudo $x \in \underline{\mathsf{S}_n}^*$, escogemos $c(x)$ como el vértice de $\mathfrak{A}$ al que se enfoca el bloque de altura m en t enraizado en x, o en otras palabras, tomamos el bloque en t de altura m y enraizado en x, y realizamos un cómputo (el cual existe porque $\mathfrak{A}$ acepta a t); luego escogemos el vértice que queda en punta y a ese lo asignamos como $c(x)$. Ahora consideremos $f_t(m+1, x)$, el bloque de t de altura $m+1$ enraizado en x (el subíndice nos indica que f es una restricción de t a un conjunto de nudos). Como $t \in \mathsf{X}_{\mathcal{F}}$, nuestro bloque $f_t(m+1, x)$ está permitido en X y, por lo tanto, existe un árbol $t' \in X$ que lo tiene como bloque, enraizado en algún nudo $y \in \underline{\mathsf{S}_2}^*$. Llamémosle $f_{t'}(m+1, y)$, porque precisamente es la misma función f aunque estemos cambiando el dominio a un conjunto de nudos enraizados en y. Para este árbol t' sí hay un cómputo c' en $\mathfrak{A}$, porque pertenece a X que está reconocido por $\mathfrak{A}$. Tenemos entonces que $\delta((c'(y0), c'(y1)), t'(y)) = c'(y)$ es la última transición del autómata $\mathfrak{A}$ cuando acepta $f'_t(m+1, y)$; pero como $\mathfrak{A}$ es m-local y está leyendo el mismo bloque de tamaño $m+1$, tenemos que $c(x0) = c'(y0)$, $c(x1) = c'(y1)$, $t(x) = t'(y)$ y que $c(x) = c'(y)$. Entonces $\delta((c(x0), c(x1)), t_x) = c(x)$ y en consecuencia, de forma descendente en el árbol, podemos construir un cómputo c realizando el proceso de nudo en nudo. Podemos ver que c es un cómputo de $\mathfrak{A}$ en t y por lo tanto $t \in X$ y $\mathsf{X}_{\mathcal{F}} \subseteq X$, de donde concluímos que $X = \mathsf{X}_{\mathcal{F}}$, lo que prueba la segunda parte de nuestra afirmación. □

Como se pudo ver durante esta sección, trabajar con autómatas y espacios shift no es fácil, pues involucran mucha notación y tenemos que movernos con mucha facilidad entre el alfabeto del shift y el alfabeto de los vértices del autómata. A continuación construiremos una simplificación de los autómatas y de los shifts que nos ayudarán a probar más rápido los teoremas que siguen. Si volvemos al ejemplo 4.2 nos podemos dar cuenta que en vez de construir un autómata con un conjunto de vértices complicados, podríamos diseñar uno que evaluara si fue colocado correctamente el único símbolo que puede ensamblarse en la parte superior de un bloque de altura 2; una vez que conocemos los dos símbolos inferiores. En base a este razonamiento, construímos la siguiente clase de shifts y autómatas.

Definición 4.18. Sea $\mathcal{A}$ un alfabeto. Un *shift de árbol por vértices* es el shift de árbol que resulta aceptado por un autómata $\mathfrak{A} = (\mathcal{V}, \mathcal{V}, \Delta)$, donde las transiciones tienen la forma $(q_0, q_1), q \to q$. Entonces el símbolo colocado en cada nudo de un árbol aceptado por el autómata es igual a la etiqueta del vértice correspondiente en el cómputo que realiza la aceptación. Podemos simplificar la notación diciendo que un shift de árbol por vértices es el conjunto de cómputos de un autómata no etiquetado $\mathcal{B} = (\mathcal{V}, \Gamma)$, con transiciones $\Gamma \subseteq \mathcal{V}^2 \times \mathcal{V}$ escritas como $(q_0, q_1) \to q$.

Ejemplo 4.4. Volvamos a nuestro ejemplo 4.2, en donde X acepta únicamente bloques de tamaño 2 para los que la suma de sus símbolos es par. Entonces podemos decir que X es

aceptado por el autómata $\mathfrak{A} = (\mathcal{V}, \Gamma)$ con las transiciones que están expresadas en la tabla de abajo. Por ejemplo, la transición $(0,1) \to 1$ viene del valor en la casilla en la fila cero y columna uno.

	0	1
0	0	1
1	1	0

Proposición 4.4. *Todo shift de árbol de tipo finito TSFT es conjugado a un shift de árbol por vértices.*

Demostración. Sea $X = \mathsf{X}_{\mathcal{F}}$ un shift de árbol de tipo finito, definido a partir de un conjunto de bloques prohibidos $\mathcal{F}$ de altura $m+1$ con $m \in \mathbb{Z}^+$. Sea $\mathfrak{A}$ el autómata determinista m local definido por $\mathcal{V} = \mathcal{B}_m(X)$, tal que para cada par de bloques de tamaño m, digamos $p_0(m,\epsilon), p_1(m,\epsilon) \in \mathcal{V}$ y $a \in \mathcal{A}$, la evaluación $\delta((p_0,p_1),a)$ es el bloque de altura m enraizado en ϵ descrito por $(a, p_0(m-1,0), p_1(m-1,1))$, siempre que el bloque de altura $m+1$ descrito por $(a, p_0(m,0), p_1(m,1))$ sea un bloque permitido en X, en otras palabras, que no pertenezca a $\mathcal{F}$, que envía a $\delta((p_0,p_1),a)$ a un estado muerto m_0 siempre que $(a, p_0(m,0), p_1(m,1)) \in \mathcal{F}$. Resaltemos que el autómata $\mathfrak{A}$ acepta a X y además dado un árbol fijo $t \in X$ hay un único cómputo de $\mathfrak{A}$ que acepta a t y es aquel que acabamos de presentar, porque para cada par de bloques específicos enraizados en ciertos nudos $x0$ y $x1$, con $x \in \underline{\mathsf{S}_2}^*$, hay un único símbolo $t(x) = a$ para el cual $\delta((p_0(m,x0), p_1(m,x1)), a)$ no es m_0.

Ahora vamos a construir un autómata $\mathfrak{B} = (\mathcal{V}, \Gamma)$ que genere un shift de árbol por vértices. Consideremos $\mathcal{V} = \mathcal{B}_m(X)$, y coloquemos en Γ a la transición

$$((p_0(m,x0), p_1(m,x1)), (a, p_0(m-1,x0), p_1(m-1,x1))) \in \mathcal{V}^2 \times \mathcal{V}$$

siempre que la tercera entrada sea aceptada por $\mathfrak{A}$ y, por lo tanto, sea una entrada de X. En síntesis, nuestro conjunto de transiciones Γ está formado por todos los cómputos válidos de $\mathfrak{A}$. Si volvemos a nuestra definición de shifts por vértices, podemos ver que $\mathfrak{B}$ tiene la forma de los autómatas que aceptan shifts por vértices. Sea Y el espacio shift de árbol por vértices que acepta el autómata $\mathfrak{B}$ para el que los vértices están etiquetados por bloques de tamaño m.

Definimos el código de m-bloques $\Phi : X \to Y$ a partir de la función de m-bloques $\phi : \mathcal{B}_m(X) \to Y$, donde $\phi(p) = p$. El código Φ le asigna a cada árbol t de X el único cómputo que hay de $\mathfrak{A}$ en t. Además el código de bloques $\Psi : Y \to X$ dado por $\psi(p) = p_\epsilon$ es la inversa de ϕ lo cual convierte a Φ en una conjugación. Luego entonces el espacio shift de árbol X y el espacio shift de árbol por vértices Y son conjugados. □

Con todas estas herramientas que hemos construído a partir de autómatas, estamos preparados para probar la decidibilidad de la conjugación entre shifts de árbol de tipo finito.

4.4. Decidibilidad de la conjugación para TSFTs

En esta sección probaremos que es posible decidir si dos TSFTs dados son conjugados. El punto clave está en que las separaciones interiores son conmutativas y que podemos trabajar sobre shifts de árbol por vértices gracias a la proposición final de la sección anterior. Vamos a hacer un repaso de las separaciones y amalgamaciones interiores para los shifts de árbol por vértices, para llegar a encontrar una amalgamación mínima entre cualesquiera dos shifts amalgamables.

Sea X un espacio shift de árbol por vértices aceptado por un autómata $\mathfrak{A} = (\mathcal{V}, \Delta)$, y sea Φ un código de separación interior de X en $\tilde{X}$. Recordemos que el conjunto $\tilde{X}$ resultaba ser la imagen bajo Φ de una partición de los bloques de tamaño 2 de X. Ahora mostraremos que cuando X es shift de árbol por vértices también lo es su separación $\tilde{X}$. Como X es un shift por vértices podemos hablar indistintamente de su alfabeto o de sus vértices porque son el mismo conjunto. Como acordamos en la definición 4.15, la partición en el código de separación se realizaba de tal manera que, para cada $r \in \mathcal{V}$, los bloques de altura 2 enraizados en r se parten en $l(r)$ partes que denotamos $r^1, ..., r^{l(r)}$ y a los que llamaremos emphlos vértices separados de r. Sea entonces $\tilde{\mathcal{V}}$ el conjunto de vértices separados. Ahora separaremos el conjunto $\Delta(r)$, que son todas las transiciones de nuestro autómata $\mathfrak{A}$ que entran a r, en $l(r)$ partes $\Delta(r)^1, ..., \Delta(r)^{l(r)}$, de acuerdo a la siguiente relación: $(p, q) \to r \in \Delta(r)^i$ si y sólo si $(r, p, q) \in r^i$, donde p, q, r son símbolos del alfabeto de X. Podemos construir un nuevo autómata, $\tilde{\mathfrak{A}} = (\tilde{\mathcal{V}}, \tilde{\Delta})$, donde para cada $1 \leq i \leq l(p)$, $1 \leq j \leq l(q)$ y cada $1 \leq k \leq l(r)$ tenemos que

$$(p^i, q^j) \to r^k \in \tilde{\Delta} \text{ si y sólo si } (p, q) \to r \in \Delta(r)^k.$$

Entonces $\tilde{X}$ es justamente el espacio shift por vértices que es aceptado por $\tilde{\mathfrak{A}}$. Resaltemos que siempre que $(p^k, q^s) \to r^i \in \tilde{\Delta}$ entonces $(p^k, q^s) \to r^j \notin \tilde{\Delta}$ para cualesquiera $p^k, q^s \in \tilde{\mathcal{V}}$ y cualquier $1 \leq i \neq j \leq l(r)$, lo que quiere decir que la asignación de vértice es inyectiva, porque la terna (r, p, q) sólo pertenece a un elemento de la partición de los bloques de X.

También hay una segunda consecuencia:

$$(r^i, p^k) \to q^s \in \tilde{\Delta} \text{ si y sólo si } (r^j, p^k) \to q^s$$

y simétricamente

$$(p^k, r^i) \to q^s \in \tilde{\Delta} \text{ si y sólo si } (p^k, r^j) \to q^s \in \tilde{\Delta}$$

para cualesquiera $p^k, q^s \in \tilde{\mathcal{V}}$ y $1 \leq i, j \leq l(r)$, lo que implica que todos los 2-bloques que comienzan en q^s y que difieren por un único símbolo a otro 2-bloque que sí tiene una asignación en el autómata, son asignados al mismo vértice $q^s \in \tilde{\mathcal{V}}$. Las dos observaciones anteriores nos dicen lo siguiente: cuando el conjunto de bloques que comienzan en un mismo símbolo r se separa en $r^1, ..., r^{l(r)}$ vértices de nuestro autómata $\tilde{\mathfrak{A}}$, una transición que entraba a r antes de la separación interior sólo llegará a uno de los estados $r^1, ..., r^{l(r)}$ derivados de la partición de $\Delta(r)$, mientras que una transición que sale de r se ramifica después de la separación interior porque entrará a cada uno de los estados $r^1, ..., r^{l(r)}$.

Una *amalgamación interior de un shift de árbol por vértices* X es un shift de árbol por vértices Y, tal que X es una separación interior de Y. Sea X un shift de árbol por vértices aceptado por el autómata $\mathfrak{A} = (\mathcal{V}, \Delta)$. Vamos a suponer que podemos generar una particion de los vértices $\mathcal{V}$ en subconjuntos denotados por $\mathcal{V}^1, ..., \mathcal{V}^l$ que cumple las siguientes condiciones:

- Siempre que $(p, q) \to r \in \Delta$ y $(p, q) \to s \in \Delta$ con r, s en el mismo conjunto de la partición entonces $r = s$, es decir, cada base de bloque sólo puede tener encima un vértice específico.
- Cuando r, s pertenezcan al mismo elemento de la partición se cumplen las relaciones:

$$\begin{array}{l} (r, p) \to q \in \Delta \text{ si y solo si } (s, p) \to q \in \Delta \\ (p, r) \to q \in \Delta \text{ si y solo si } (p, s) \to q \in \Delta. \end{array} \tag{4.2}$$

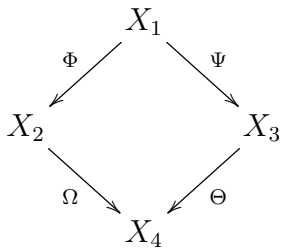
Entonces podemos formar el conjunto $\mathcal{V}' = \{\mathcal{V}^1, ..., \mathcal{V}^l\}$, y definimos la proyección $\pi : \mathcal{V} \to \mathcal{V}'$ que asigna a cada $r \in \mathcal{V}$ aquel conjunto de la partición al que fue asignado. Construimos un nuevo autómata $\mathfrak{A}' = (\mathcal{V}', \Delta')$ donde el conjunto de transiciones está formado por $(\pi(p), \pi(q)) \to \pi(s) \in \Delta'$ siempre que $(p, q) \to s \in \Delta$. Este autómata $\mathfrak{A}$ acepta justamente un shift de árbol por vértices Y que es una amalgamación interior de X. Lo que estamos haciendo es amalgamar todos los vértices que pertenecen a cierto $\mathcal{V}^i$ en uno sólo que llamaremos una *amalgama*. Podríamos encontrar una *amalgamacion* con una única amalgama, que resultaría de una partición en la que todos menos uno de los $\mathcal{V}^i$ son singuletes.

Diremos que dos vértices p, q son *pre-fusionables* si p y q no tienen transiciones entrantes en común, i.e, para cualquier par (r, s) de vértices se cumplen las condiciones descritas en 4.2 arriba del párrafo anterior. Llamaremos la *característica* de un vértice p, denotada por

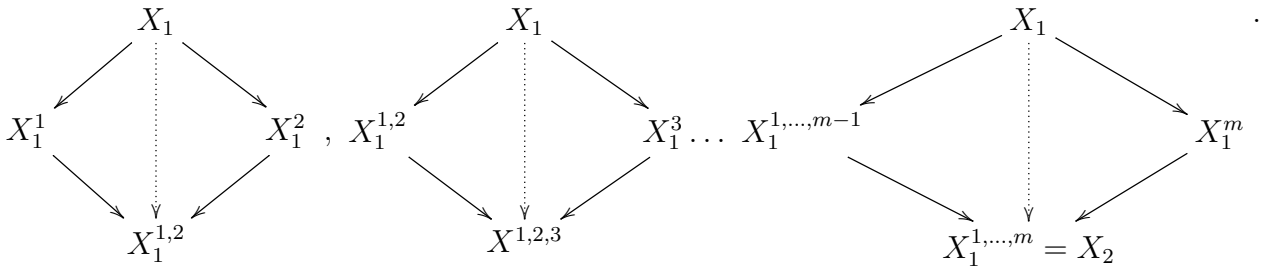
$\kappa(p)$, al conjunto ordenado que tiene una terna $(0, r, s)$ siempre que $(p, r) \to s \in \Delta$ y una terna de la forma $(1, r, s)$ siempre que $(r, p) \to s \in \Delta$. Ordenaremos este conjunto $\kappa(p)$ en órden lexicográfico para cada $p \in \mathcal{V}$. Uno puede fusionar, que significa colocarlos juntos en un conjunto $\mathcal{V}^i$ de la partición de amalgamación, dos vértices p y q si y sólo si son pre-fusionables y además $\kappa(p) = \kappa(q)$.

Ahora podemos pasar a los últimos dos teoremas de este trabajo. Esta última discusión sobre amalgamaciones y separaciones interiores en shifts de árbol por vértices es tremendamente importante.

Proposición 4.5. *Supongamos que X_1 es un shift de árbol por vértices y que X_2 y X_3 son shifts de árbol por vértices obtenidos a partir de X_1 por medio de amalgamaciones. Entonces existe y es posible construir un shift de árbol por vértices X_4, obtenido a partir de ambos X_2 y X_3 por medio de amalgamaciones interiores y tal que hace al siguiente diagrama conmutar:*



Demostración. Primero vamos a hacer una reducción del problema. Lo probaremos para amalgamaciónes interiores que tienen únicamente una amalgama; ésto es suficiente porque si nuestra amalgamación X_2 tiene varias amalgamas, digamos m amalgamas numeradas del 1 hasta n, que están en biyección con los conjuntos $\mathcal{V}^k, 1 \leq k \leq m, k \in K$, donde K es el subconjunto de los indices de la partición que marcan a los conjuntos que no son singuletes, podemos ir encontrando inductivamente el shift de árbol por vértices usando este mismo resultado: primero formamos los shifts por vértices X_1^k que representan el shift por vértices donde únicamente se ha amalgamado el conjunto $\mathcal{V}^k$; luego podemos encontrar la amalgamación de $X_1^{1,\dots,m} = X_2$ siguiendo los diagramas mostrados a continuación:



Supongamos que X_n es el shift de árbol por vértices aceptado por el autómata $\mathfrak{A}_n = (\mathcal{V}_n, \Delta_n)$ para $n = 1, 2, 3$. Asumiremos que existe una amalgamación interior $\Phi : X_1 \to X_2$ y otra $\Psi : X_1 \to X_3$ y, por el comentario anterior, también pensaremos que sólo tienen un vértice amalgamado. Luego podemos asumir que los vértices $p^1, ..., p^{l(p)}$ de $\mathcal{V}_1$ fueron todos amalgamados en el vértice p de $\mathcal{V}_2$. Entonces la transición Δ_1 cumple la primera condición de la definición de pre-fusionable. Supongamos además que los vértices $p^1, ..., p^l(q)$ de $\mathcal{V}_1$ fueron amalgamados en un vértice q de $\mathcal{V}_3$; vamos a dividir la prueba en dos partes.
Pensemos primero que todos los vértices $p^1, ..., p^{l(p)}$ y $q^1, ..., q^{l(q)}$ son distintos; definimos X_4 como la amalgamación interior de X_2 obtenida cuando creamos la amalgama de vértices que tiene a $p, q^1, ..., q^{l(q)}$ en un vértice q; que es la misma amalgama que resultaría de amalgamar $q, p^1, ..., p^{l(p)}$ en un vértice q.
También podría ocurrir que $p^1 = q^1, ..., p^l = q^l$ para algún entero $1 \leq l \leq \text{mín}\,\{l(p), l(q)\}$. Esto implicaría que para cualquier $1 \leq i \leq l(p)$ y $1 \leq j \leq l(q)$ se cumplen todas las condiciones siguientes:

- $(p^i, q^s) \to r \in \Delta_1$ si y sólo si $(p^j, q^s) \to r \in \Delta_1$,
- $(p^i, q^s) \to r \in \Delta_2$ si y sólo si $(p^j, q^s) \to r \in \Delta_2$,
- $(q^s, p^i) \to r \in \Delta_1$ si y sólo si $(q^s, p^j) \to r \in \Delta_1$,
- $(q^s, p^i) \to r \in \Delta_2$ si y sólo si $(q^s, p^j) \to r \in \Delta_2$,

lo cual quiere decir que $p, q^{l+1}, ..., q^{l(q)}$ son pre-fusionables y que tienen la misma característica, asegurándonos que estan en condiciones para poder amalgamarse. Entonces definimos X_4 como la amalgamación interior de X_2 obtenida amalgamando los vértices $p, q^{l+1}, ..., q^{l(q)}$ en el vértice p. Es la misma amalgamación que obtenemos si amalgamamos en X_3 los vértices $q, p^{l+1}, ..., p^{l(p)}$ en un vértice p. Entonces siempre que Φ y Ψ sean amalgamaciones interiores resulta que Ω y Θ son también amalgamaciones interiores. □

Ahora buscaremos hacer la mayor cantidad de amalgamaciones posibles en un shift de árbol cuando amalgamemos, para asegurarnos de no estar definiendo un concepto ambiguo.

Definición 4.19. Sea X un shift de árbol por vértices. Llamaremos la *amalgamación mínima de* X a aquel shift de árbol por vértices definido por el autómata de menor cantidad de vértices posible, que es resultado de amalgamar recursivamente a X siempre que podamos encontrar una amalgamación definida por un autómata de menos vértices.

Corolario 4.1. *Cualquier shift de árbol por vértices tiene una única amalgamación mínima.*

Demostración. Supongamos que X tiene dos amalgamaciones mínimas X_2 y X_3. Entonces por la proposición 4.5 las dos amalgamaciones X_2 y X_3 tienen una amalgamación común Y que también es mínima de donde obtenemos que $Y = X_2 = X_3$. □

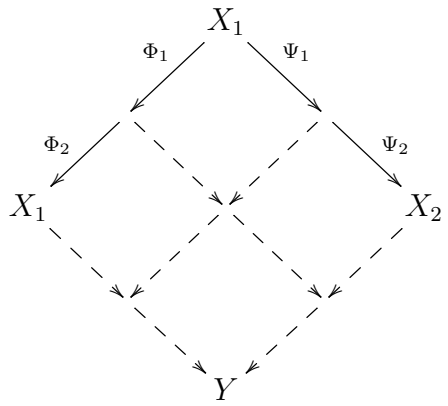
Ahora vamos a escribir un algoritmo para computar la amalgamación mínima de un shift de árbol por vértices. Sea X un shift de árbol por vértices definido por un autómata de n-vértices llamado $\mathfrak{A} = (\mathcal{V}, \Delta)$. Queremos contar cuántas transiciones hay en nuestro autómata (cuántas asignaciones tiene definidas Δ). En el peor caso todas las transiciones son posibles, es decir $(p, q) \to r$ para cualesquiera $p, q, r \in \mathcal{V}$. La combinatoria nos dice que tenemos n posibles opciones para p; n para q y n maneras de escoger r, de donde $|\Delta| \leq n^3$. Utilizando el término algebráico para ésto, decimos que el número de transiciones es, en el peor caso, de orden cúbico en el número de vértices $O(|\mathcal{V}|^3)$. Asumamos que las transiciones de $\mathfrak{A}$ están escritas en una tabla que nos permite saber en tiempo constante si una transición cualquiera $(p, q) \to r$ pertenece a Δ o no. Entonces dado un par de estados (p, q) quisiéramos saber si los estados son pre-fusionables o no. Para ello tomamos uno de los n estados de la tabla (r, s) y verificamos si encontramos alguna transición que entre a p, lo cual toma a lo más n verificaciones. Después verificamos si todos aquellos (r, s) que dieron positivos entran también a q, lo cual toma n verificaciones. De allí que para cada estado, éste proceso toma un tiempo de $O(|\mathcal{V}^2|)$. Tenemos una condición idéntica pero recíproca que intercambia a p y a q, lo cual agrega una cantidad de tiempo igual de $O(|\mathcal{V}^2|)$. Si quisiéramos hacerlo para todo par de vértices podríamos escoger de todos los posibles pares que son $n^2 = \mathcal{V}^2$. Entonces multiplicamos lo que tardamos en verificar un par por la cantidad de pares y obtenemos que el cómputo está acotado por $2n^4$ que es de $O(2|\mathcal{V}^4|)$. Ahora queremos ver cuánto tardamos en calcular la característica de p, para lo cual tenemos dos lugares libres para r y s, ya que formaremos la transición $(p, r) \to s$, verificaremos si está en Δ y formaremos la terna $(0, r, s)$ en caso afirmativo. Esto toma n^2 verificaciones. Hacemos lo mismo para la transición $(r, p) \to s$ que es simétrica y por lo tanto toma n^2. De allí que obtener $\kappa(p)$ es de orden $O(2|\mathcal{V}^2|)$. Si lo hacemos para los n vértices entonces el proceso es de orden $O(2|\mathcal{V}^3|)$. Para poder fusionar dos vértices primero necesitamos la característica de cada uno, lo que nos tomará $O(2|\mathcal{V}^2|) + O(2|\mathcal{V}^2|)$, y saber si son pre-fusionables toma

un tiempo del orden $O(|\mathcal{V}^2|)$; entonces encontrar y fusionar dos vértices puede ser calculado en un tiempo de orden $O(5|\mathcal{V}|^2)$; si lo hacemos para todos los vértices entonces nos toma $O|5|\mathcal{V}^3||$, que es precisamente la complejidad computacional del cálculo de la amalgamación mínima.

Por fin tenemos todo lo necesario para probar nuestro último teorema del trabajo:

Teorema 4.2. *Sean X_1 y X_2 dos shifts de tipo finito TSFT. Es decidible si X_1 y X_2 son conjugados.*

Demostración. Vamos a asumir que X_1 y X_2 son conjugados. Gracias a la proposicion 4,3 podemos asumir además que X_1 y X_2 son shifts de árbol por vértices. Por el teorema de descomposición 4.1, hay un espacio shift X y una sucesión de separaciones interiores desde X_1 hasta X, inducidos por códigos de separación cuyas inversas son códigos de amalgamación, seguidas de un renombramiento de X y de una sucesión de amalgamaciones desde X hasta X_2. Como X_1 y X_2 son shifts de árbol por vértices podemos asumir que dicho X es también un shift de árbol por vértices. Ilustramos este proceso en la figura siguiente:



Usando la proposición 4.5 podemos construir, para cada punto de encuentro de dos flechas punteadas, un shift de árbol por vértices que es amalgamación común de los dos espacios donde comienzan las flechas. Como consecuencia de ésto, X_1 y X_2 tienen una amalgamación en común y, por lo tanto, la misma amalgamación mínima Y. De allí que nuestro proceso de decisión consiste simplemente en computar ambas amalgamaciones mínimas de los espacios shift X_1 y X_2, que nos tomará un tiempo de orden $O(10|\mathcal{V}|^3)$. Siempre que encontremos la misma amalgamación mínima para ambos espacios (módulo un renombramiento) podemos asegurar que son conjugados. Además, siempre que dos espacios tienen la misma amalgamación mínima son conjugados, pues tenemos códigos de amalgamación

$\Psi_i, 1 \leq i \leq n$ que nos llevan desde X_1 hasta la amalgamación mínima Y y tenemos otra sucesion $\Phi_i, 1 \leq i \leq m$ de códigos de amalgamación que nos llevan desde X_2 hasta Y, por lo que podemos construir la conjugación $\Lambda : X_1 \to X_2$, tal que $\Lambda = \Psi_1 \circ \Psi_2 \circ ... \circ \Psi_n \circ \Phi_1^{-1} \circ ... \circ \Phi_m^{-1}$. Por lo que hemos demostrado, X_1 y X_2 son conjugados únicamente si y sólo si, tras calcular la amalgamación mínima de ambos, obtenemos el mismo espacio shift Y módulo un renombramiento. □

Un último detalle importante que queremos resaltar es el siguiente: el tiempo que toma la verificación completa sobre la conjugación de dos espacios shift no es polinomial. Ya mencionamos en el teorema 4.4 que obtener la amalgamación mínima de ambos sí es de orden polinomial, pero en caso de tener que encontrar un renombramiento tenemos que implementar el algoritmo de complejidad exponencial siguiente:
Digamos que ya encontramos las amalgamaciones mínimas de los shifts por vértices X_i de alfabeto $\mathcal{A}_i$ reconocido por $\mathfrak{A}_i = (\mathcal{A}_i = \mathcal{V}_i, \Delta_i)$, que llamaremos Y_i, cuyo autómata reconocedor es $\mathfrak{B} = (\mathcal{V}_i^Y, \Delta_i^Y)$ con $i = 1, 2$. Sabemos que $\mathcal{V}_i^Y \subseteq \mathcal{A}_i$, pero en el peor de los casos pueden ser iguales. Ahora ¿cómo hacer para saber si Y_2 es un renombramiento de Y_1?. Sabemos que la cantidad de transiciones en Δ_i es, a lo más, todos los bloques de tamaño 2 que podemos formar sobre el alfabeto $\mathcal{A}_i$, por lo que $|\Delta_i^Y| \leq |\Delta_i| \leq |\mathcal{A}|^3$, con la primera desigualdad derivada de que cuando amalgamamos un espacio le quitamos transiciones. Teniendo todas las transiciones en Δ_1^Y, nuestro algoritmo cambia cada una de las apariciones de un vértice $p \in \mathcal{V}_1^Y$ en cada transición por uno $p' \in \mathcal{V}_2^Y$ y evalúa si el conjunto de transiciones resultante es justamente Δ_2^Y; ésto tiene una cota mínima de $|\mathcal{V}_1^Y|^{|\mathcal{V}_2^Y|}$, lo cual tiene un orden $O(|\mathcal{V}^{|\mathcal{V}|}|)$, que es exponencial sobre el número de vértices.

Por último, vamos a dar un ejemplo donde verificamos la conjugación de dos espacios shift de árbol por vértices.

Ejemplo 4.5. Sean X_1 y X_2 dos shift de árbol por vértices sobre el alfabeto $\mathcal{V} = \{a, b, c\}$. El espacio X_1 es aceptado por el autómata $\mathfrak{A}_1 = (\mathcal{V}, \Delta_1)$ y el espacio X_2 por el autómata $\mathfrak{A}_2 = (\mathcal{V}, \Delta_2)$, donde las transiciones están dadas en las tablas abajo:

$\Delta_1 =$

	a	b	c
a	a	c	c
b	b	a	a
c	b	a	a

$\Delta_2 =$

	a	b	c
a	c	a	a
b	a	b	b
c	a	b	b

$\Delta_3 =$

	a	b
a	a	b
b	b	a

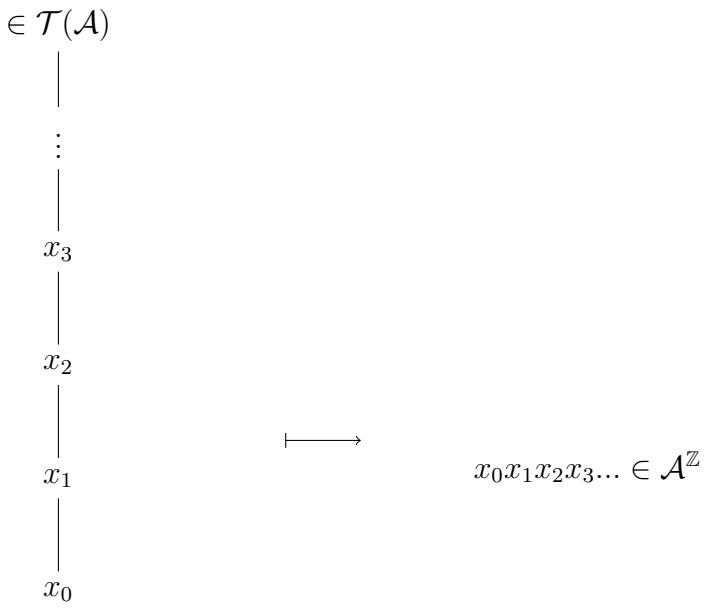
$\Delta_4 =$

	a	b
b	a	b

Podemos ahora ver que no es muy complejo saber si dos vértices tienen la misma característica, pues lo único que necesitamos es fijarnos en que sus filas y columnas en la tabla sean iguales, como podemos observar en la tabla de Δ_1 en la que las filas y columnas que corresponden a b y c son iguales, de donde sabemos que $\kappa b = \kappa c$. También es sencillo ver que son pre-fusionables, pues no hay estado que entre a ambos, ya que en ninguna fila ni columna aparecen juntos.

Así pues, amalgamamos a nuestro X_1 en un X_3 reconocido por el autómata $\mathfrak{A}_3 = (\mathcal{V}, \Delta_3)$, como nos explicó el teorema 4.5, y hacemos lo mismo amalgamando X_2 en X_4, reconocido por $\mathfrak{A}_4 = (\mathcal{V}, \Delta_4)$. Como ya no es posible amalgamar más, ambos X_3 y X_4 son amalgamaciones mínimas. Podemos ver que, permutando los dos símbolos del alfabeto que aparecen en X_3, obtenemos a X_4 lo que significa que X_1 y X_2 son conjugados.

La gran razon por la que hemos trabajado en la primera parte presentando conceptos de la teoría de espacios shift bidireccionales es porque, para los shifts sobre $\mathbb{N}$, la teoría de los espacios de shift de árbol es una extensión. Podemos convencernos fácilmente de que es idéntica la presentación de un espacio shift de sucesiones unidireccionales y la de árboles 1-arios. Por lo tanto, todos los teoremas y conceptos que hemos desarrollado en el último capítulo son válidos en una teoría de espacios shift con sucesiones unidireccionales. Por lo tanto, también hemos probado la decidibilidad de la conjugación para ellos. Vemos en la figura como los puntos de los dos espacios son isomorfos bajo el código de uno bloques $t(0^n) \mapsto x_n$



Equivalencia entre puntos del shift de árboles unarios completo y el de sucesiones sobre $\mathbb{N}$.

Conclusión

Hemos logrado la meta de probar el teorema de descomposición para los TSFTs y su decidibilidad. Como vimos en el último párrafo del trabajo, tenemos como corolario ambos resultados para sucesiones uno-direccionales y, por lo tanto, una prueba paralela a la que se da en [7], aunque posiblemente un poco más complicada. El mismo problema para los espacios shift de sucesiones sobre $\mathbb{Z}$ es todavía un problema abierto, pues en este caso nos hemos aprovechado del buen orden de los números naturales para encontrar la amalgamación mínima. En el tercer capítulo de [1] se presenta la cubierta de fischer para espacios sobre $\mathbb{Z}$ y es un paralelismo en la teoría a los autómatas y amalgamaciones mínimas que hemos construído nosotros. Podríamos considerar a las presentaciones derechas resolventes definidas en [9],[10] como el análogo a las amalgamaciones mínimas; sin embargo, éste camino no ha dado suficientes respuestas hasta ahora, como se puede constatar en el compendio de problemas que publicó en 2010 el gurú de la dinámica simbólica Mike Boyle en [15]. En el desarrollo de los $\mathbb{Z}^d$ shifts se ha avanzado mucho dentro de la categoría de los que son "mixing", pero todavía quedan muchas brechas por explorar. Como en este caso hemos pasado al punto de vista de autómatas, probablemente el análisis de la conjugación en dimensiones mayores también siga este camino aunque, por ahora, todavía se trabaja en su definición y en otros conceptos que no surgían de manera tan natural en $\mathbb{Z}$, como la expansividad y las acciones de grupo como puede verse en [12]. Los autómatas parecen ser una teoría puente entre todo el trabajo une el área de la dinámica simbólica con la teoría de la computación y la de los lenguajes regulares. El libro [5] tiene muchas pruebas que resultan muy útiles cuando se está analizando la decidibilidad de propiedades de espacios shift y me han parecido tremendamente divertidos. Sin duda, de la mano de la teoría de la computación, la dinámica simbólica se convierte en un área con mucho potencial y tremenda importancia dentro de las matemáticas y estoy muy contento de haber explorado un pequeño planeta dentro de su vasto universo.

Bibliografía

[1] D. Lind and B. Marcus. An introduction to symbolic dynamics and coding. Cambridge University Press 1995.

[2] N. Aubrun and M.P Béal Tree-shifts of finite type Theoretical Computer Science 459 (2012) 16-25

[3] Kitchens, Bruce. Symbolic Dynamics: One Sided, Two-sided and Countable State Markov shifts. Springer, 1998.

[4] Hochman, Michael. On the dynamics and recursive properties of multidimensional symbolic systems. Inventiones mathematicae 176.1 (2009): 131-167.

[5] Hopcroft, John E., Rajeev Motwani, and Jeffrey D. Ullman. Introduction to Automata Theory, Languages, and Computation. 2nd edition (2001): 96.

[6] Baker, Kirby A. Strong shift equivalence and shear adjacency of nonnegative square integer matrices. Linear Algebra and its Applications 93 (1987): 131-147.

[7] Kim, Ki Hang, and Fred W. Roush. Decidability of shift equivalence. Dynamical systems. Springer Berlin Heidelberg, 1988. 374-424.

[8] M.-P. Beal. Codage Symbolique. Masson, Paris, 1993

[9] Fischer, Roland. Sofic systems and graphs. Monatshefte für Mathematik 80.3 (1975): 179-186.

[10] Fischer, Roland. Graphs and symbolic dynamics. Colloq. Math. Soc. János Bólyai: Topics in Information Theory. Vol. 16. 1975.

[11] Prieto de Castro, Carlos Topología básica. Carlos Prieto—México. FCE, 2003 519 pp.

[12] Boyle, Mike, Ronnie Pavlov, and Michael Schraudner. Multidimensional sofic shifts without separation and their factors. Transactions of the American Mathematical Society 362.9 (2010): 4617-4653.

[13] Bravo, Alejandro, H. Rincón, and C. Rincón. Álgebra superior. Las prensas de ciencias (2006).

[14] Williams, R. F. Classification of subshifts of finite type. Recent Advances in Topological Dynamics. Springer Berlin Heidelberg, 1973. 281-285.

[15] Boyle, Mike. Open problems in symbolic dynamics. Contemporary Mathematics 469 (2008): 69-118.

[16] Boyle Mike and Alejandro Maass. Expansive invertible onesided cellular automata. Journal of the Mathematical Society of Japan 52.4 (2000): 725-740.

[17] Nasu, Masakazu. Textile systems for endomorphisms and automorphisms of the shift. Vol. 114. No. 546. American Mathematical Society, 1995.

[18] Wagoner, J. B. Strong shift equivalence theory and the shift equivalence problem. Bulletin AMS 36 (1999): 271-296.