Introducción a la Lógica y la Computación

Parte III: Lenguajes y Autómatas

Autores: Alejandro Tiraboschi y colaboradores

Contenidos

1 Introducción

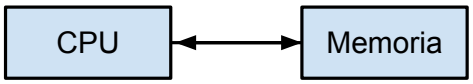
En esta parte de la materia vamos a empezar a estudiar *modelos de computación*. Es decir, modelos teóricos que aproximan el comportamiento de una computadora. Este tema se verá en más detalle en otras materias más avanzadas, cuando se estudien las *Máquinas de Turing* que nos servirán para definir exactamente que significa *computar* y cuales son sus límites.

En esta materia estudiaremos un tipo de máquinas más simples que las máquinas de Turing que se llaman *autómatas*, y veremos que existe una relación muy fuerte entre modelos de computación y lenguajes formales.

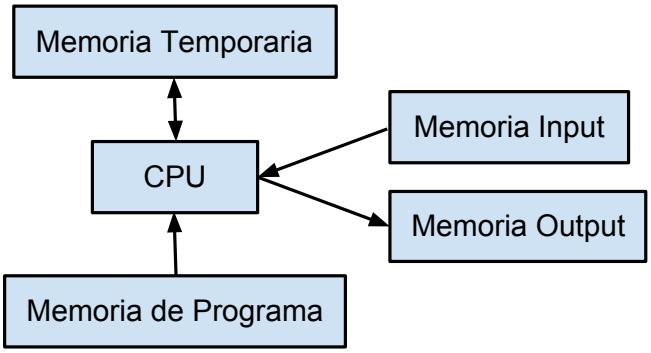
Una computadora que resuelve un problema mediante algún algoritmo puede verse en forma abstracta como "hablando el lenguaje" $\{(\text{input}_1,\text{output}_1), \ldots, (\text{input}_n,\text{output}_n), \ldots\}$. Notar que podemos pensar que este lenguaje es un conjunto de cadenas $\{\text{input}_1\#\text{output}_1, \ldots, \text{input}_n\#\text{output}_n, \ldots\}$, donde

es algún símbolo que no apareza en los inputs o los outputs. Por ejemplo la funcion $f(x) = x^3$ puede asociarse con el conjunto de cadenas {0#0, 1#1, 2#8, 3#9, ... }. Es por esto que una forma de estudiar las cosas que una computadora puede realmente computar, es investigando que lenguajes que distintos tipos de máquinas pueden producir.

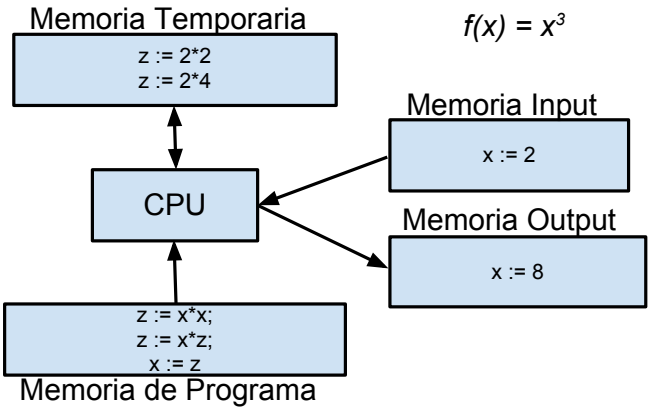
Miremos un poco mas en detalle a que nos referimos cuando decimos 'máquinas'. De forma muy abstracta, podemos representar una computadora simplemente con el siguiente diagrama:



donde sólo diferenciamos una unidad con capacidad de procesamiento (el CPU) de una unidad con capacidad de almacenamiento (la memoria). Nos va a interesar particularmente distinguir distintos tipos de memoria:



No nos interesa, en este momento, definir más precisamente cuales son las diferencias entre los distintos tipos de memoria, y nos alcanza con el siguiente ejemplo:



Ahora sí, vamos a considerar que las 'máquinas' que vamos a estudiar siempre tienen memoria input, memoria output y memoria de programa (esto simplemente quiere decir que toman un input, producen un output y siguen algún tipo de programa), pero difieren en el tipo de *memoria temporaria* que pueden tener:

- **Autómatas de Estados Finitos** (Finite State Automata): no tienen ningún tipo de memoria temporaria.

- **Autómatas con Pila** (Pushdown Automata): tienen una pila como memoria temporaria.
- **Máquinas de Turing** (Turing Machines): tienen memoria de lectura y escritura de acceso aleatoria.

Cada tipo de máquina tiene distinto poder de cómputo, que se va incrementando a medida que el tipo de memoria es menos restrictivo. Como dijimos al principio, en esta materia no llegaremos a estudiar las Máquinas de Turing, pero sí veremos distinto stipos de Autómatas de Estados Finitos y con Pila y estudiaremos sus propiedades.

2 Autómatas finitos y expresiones regulares

2.1 Cadenas, alfabetos y lenguajes

Un “símbolo" es una entidad abstracta que no definiremos formalmente, así como en geometría no definimos los “puntos" y las “líneas". Letras y dígitos son ejemplos de símbolos frecuentemente usados. Una *cadena* o *palabra* es una secuencia finita de símbolos yuxtapuestos. Por ejemplo a, b y c son símbolos y *abbba*, *abcc* y *bcccc* son cadenas. La *longitud* de una cadena w, que denotaremos $\mid w \mid$, es el número de símbolos que componen la cadena. Por ejemplo la cadena *aabc* tiene longitud 4, o en símbolos, $\mid w \mid = 4$. La *cadena vacía*, que denotaremos ε, es la cadena que tiene cero símbolos, luego $\mid \varepsilon \mid = 0$. Es importante notar que, por ejemplo, la cadena *aba* es igual a la cadena *ab*ε*a* o $\varepsilon\varepsilon$*aba* pues ε es la cadena vacía y no es símbolo.

Una *subcadena* de una cadena es una secuencia de símbolos incluida en la cadena, por ejemplo *ab* y ε son subcadenas de *cccabccc*, mientras que *cb* no lo es. Un *prefijo* de una cadena w, es una subcadena de w que comienza con el primer símbolo de w. Un *sufijo* de una cadena w, es una subcadena de w que termina con el último símbolo de w. Por ejemplo la cadena *abc* tiene como prefijos a las cadenas ε, a, *ab* y *abc*, y como sufijos a ε, c, *bc* y *abc*.

La *concatenación* de dos cadenas w y x, que se denota wx, es la cadena formada escribiendo la primera cadena seguida de la segunda, sin espacios en el medio. Por ejemplo, la concatenación de *aaa* y *bca* es *aaabca*. Observemos que la cadena vacía concatenada con otra cadena no la modifica, es decir, $\varepsilon w = w\varepsilon = w$.

Un *alfabeto* es un conjunto finito de símbolos. Un *lenguaje (formal)* es un conjunto de cadenas de símbolos de un alfabeto dado. El conjunto vacío, $\emptyset$, y el conjunto $\{\varepsilon\}$, es decir el conjunto que sólo tiene la cadena vacía, son lenguajes. Observar que estos dos lenguajes son distintos, el primero no tiene elementos, mientras que el segundo tiene un elemento.

Ejemplo 2.1. El conjunto de *palindromes* o *capicúas* sobre el alfabeto $\{0,1\}$ es un lenguaje infinito. Algunos elementos de este lenguaje son ε, 0, 1, 00, 11, 000, 010, 101, 111.

Dado un alfabeto Σ, denotaremos con Σ^* al lenguaje formado por todas las cadenas sobre el alfabeto.

Ejemplo 2.2. Sea el alfabeto $\Sigma = \{a\}$, entonces $\Sigma^* = \{\varepsilon, a, aa, aaa, aaaa, \ldots\}$. Si $\Sigma = \{0,1\}$, entonces $\Sigma^* = \{\varepsilon, 0, 1, 00, 01, 10, 11, 000, \ldots\}$

2.2 Sistemas de estados finitos

Los autómatas finitos son modelos matemáticos de sistemas, con entradas y salidas discretas. El sistema puede estar en una, entre un número finito, de configuraciones internas o “estados". El estado

del sistema es la única información que se debe tomar en cuenta para pasar a otro estado con una entrada o input dado. Un buen ejemplo, que desarrollaremos más adelante, es el mecanismo de control de un ascensor. El mecanismo no recuerda los pedidos previos, si no solamente el piso en que está el ascensor, la dirección de movimiento (arriba, abajo o quieto) y la colección de pedidos no satisfechos.

En ciencias de la computación es posible encontrar numerosos ejemplos de sistemas de estados finitos, y la teoría de autómatas es una herramienta útil para dichos sistemas. Ciertos programas tales como editores de texto y analizadores léxicos, encontrados en la mayoría de los compiladores, son a menudo designados como sistemas de estados finitos. Por ejemplo, un analizador léxico recorre los símbolos de un programa de computación para localizar las cadenas de caracteres correspondientes a identificadores, constantes numéricas, palabras reservadas, etc.

Antes de definir formalmente un sistema de estados finito consideremos el siguiente ejemplo.

Ejemplo 2.3. Un hombre con un lobo, una cabra y un repollo está en la ribera izquierda de un río. Hay un bote que puede transportar solamente al hombre y a alguno de los otros tres. El hombre quiere transportar al lobo, la cabra y el repollo al otro lado del río, sin embargo si deja solos al lobo y la cabra, el lobo devorará la cabra, y si deja solos a la cabra y el repollo, la cabra devorará al repollo. >Es posible cruzar el río sin que el lobo coma a la cabra y sin que la cabra coma el repollo?

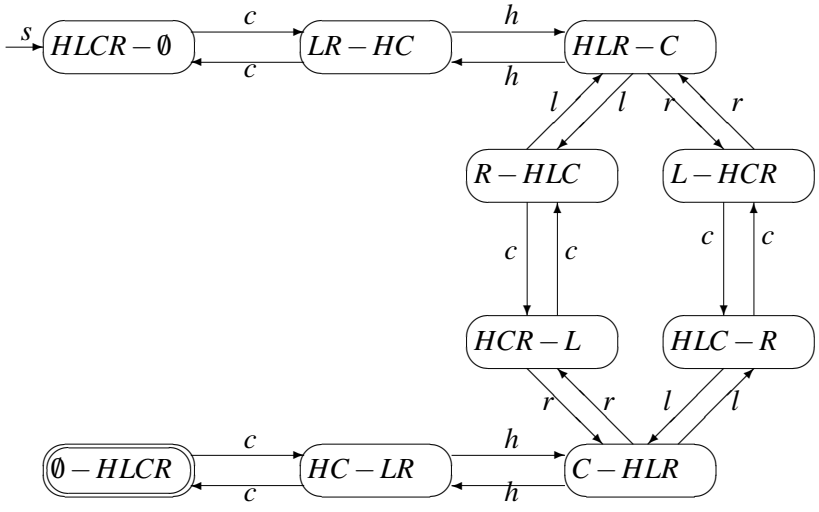


Figura 1: Diagrama de transición para el problema del hombre, el lobo, la cabra y el repollo.

El problema se modela observando que la información pertinente es la de quien ocupa cada ribera del río después de un cruce, esos serán los posibles estados, 16 en total. Un estado se denotará, por ejemplo, $LR-HC$ que significará que en la ribera izquierda están el lobo y el repollo y en la ribera derecha están el hombre y la cabra. Algunos de los 16 estados posibles, tal como $HL-CR$, son fatales y por lo tanto nunca debemos “entrar" a ellos.

Los “inputs" del sistema son las acciones que el hombre toma. Él puede cruzar sólo (input h), con el lobo (input l), con la cabra (input c) o con el repollo (input r). El estado inicial es $HLCR-\emptyset$ y el final es $\emptyset-HLCR$. El diagrama de transición (de todos los estados posibles no fatales y sus inputs) se muestra en la Fig. 1.

Hay dos soluciones cortas a este problema, tal como se deduce del gráfico: son caminos que salen del estado inicial (el círculo marcado con una flecha) y llegan al estado final (en doble círculo). Hay también un número infinito de soluciones, que, excepto las dos cortas tienen ciclos inútiles. Dado un sistema de estados finitos podemos definir el siguiente lenguaje: el conjunto de todas las cadenas de inputs que llevan del estado inicial al estado final. Por ejemplo *chlcrhc*, *chrclhc*, *chhhlcrhc* son cadenas de este lenguaje (las dos primeras corresponden a las soluciones cortas).

Finalmente debemos observar que este ejemplo tiene dos características particulares que en general los sistemas de estados finitos no tienen. La primera es que hay un solo estado final, cosa que en general no ocurre y la segunda es que por cada transición hay una transición inversa.

2.3 Autómatas finitos

Definición 2.1. Un *autómata finito determinístico (DFA)* es una 5-upla $(Q, \Sigma, \delta, q_0, F)$ que consiste de

1. un conjunto finito $Q = \{q_1, q_2, \ldots, q_m\}$ de *estados*,
2. un conjunto finito $\Sigma = \{a_1, a_2, \ldots, a_n\}$ de *símbolos de entrada* o *input*,
3. un conjunto de *reglas de transición* que transforma un estado con un input dado en otro estado, independientemente de lo que haya sucedido en las lecturas anteriores. Formalmente las reglas de transición se definen con una función $\delta : Q \times \Sigma \to Q$.
4. Además habrá un *estado inicial* q_0 y un conjunto de estados F, que llamaremos *estados finales*.

Muchas veces representaremos las reglas de transición por una tabla a doble entrada: buscando el cruce entre la entrada de un estado y la entrada de un input sabemos como cambia el estado.

Ejemplo 2.4. Sea M un autómata finito determinístico con estados $\{q_0, q_1, q_2\}$, símbolos de entrada $\Sigma = \{a, b\}$ y las siguientes reglas de transición: el estado q_0 se transforma en q_1 si el input es a y en q_0 si el input es b; el estado q_1 se transforma en q_1 si el input es a y en q_2 si el input es b; finalmente, el estado q_2 se transforma en q_2 si el input es a y en q_0 si el input es b. Es decir $\delta : Q \times \Sigma \to Q$ está definida por:

$$\begin{array}{llll} \delta(q_0, a) & = q_1, & \delta(q_0, b) & = q_0 \\ \delta(q_1, a) & = q_1, & \delta(q_1, b) & = q_2 \\ \delta(q_2, a) & = q_2, & \delta(q_2, b) & = q_0 \end{array}$$

que puede ser representada (en forma más compacta) con la siguiente tabla.

	a	b
q_0	q_1	q_0
q_1	q_1	q_2
q_2	q_2	q_0

Hablando en forma estricta el Ejemplo 2.3 no es un autómata en el sentido de la Definición 2.1 pues hay estados que no aceptan ciertos inputs. Sin embargo, en la sección siguiente extenderemos el concepto de autómata, con lo cual este tipo de situaciones será admisible.

Ejemplo 2.5. En el Ejemplo 2.3 los estados son:

$$HCLR-,\ LR-HC,\ etc.$$

el estado inicial es $HCLR-$ y el estado final es $-HCLR$. los símbolos de input son h, c, l, r y las reglas de transición son: $HCLR-$ con input c va a $LR-HC$, etc.

Como se observa en los Ejemplos 2.3 y 2.5 hay una relación estrecha entre un autómata finito determinístico y “grafos". A cada autómata finito determinístico se le puede asignar un dibujo, que llamaremos grafo, de la siguiente manera: a cada estado le corresponde un círculo con el nombre del estado dentro, el estado inicial está distinguido con una flecha apuntando al círculo y los estados finales se dibujan con doble círculo. En lo sucesivo llamaremos a estos círculos y doble círculos *nodos*. Si hay una transición del estado q al estado p con input a, entonces se dibuja una *flecha* que va del nodo correspondiente a q al nodo correspondiente a p, además a esta flecha le ponemos la etiqueta a. A este tipo de grafo lo llamaremos un *diagrama de transición*.

Es claro además, que si tenemos un diagrama de transición hay un autómata finito asociado a él; y por lo tanto en el futuro no distinguiremos autómatas finitos y diagramas de transición.

Ejemplo 2.6. Sea M un autómata con estados $\{q_0, q_1, q_2, q_3\}$, símbolos de input $\{0, 1\}$, estado inicial q_0 y estado final q_0 también. Las reglas de transición están dadas por la Tabla 1.

	0	1
q_0	q_2	q_1
q_1	q_3	q_0
q_2	q_0	q_3
q_3	q_1	q_2

Tabla 1:

El diagrama de transición de M será el dado por la Fig. 2

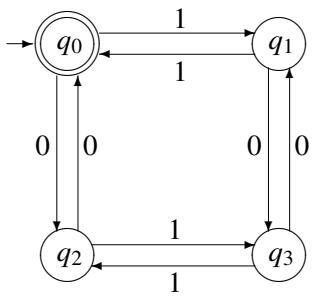


Figura 2:

Ejemplo 2.7. El siguiente autómata sirve para modelar el mecanismo de control de un ascensor automático, en este caso de un edificio con pisos 0, 1 y 2.

Un estado debe describir el piso en que se encuentra el ascensor (piso 0, 1 o 2) , si está detenido (p) o va hacia arriba (u) o abajo (d), y las demandas no satisfechas. Por ejemplo podríamos representar el estado: el ascensor está en el piso 1, va hacia arriba y debe ir al piso 0, por la upla $(1, u, V, F, F)$, las últimas tres componentes de la upla denotan los tres pisos posibles, una V denota que debe ir a ese piso, mientras que F denota que no debe ir. Los posibles inputs son los llamados que hace la gente a ciertos pisos (no es necesario distinguir si el llamado se hace desde dentro del ascensor o desde un piso) por lo tanto los inputs posibles son 0, 1 y 2. Además hay otro input que no viene dado por los usuarios si no por el movimiento del ascensor: cuando, por ejemplo, el ascensor tiene el piso 1 como

demanda insatisfecha y llega a este piso, entonces la demanda deja de ser insatisfecha. Denotemos este input con m. Notemos que hay ciertos estados que no son posibles, como por ejemplo $(1,u,F,V,F)$, pues si estamos en cierto piso no tiene sentido pedir que vaya al mismo piso. Tampoco tienen sentido, por ejemplo, $(1,u,F,F,F)$ o $(1,u,V,F,F)$, pues si no hay demandas insatisfechas arriba del piso en que está el ascensor, este no puede ir hacia arriba. Otro caso no admisible es un estado en que el ascensor está detenido y hay demandas insatisfechas, pues si el ascensor está detenido y entra una demanda, el ascensor comienza a moverse en el sentido de la demanda. Los estados finales serán aquellos en que el ascensor está detenido. Consideremos que el estado inicial es $(0,p,F,F,F)$. La Tabla 2 da los posibles cambios de estados.

	0	1	2	m
$(0,p,F,F,F)$	$(0,p,F,F,F)$	$(0,u,F,V,F)$	$(0,u,F,F,V)$	$(0,p,F,F,F)$
$(0,u,F,V,F)$	$(0,u,F,V,F)$	$(0,u,F,V,F)$	$(0,u,F,V,V)$	$(1,p,F,F,F)$
$(0,u,F,F,V)$	$(0,u,F,F,V)$	$(0,u,F,V,V)$	$(0,u,F,F,V)$	$(1,u,F,F,V)$
$(0,u,F,V,V)$	$(0,u,F,V,V)$	$(0,u,F,V,V)$	$(0,u,F,V,V)$	$(1,u,F,F,V)$
$(1,p,F,F,F)$	$(1,d,V,F,F)$	$(1,p,F,F,F)$	$(1,u,F,F,V)$	$(1,p,F,F,F)$
$(1,u,F,F,V)$	$(1,u,V,F,V)$	$(1,u,F,F,V)$	$(1,u,F,F,V)$	$(2,p,F,F,F)$
$(1,d,V,F,F)$	$(1,d,V,F,F)$	$(1,d,V,F,F)$	$(1,d,V,F,V)$	$(0,p,F,F,F)$
$(1,u,V,F,V)$	$(1,u,V,F,V)$	$(1,u,V,F,V)$	$(1,u,V,F,V)$	$(2,d,V,F,F)$
$(1,d,V,F,V)$	$(1,d,V,F,V)$	$(1,d,V,F,V)$	$(1,d,V,F,V)$	$(0,u,F,F,V)$
$(2,p,F,F,F)$	$(2,d,V,F,F)$	$(2,d,F,V,F)$	$(2,p,F,F,F)$	$(2,p,F,F,F)$
$(2,d,V,F,F)$	$(2,d,V,F,F)$	$(2,d,V,V,F)$	$(2,d,V,F,F)$	$(1,d,V,F,F)$
$(2,d,F,V,F)$	$(2,d,V,V,F)$	$(2,d,F,V,F)$	$(2,d,F,V,F)$	$(1,p,F,F,F)$
$(2,d,V,V,F)$	$(2,d,V,V,F)$	$(2,d,V,V,F)$	$(2,d,V,V,F)$	$(1,d,V,F,F)$

Tabla 2:

Nos interesa ahora estudiar las sucesiones de símbolos de input de un autómata dado que nos lleven del estado inicial a uno final. Más precisamente

Definición 2.2. Sea $M=(Q,\Sigma,\delta,q_0,F)$ un autómata finito determinístico. Una *cadena en M* es un elemento de Σ^*. Sean p y q estados de M y $\alpha=a_0a_1\dots a_n$ una cadena en M, diremos que α *transforma* q en p (un estado de M) si partiendo del estado q y aplicando sucesivamente los inputs a_0, a_1, hasta a_n, obtenemos el estado p. También definiremos que ε transforma un estado en si mismo. Finalmente, diremos que una cadena α *es aceptada por M*, si α transforma el estado inicial en uno final.

A nivel de diagramas de transición, podemos mirar el diagrama como un conjunto de sitios o ciudades (los nodos) con rutas que conectan a algunos de ellos (las flechas), entonces una cadena $\alpha=a_0a_1\dots a_n$, que también llamaremos *recorrido*, es aceptada si partiendo del sitio q_0 y yendo por las rutas $a_0,a_1,\dots,a_n$ llegamos a un sitio final.

Observación 2.1. Denotaremos el hecho de que un símbolo de input a transforma un estado q en p, por

$$q \xrightarrow{a} p.$$

Sea $\alpha=a_0a_1\dots a_n$ una cadena que transforma un estado q en p, entonces se denota

$$q \xrightarrow{a_0} q_1 \xrightarrow{a_1} \cdots \xrightarrow{a_{n-1}} q_{n-1} \xrightarrow{a_n} p \quad \text{o bien} \quad q \xrightarrow{\alpha}$$

Observemos que $q \xrightarrow{\varepsilon} q$, para todo estado q.

Notemos que esta notación tiene la siguiente propiedad : si $\alpha = \beta\gamma$, donde α, β y γ son cadenas, entonces

$$q \xrightarrow{\alpha} p \text{ si y sólo si existe } r \in Q \text{ tal que } q \xrightarrow{\beta} r \xrightarrow{\gamma} p. \tag{1}$$

Definición 2.3. Sea M un autómata finito determinístico, entonces el *lenguaje aceptado por M*, denotado $L(M)$, es el conjunto de cadenas aceptadas por el autómata. Si M y M' son dos autómatas finitos diremos que son *equivalentes* si $L(M) = L(M')$.

En la notación explicada más arriba si $M = (Q, \Sigma, \delta, q_0, F)$ es un autómata finito determinístico, entonces

$$L(M) = \{\alpha \in \Sigma^* \mid \text{ existe } p \in F \text{ tal que } q_0 \xrightarrow{\alpha} p\}.$$

Ejemplo 2.8. El lenguaje aceptado por el autómata del Ejemplo 2.6 es $L(M) =$ el conjunto de cadenas que tienen un número par de 0's y un número par de 1's.

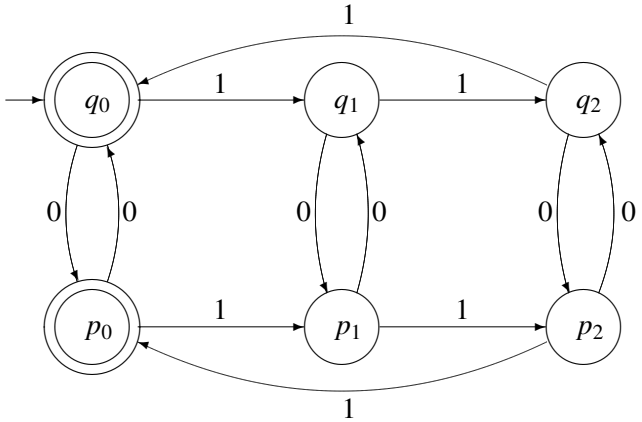
Demostración. Si observamos el diagrama de transición correspondiente notamos que ir de izquierda a derecha o de derecha a izquierda significa aplicar el input 1. Ir de arriba a abajo o de abajo a arriba significa aplicar el input 0. Ahora bien, como q_0 es el estado inicial y final, una palabra aceptada va "recorriendo" el diagrama de tal forma que cuando termina subió tantas veces como bajó y fue a la izquierda tantas veces como fue a la derecha. Es decir que esta palabra tiene un número par de 0's y un número par de 1's.

Por otro lado, si una palabra tiene un número par de 0's y un número par de 1's es claro, por las mismas razones expuestas arriba, que termina su recorrido en q_0.

Observemos que ε es una cadena aceptada, debido a que q_0 es estado final y que $q_0 \xrightarrow{\varepsilon} q_0$. Claramente, la cadena ε tiene un número par de ceros y unos, pues tiene 0 ceros y 0 unos.

□

Ejemplo 2.9. El lenguaje reconocido por el siguiente autómata es el de las cadenas de 0's y 1's con tres o un múltiplo de tres 1's. Por ejemplo las cadenas 00, 01011, 110011011, pertenecen a este lenguaje.



Demostración. Como en el anterior ejemplo la idea es geométrica: observemos que para que una cadena termine en q_0 o p_0, debe haber ido a un estado con subíndice 1, luego a otro con subíndice 2 y luego volver a un estado con subíndice 0, y esto se puede hacer un número arbitrario de veces. Ahora

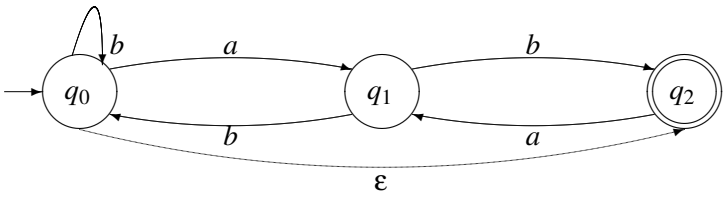
bien, cada vez que se hace esto estamos usando tres 1's de la cadena. Además, el número de ceros es arbitrario. Luego una cadena aceptada por el autómata tiene la propiedad requerida. La recíproca es similar. □

2.4 Autómatas no determinísticos

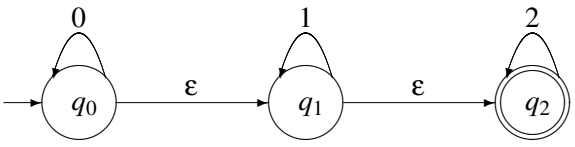
Ahora introduciremos la noción de autómata finito no determinístico. No es difícil ver que un conjunto aceptado por un autómata no determinístico también es aceptado por uno determinístico y viceversa. Sin embargo, el autómata finito no determinístico resultará útil para probar ciertos resultados y para ciertas aplicaciones. Daremos una definición no formal de este tipo de autómatas. El lector interesado en la definición formal puede consultar la subsección 2.4.1.

Modifiquemos la definición de autómata finito permitiendo cero, una, o más transiciones de un estado con el mismo símbolo de input. Es decir de un estado pueden partir un número arbitrario de flechas cada una etiquetada con cualquier símbolo de input. También permitamos transiciones sin inputs. Esta nueva definición nos da los *autómatas finitos no determinísticos (NFA) con* ε*-movimientos*. A nivel de diagramas de transición los grafos que representan estos autómatas serán de la siguiente forma: de cada nodo puede partir ninguna, una o varias flechas con la misma etiqueta de input. Además puede haber flechas que no se correspondan a ningún input, los ε-movimientos. A estas flechas les pondremos la etiqueta ε. Como en el caso de los autómatas determinísticos los NFA tienen un estado inicial y estados finales, que se denotarán en el diagrama de transición de la misma forma que en el caso determinístico

Ejemplo 2.10. El siguiente es el diagrama de un autómata finito no determinístico con ε-movimientos: los estados son q_0, q_1 y q_2. Los símbolos de input son a y b. El estado inicial es q_0 y el estado final es q_2.



Ejemplo 2.11. En este NFA los estados son q_0, q_1 y q_2. Los símbolos de input son 0, 1 y 2. El estado inicial es q_0 y el estado final es q_2.



El nombre "no determinístico" viene del hecho que si un estado recibe un input, entonces no está determinado cual es el próximo estado, si no que hay ciertos estados posibles. Los ε-movimientos reflejan la posibilidad de cambio de estado sin que necesariamente haya un input. Observar también, que existe la posibilidad que dado un estado no salga ninguna flecha de él. Esto denotará que en este

estado el autómata “aborta". Es claro que no es fácil imaginarse una “máquina" que se comporte de esta manera, sin embargo el concepto de NFA con ε-movimientos resulta muy útil en la teoría de lenguajes.

Las tablas que describirán las reglas de transición de los NFA con ε-mov serán similares a las tablas que hemos hecho para los autómatas finitos determinísticos. Las diferencias son que cuando aplicamos un input a un estado obtenemos un conjunto de estados (hacia donde van las flechas) o $\emptyset$ en el caso que no salga ninguna flecha. Además debemos agregar una entrada ε en la tabla para reflejar los posibles cambios de estado que producen los ε-movimientos.

Ejemplo 2.12. La tabla de transición correspondiente al NFA con ε-mov del ejemplo 2.10 es dada por la siguiente tabla:

	a	b	ε
q_0	q_1	q_0	q_2
q_1	$\emptyset$	q_0, q_2	$\emptyset$
q_2	q_1	$\emptyset$	$\emptyset$

Ejemplo 2.13. La tabla de transición correspondiente al NFA con ε-mov del ejemplo 2.11 es dada por la siguiente tabla:

	0	1	2	ε
q_0	q_0	$\emptyset$	$\emptyset$	q_1
q_1	$\emptyset$	q_1	$\emptyset$	q_2
q_2	$\emptyset$	$\emptyset$	q_2	$\emptyset$

Veamos ahora el lenguaje asociado a un NFA con ε-movimientos.

Definición 2.4. Sea M un NFA con ε-movimientos con símbolos de input Σ. Si $\alpha \in \Sigma^*$, diremos que α *transforma* p en q y lo denotaremos $p \xrightarrow{\alpha} q$, si existen $a_0, a_1, \ldots, a_n \in \Sigma$ tal que $\alpha = a_1 a_2 \ldots a_n$ y tal que existe un recorrido por flechas con etiquetas $a_0, a_1, \ldots, a_n$ que va del estado p al estado q. En el recorrido se pueden intercalar arbitrariamente flechas con etiqueta ε.

Es conveniente también definir que ε transforma todo estado en sí mismo, es decir $q \xrightarrow{\varepsilon} q$.

Ejemplo 2.14. En el autómata del ejemplo 2.11 tenemos, por ejemplo, que $q_0 \xrightarrow{01} q_1$, pues hay un camino que lleva q_0 a q_0 por 0, q_0 a q_1 por ε y q_1 a q_1 por 1.

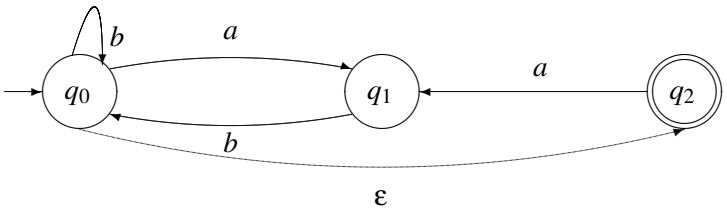
Observación 2.2. Si p y q estados, a símbolo de input y $p \xrightarrow{a} q$, no podemos asegurar que de p podemos pasar directamente a q por una flecha con etiqueta a. Por definición $p \xrightarrow{a} q$ significa que podemos llegar de p a q usando ε-movimientos además de una transición por a. Observando el autómata del ejemplo 2.11 vemos que $q_0 \xrightarrow{0} q_1$, pero no es cierto que hay una flecha con etiqueta 0 que manda q_0 a q_1.

En gran parte de las situaciones que necesitemos usar transiciones podremos tomarnos la licencia de usar la notación

$$p \xrightarrow{a_0} q_1 \xrightarrow{a_1} \cdots \xrightarrow{a_{n-1}} q_n \xrightarrow{a_n} q,$$

para indicar que hay un camino de flechas con etiquetas $a_0, \ldots, a_n$ que pasa por los estados $q_1, \ldots, q_n$ y que va de p a q (aquí permitiremos que los a_i puedan ser ε's).

Ejemplo 2.15. Sea M un NFA con ε-movimientos,con diagrama de transición :



Entonces, la tabla de transición es:

	a	b	ε
q_0	q_1	q_0	q_2
q_1	$\emptyset$	q_0	$\emptyset$
q_2	q_1	$\emptyset$	$\emptyset$

Veamos cuales son las transformaciones correspondientes a un input o un ε-movimiento:

$$\begin{array}{lll} q_0 \xrightarrow{\varepsilon} q_0, q_2 & q_0 \xrightarrow{a} q_1 & q_0 \xrightarrow{b} q_0, q_2 \\ q_1 \xrightarrow{\varepsilon} q_1 & q_1 \xrightarrow{b} q_0, q_2 & q_2 \xrightarrow{a} q_1 \\ q_2 \xrightarrow{\varepsilon} q_2 & q_2 \xrightarrow{a} q_1 & \end{array}$$

donde denotamos $p \xrightarrow{a} q, q'$ cuando $p \xrightarrow{a} q$ y $p \xrightarrow{a} q'$. La mayoría de las transformaciones son obvias y las otras se deducen de la definición. Sin embargo, creemos conveniente explicar algunas de ellas. Por ejemplo, recordemos que, por definición, $q \xrightarrow{\varepsilon} q$ para todo q estado, de allí que se justifica la primera columna de transformaciones. La transformación $q_1 \xrightarrow{b} q_2$ se deduce de que podemos llegar de q_1 a q_2 primero aplicando b y luego ε. En forma análoga se justifica $q_0 \xrightarrow{b} q_2$.

Como en el caso de autómatas determinísticos los estados finales nos determinan el lenguaje asociado al autómata.

Definición 2.5. Una cadena α es *aceptada* por el autómata si α transforma el estado inicial en uno final. Finalmente el *lenguaje aceptado por M* es el conjunto $L(M)$ de todas las cadenas aceptadas. Simbólicamente:

$$L(M) = \{\alpha \in \Sigma^* \mid q_0 \xrightarrow{\alpha} p, \text{ para algún } p \in F\}.$$

A nivel de diagramas de transición podemos entender que el lenguaje aceptado por M, un NFA con ε-movimientos, está formado por las cadenas $\alpha = b_0 b_1 \dots b_n$ tal que existe un recorrido por flechas con etiquetas $b_0, b_1, \dots, b_n$ que va del estado inicial a uno final, en el recorrido se pueden intercalar arbitrariamente flechas con etiqueta ε.

Observemos que como el ε es la cadena vacía el intercalar flechas con etiqueta ε no agrega nada a la palabra, y la palabra o cadena definitiva es aquella que no tiene ningún ε. También observemos que dada una cadena puede haber muchos recorridos partiendo del estado inicial, algunos de ellos pueden terminar en un estado final y otros no, sin embargo lo que interesa, para saber si una cadena es aceptada o no, es si por lo menos un recorrido alcanza un estado final.

Ejemplo 2.16. Averigüemos el lenguaje del autómata definido en el Ejemplo 2.11: observemos primero que si abandonamos el estado q_0 entonces vamos al estado q_1 y no podemos volver atrás. Análogamente si abandonamos el estado q_1, vamos al estado q_2 y ya no podemos salir de este estado. Luego si una cadena w alcanza el estado final, entonces comienza por q_0, pasa por q_1 y llega a q_2.

Es decir que w es de la forma $0^i1^j2^k$ (i,j,k enteros mayores o iguales a 0). Recíprocamente una cadena de la forma $0^i1^j2^k$ puede alcanzar el estado final con el siguiente recorrido: $0,0,\ldots,0$ (i-veces), ε, $1,1,\ldots,1$ (j-veces), ε, $2,2,\ldots,2$ (k-veces). Es decir que el lenguaje asociado a este autómata es $\{0^i1^j2^k : 0 \leq i,j,k\}$.

2.4.1 Formalización de los NFA

La definición formal de NFA es:

Definición 2.6. Un *autómata finito no determinístico con* ε*-movimientos (NFA con* ε*-mov)* es una 5-upla (Q,Σ,δ,q_0,F) que consiste de

1. un conjunto finito $Q = \{q_1,q_2,\ldots,q_n\}$ de *estados*,
2. un conjunto finito $\Sigma = \{a_1,a_2,\ldots,a_n\}$ de *símbolos de entrada* o *input*,
3. un conjunto de *reglas de transición* que transforma un estado con un input dado, en un conjunto posible de estados. También las reglas de transición describen la transformación de un estado en un conjunto de estados sin haber recibido input (ε-movimiento). Formalmente, las reglas de transición son dadas por una función $\delta : Q \times (\Sigma \cup \{\varepsilon\}) \to \mathcal{P}(Q)$, donde $\mathcal{P}(Q)$ es el conjunto formado por los subconjuntos de Q.
4. Además, como en el caso del DFA, hay un *estado inicial* q_0 y un conjunto de estados finales F.

Veamos ahora el lenguaje asociado a un NFA con ε-movimientos:

Definición 2.7. Sea $M = (Q,\Sigma,\delta,q_0,F)$ un NFA con ε-movimientos. Si $\alpha \in \Sigma^*$, diremos que α *transforma* p en q y lo denotaremos $p \xrightarrow{\alpha} q$, si existen $a_0,a_1,\ldots,a_n \in \Sigma \cup \{\varepsilon\}$ y $q_1,\ldots,q_n$ en Q , tal que $\alpha = a_1a_2\ldots a_n$ y

$$q_1 \in \delta(p,a_0),\ q_2 \in \delta(q_1,a_1),\ \ldots q_{i+1} \in \delta(q_i,a_i),\ \ldots q_n \in \delta(q_{n-1},a_{n-1}),\ q \in \delta(q_n,a_n).$$

Es conveniente también definir que ε transforma todo estado en sí mismo, es decir $q \xrightarrow{\varepsilon} q$.

Observación 2.3. Sea $M = (Q,\Sigma,\delta,q_0,F)$ un NFA con ε-movimientos.

1. Como dijimos antes, si $a \in \Sigma$, entonces si $q \in \delta(p,a)$ tenemos que $p \xrightarrow{a} q$. Pero no necesariamente si $p \xrightarrow{a} q$, entonces se cumple que $q \in \delta(p,a)$. La definición de $p \xrightarrow{a} q$ denota el hecho de que podemos llegar de p a q mediante ε-movimientos, luego a, luego ε-movimientos. En el autómata del ejemplo 2.11 es cierto que $q_0 \xrightarrow{0} q_1$, pues $q_0 \in \delta(q_0,0)$ y $q_1 \in \delta(q_0,\varepsilon)$, pero no es cierto que $q_1 \in \delta(q_0,0)$.
2. Cuando un autómata no tiene ε-mov, entonces coinciden la función de transición y las transformaciones por símbolos de input, es decir si p,q son estados y a es un símbolo de input,

   $$p \in \delta(q,a) \quad \text{si y sólo si} \quad q \xrightarrow{a} p.$$

   Por lo tanto, la notación $p \xrightarrow{\alpha} q$ para NFA con ε-mov, es consistente con la misma notación para DFA.
3. Es claro, en forma intuitiva, que si $\alpha,\beta \in \Sigma^*$, entonces para $p,q \in Q$,

   $$p \xrightarrow{\alpha\beta} q \quad \text{si y sólo si existe } r \in Q \text{ tal que } \quad p \xrightarrow{\alpha} r \xrightarrow{\beta} q. \tag{2}$$

4. Sea $M' = (Q, \Sigma, \delta', q_0, F)$ con δ' definida de la siguiente manera:

$$q \in \delta'(p,a) \quad \text{sii} \quad p \xrightarrow{a} q.$$

Entonces M' resulta ser un NFA con ε-mov que acepta el mismo lenguaje que M. La demostración se deja como ejercicio.

5. Debido a la observación anterior, en gran parte de las situaciones que necesitemos usar transiciones podremos tomarnos la licencia de usar la notación

$$p \xrightarrow{a_0} q_1 \xrightarrow{a_1} \cdots \xrightarrow{a_{n-1}} q_n \xrightarrow{a_n} q,$$

para indicar que hay un camino de flechas con etiquetas $a_0, \ldots, a_n$ que pasa por los estados $q_1, \ldots, q_n$ y que va de p a q. Dicho más formalmente:

$$q_1 \in \delta(p, a_0),\ q_2 \in \delta(q_1, a_1),\ \ldots q_n \in \delta(q_{n-1}, a_{n-1}),\ q \in \delta(q_n, a_n).$$

Repetimos la

Definición 2.8. Una cadena α es *aceptada* por el autómata si α transforma el estado inicial en uno final. Finalmente el *lenguaje aceptado por M* es el conjunto $L(M)$ de todas las cadenas aceptadas. Simbólicamente:

$$L(M) = \{\alpha \in \Sigma^* \mid q_0 \xrightarrow{\alpha} p, \text{ para algún } p \in F\}.$$

El siguiente Teorema fue anunciado al comienzo de la sección:

Teorema 2.1. *Los lenguajes aceptados por los autómatas finitos determinísticos son los mismos que los aceptados por los autómatas finitos no determinísticos con ε-movimientos. Más precisamente, dado un DFA (NFA con ε-movimientos) M, existe un NFA con ε-movimientos (resp. DFA) M', tal que $L(M) = L(M')$.*

Demostración. Si $M = (Q, \Sigma, \delta, q_0, F)$ un DFA, entonces sea $M' = (Q, \Sigma, \delta', q_0, F)$ el NFA definido por $\delta'(q,a) = \{\delta(q,a)\}$ para $q \in Q$ y $a \in \Sigma$. Es trivial comprobar, entonces, que $L(M) = L(M')$.

Veremos ahora que, dado $M = (Q, \Sigma, \delta, q_0, F)$ un NFA con ε-mov, podemos construir $M' = (Q'$, Σ, δ', $q'_0, F')$ un DFA, tal que $L(M) = L(M')$.

Sea $q \in M$, definamos $[q] = \{p \in Q \mid q \xrightarrow{\varepsilon} p\} \subseteq Q$. Sean $q_1, \ldots, q_i$ elementos de Q, entonces denotemos

$$[q_1, \ldots, q_i] = \cup_{j=1}^{i} [q_j].$$

Definamos M' de la siguiente manera: $Q' = \{[q_1, \ldots, q_i] \mid q_1, \ldots, q_i \in Q\}$. El nuevo conjunto de estados finales, F', es el conjunto de estados que contienen un estado que se transforma por ε en un estado final de M. El estado inicial será $q'_0 = [q_0]$. Finalmente, definimos $\delta'([q_1, \ldots, q_i], a)$ igual al conjunto $\{p \in Q \mid \text{existe } q_s \text{ tal que } q_s \xrightarrow{a} p\}$. Es decir, un $a \in \Sigma$ transforma $q' \in Q'$ en $p' \in Q'$, si el conjunto p' está formado por todos los elementos de Q que son los transformados por a de los elementos de q'. Resumiendo, si $q' \in Q'$ y $a \in \Sigma$:

$$\begin{aligned}
Q' &= \{[q_1, \ldots, q_i] \mid q_1, \ldots, q_i \in Q\} \subseteq \mathcal{P}(Q) \\
q'_0 &= [q_0] \\
F' &= \{[q_1, \ldots, q_i] \mid \text{ existe } k \text{ y } p \in F \text{ tal que } q_k \xrightarrow{\varepsilon} p\} \\
\delta(q', a) &= \{p \in Q \mid \exists q \in q' \text{ tal que } q \xrightarrow{a} p\} \quad \text{o equivalentemente} \\
q' &\xrightarrow{a} \{p \in Q \mid \exists q \in q' \text{ tal que } q \xrightarrow{a} p\}.
\end{aligned}$$

No es difícil comprobar que $M' = (Q', \Sigma, \delta', q_0', F')$ es un autómata finito determinístico.

No es difícil comprobar que si $q' \in Q'$ entonces

$$q' = \{p \in Q \mid \exists q \in q' \text{ tal que } q \xrightarrow{\varepsilon} p\}. \tag{3}$$

Esto se debe a que como $q' = [q_1, \ldots, q_i]$, por definición

$$\begin{aligned} q' &= \cup_{j=1}^{i} [q_j] = \cup_{j=1}^{i} \{p \in Q \mid q_j \xrightarrow{\varepsilon} p\} \\ &= \{p \in Q \mid \exists q_j \text{ tal que } q_j \xrightarrow{\varepsilon} p\} \subset \{p \in Q \mid \exists q \in q' \text{ tal que } q \xrightarrow{\varepsilon} p\}. \end{aligned}$$

Para demostrar la otra inclusión, consideremos p tal que $q \xrightarrow{\varepsilon} p$ con $q \in q'$. Como $q \in q'$ existe q_j tal que $q_j \xrightarrow{\varepsilon} q$, luego $q_j \xrightarrow{\varepsilon} q \xrightarrow{\varepsilon} p$, de lo cual se deduce que $q_j \xrightarrow{\varepsilon} p$ y por lo tanto $p \in \{p \in Q \mid \exists q_j \text{ tal que } q_j \xrightarrow{\varepsilon} p\}$.

Veamos ahora que si $\alpha \in \Sigma^*$, q' en Q', entonces:

$$q' \xrightarrow{\alpha} \{p \in Q \mid \exists q \in q' \text{ tal que } q \xrightarrow{\alpha} p\}.$$

Supongamos que $q' \xrightarrow{\alpha} p'$ y veamos que $p' = \{p \in Q \mid \exists q \in q' \text{ tal que } q \xrightarrow{\alpha} p\}$: hagamos inducción sobre $|\alpha|$. Cuando $|\alpha| = 0$, es decir cuando $\alpha = \varepsilon$, tenemos que $q' \xrightarrow{\varepsilon} q'$ y el párrafo anterior demuestra el resultado. Cuando $|\alpha| = 1$ el resultado se deduce trivialmente por definición de transición. Si $|\alpha| > 1$, entonces $\alpha = \beta a$, con $|\beta| \geq 1$ y $a \in \Sigma$. Por (1) de la observación 2.1 sabemos que existe $r' \in Q'$ tal que $q' \xrightarrow{\beta} r' \xrightarrow{a} p'$. Por hipótesis inductiva y el caso de longitud 1 tenemos que

$$\begin{aligned} q' \xrightarrow{\beta} r' &= \{r \in Q \mid \exists q \in q' \text{ tal que } q \xrightarrow{\beta} r\}, \\ r' \xrightarrow{a} p' &= \{p \in Q \mid \exists r \in r' \text{ tal que } r \xrightarrow{a} p\}. \end{aligned}$$

Si $p \in p'$, existe $r \in r'$ tal que $r \xrightarrow{a} p$, como $r \in r'$, existe $q \in q'$, tal que $q \xrightarrow{\beta} r$. Es decir que $q \xrightarrow{\beta a} p$ o equivalentemente $q \xrightarrow{\alpha} p$. Por lo tanto está probado $p' \subseteq \{p \in Q \mid \exists q \in q' \text{ tal que } q \xrightarrow{\alpha} p\}$. Por otro lado, si $p \in \{p \in Q \mid \exists q \in q' \text{ tal que } q \xrightarrow{\alpha} p\}$, existe $q \in q'$ tal que $q \xrightarrow{\alpha} p$, luego existe $r \in Q$ tal que $q \xrightarrow{\beta} r \xrightarrow{a} p$, por lo tanto $r \in r'$ y $p \in p'$. Esto prueba la otra inclusión y por lo tanto la igualdad deseada.

La aplicación directa del resultado anterior a q_0', nos dice que dada $\alpha \in \Sigma^*$, entonces

$$q_0' \xrightarrow{\alpha} \{p \in Q \mid q_0 \xrightarrow{\alpha} p\}. \tag{4}$$

Ahora bien, sea $\alpha \in L(M')$, es decir $q_0' \xrightarrow{\alpha} p'$, con $p' \in F'$. Como p' es final de M', entonces existe p en p' que también pertenece a F, luego por (4) obtenemos que $q_0 \xrightarrow{\alpha} p$, es decir $\alpha \in L(M)$. Recíprocamente, sea $\alpha \in L(M)$, es decir existe $p \in F$ tal que $q_0 \xrightarrow{\alpha} p$, por lo tanto (usando nuevamente (4)) $p \in p'$ con $q_0' \xrightarrow{\alpha} p'$. De esto se deduce que $p' \in F'$ y $\alpha \in L(M')$.

Finalmente, el párrafo anterior prueba que $L(M) = L(M')$.

□

Ejemplo 2.17. Sea M el NFA con ε-mov con estados q_0, q_1, un solo símbolo de input a, estado inicial q_0, estado final q_1 y las siguientes leyes de transición:

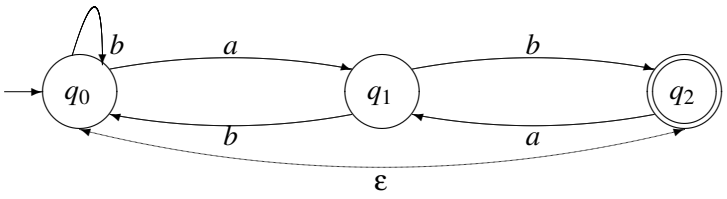
$$\delta(q_0, a) = \{q_0\}, \qquad \delta(q_0, \varepsilon) = \{q_1\}.$$

Encontremos un autómata determinístico con el mismo lenguaje que M. Por la demostración del teorema el nuevo autómata, M', tendrá estados $\emptyset, [q_0]$. Observar que $[q_0] = [q_1] = [q_0, q_1] = \{q_0, q_1\}$,

pues $q_0 \xrightarrow{\varepsilon} q_1$. El estado final e inicial es entonces $[q_0]$. Por definición, del estado $\emptyset$ no sale ninguna flecha y $\delta([q_0],\varepsilon) = [q_0]$, $\delta([q_0],a) = [q_0]$. Claramente, el lenguaje aceptado por ambos autómatas es el de todas las cadenas de a's.

Veamos un ejemplo menos trivial.

Ejemplo 2.18. Sea $M = (Q,\Sigma,\delta,q_0,F)$ un NFA con ε-movimientos definido por el siguiente diagrama de transición.



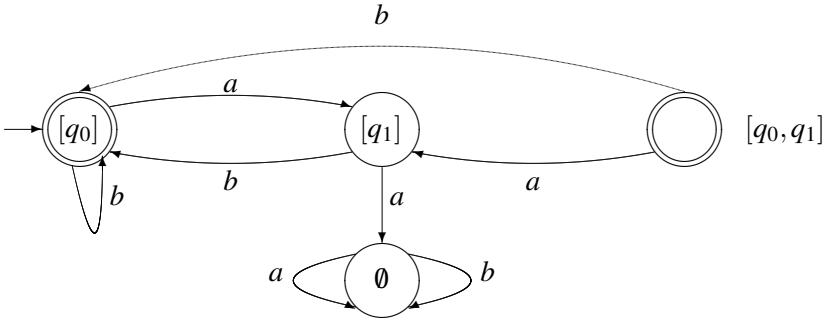
Encontremos un DFA que acepte el mismo lenguaje. Debemos primero establecer los estados del nuevo autómata:

$$\begin{aligned} [q_0] &= \{p \in Q \mid q_0 \xrightarrow{\varepsilon} p\} = \{q_0, q_2\} \\ [q_1] &= \{p \in Q \mid q_1 \xrightarrow{\varepsilon} p\} = \{q_1\} \\ [q_0,q_1] &= [q_0] \cup [q_1] = \{q_0,q_1,q_2\} \end{aligned}$$

Otro estado, siempre presente es $\emptyset$. Observemos que cualquier otro posible estado es uno de los ya listados más arriba, por ejemplo, $[q_2] = [q_0]$ y $[q_1,q_2] = [q_1] \cup [q_2] = [q_0,q_1]$. Establezcamos ahora las transiciones.

$$\begin{aligned} [q_0] &\xrightarrow{a} \{p \in Q \mid \exists q \in [q_0] \text{ tal que } q \xrightarrow{a} p\} = \\ &\qquad = \{p \in Q \mid q_0 \xrightarrow{a} p\} \cup \{p \in Q \mid q_2 \xrightarrow{a} p\} = \{q_1\} \cup \{q_1\} = [q_1] \\ [q_0] &\xrightarrow{b} \{p \in Q \mid q_0 \xrightarrow{b} p\} \cup \{p \in Q \mid q_2 \xrightarrow{b} p\} = \{q_0,q_2\} \cup \emptyset = [q_0] \\ [q_1] &\xrightarrow{a} \{p \in Q \mid q_1 \xrightarrow{a} p\} = \emptyset \\ [q_1] &\xrightarrow{b} \{p \in Q \mid q_1 \xrightarrow{b} p\} = \{q_0,q_2\} = [q_0] \\ [q_0,q_1] &\xrightarrow{a} \{p \in Q \mid \exists q \in [q_0,q_1] \text{ tal que } q \xrightarrow{a} p\} = [q_1] \cup \emptyset = [q_1] \\ [q_0,q_1] &\xrightarrow{b} [q_0] \cup [q_0] = [q_0] \end{aligned}$$

Por definición, es claro que toda transición que sale de $\emptyset$ vuelve a $\emptyset$. Finalmente, el estado inicial y final es $[q_0]$. El diagrama de transición del autómata recién construido es:



2.5 Expresiones regulares

Los lenguajes aceptados por los autómatas finitos se pueden describir fácilmente por expresiones simples llamadas expresiones regulares. En esta sección introduciremos las operaciones de concatenación y clausura sobre conjuntos de cadenas, definiremos expresiones regulares y daremos la demostración de que una expresión regular define un lenguaje que puede ser descrito por un autómata finito. La recíproca de este resultado también es cierta y veremos su demostración en la subsección final.

Definición 2.9. Sea Σ un conjunto finito de símbolos y sean L, L_1 y L_2 conjuntos de cadenas de Σ^*. La *concatenación* de L_1 y L_2, denotada L_1L_2 es un conjunto formado por cadenas de L_1 seguidas por cadenas de L_2, es decir: $L_1L_2 = \{xy \mid x \text{ está en } L_1 \text{ e } y \text{ está en } L_2\}$. Definamos $L^0 = \{\varepsilon\}$ y $L^i = LL^{i-1}$ para $i \geq 1$. La *clausura de Kleene* (o sólo *clausura*) de L, denotada L^*, es el conjunto

$$L^* = \cup_{i=0}^{\infty} L^i$$

y la clausura positiva de L, denotada L^+, es el conjunto

$$L^+ = \cup_{i=1}^{\infty} L^i.$$

L^* denota las cadenas construidas por concatenación de un número arbitrario de cadenas de L. L^+ es igual pero sin poner L^0. Nótese que L^+ contiene ε si y sólo si L lo contiene. Recordemos que en la Sección 2.1 definimos Σ^*, y observemos que ese conjunto coincide con el Σ^* que surge de la definición anterior, considerando a Σ como un lenguaje donde las cadenas son los símbolos del alfabeto.

Ejemplo 2.19. Sea $L_1 = \{11, 0\}$ y $L_2 = \{001, 10\}$, entonces

$$L_1L_2 = \{11001, 1110, 0001, 010\}.$$

Por otro lado,

$$L_1^* = \{11, 0\}^* = \{\varepsilon, 11, 0, 1111, 110, 011, 00, 111111, 11110, 11011, 1100, \ldots\}.$$

L_1^+ es L_1^* menos el ε.

Definición 2.10. Sea Σ un alfabeto. Las *expresiones regulares* sobre Σ y los *conjuntos* que ellas denotan se definen recursivamente de la siguiente manera:

1. $\emptyset$ es una expresión regular y denota el conjunto vacío.
2. ε es un expresión regular y denota el conjunto $\{\varepsilon\}$.
3. Para cada a en Σ, a es una expresión regular que denota el conjunto $\{a\}$.
4. Si r y s son expresiones regulares que denotan los conjuntos R y S, respectivamente, entonces $(r+s)$, (rs) y (r^*) son expresiones regulares que denotan los conjuntos $R \cup S$, RS y R^*, respectivamente.

Si r es una expresión regular escribiremos $L(r)$ al conjunto (o lenguaje) denotado por r.

Cuando escribimos expresiones regulares podemos omitir muchos paréntesis si asumimos que $*$ tiene más alta precedencia que la concatenación o $+$, y la concatenación tiene más alta precedencia que $+$. Por ejemplo $((0(1^*))+0)$ puede ser escrita 01^*+0. Finalmente, si r es una expresión regular denotemos r^i a la expresión $rr\cdots r$ (i veces).

Debemos tener mucho cuidado en no confundir cadenas con expresiones regulares y para ello debemos aclarar debidamente a qué nos estamos refiriendo. Por ejemplo la expresión regular ab denota un conjunto cuyo único elemento es la cadena ab.

Ejemplo 2.20. Los siguientes son ejemplo de expresiones regulares:

1. 00 es una expresión regular que representa $\{00\}$.
2. $(0+1)^*$ denota todas las cadenas de 0's y 1's.
3. $(0+1)^*00(0+1)^*$ denota todas las cadenas de 0's y 1's con al menos dos 0's consecutivos.
4. $(1+10)^*$ denota todas las cadenas de 0's y 1's que comienzan con 1 y no tienen dos 0's consecutivos. También pertenece a este conjunto la cadena vacía.

   Demostración. Probemos por inducción que $(1+10)^i$ no tiene dos 0's consecutivos. El caso $i=0$ es trivial, y si suponemos que $(1+10)^{i-1}$ no tiene dos 0's consecutivos, entonces, es claro que $(1+10)^i = (1+10)^{i-1}(1+10)$ no tiene dos 0's consecutivos, pues en el final las palabras son de la forma $\cdots\mathbf{1}1$ o $\cdots\mathbf{1}10$ o $\cdots\mathbf{10}1$ o $\cdots\mathbf{10}10$, donde las letras en negrita denotan palabras de $(1+10)^{i-1}$. De esta forma probamos que $(1+10)^i$ no tiene dos 0's consecutivos para todo i y por lo tanto $(1+10)^*$ no tiene dos 0's consecutivos. Por otro lado, dada cualquier cadena que comience con 1 y no tenga dos 0's consecutivos, se puede hacer una partición de la cadena de tal forma que si un 1 no es seguido de un 0, se toma el 1 (que pertenece a $(1+10)$) y si un 1 es seguido de un 0, se toma el 10 (que pertenece a $(1+10)$). Por ejemplo, 11010111 es particionado $1-10-10-1-1-1$ que pertenece a $(1+10)^6$. □
5. Basado en lo anterior, es claro que $(0+\varepsilon)(1+10)^*$ denota las cadenas de 0's y 1's que no tienen dos 0's seguidos.
6. $(0+1)^*011$ denota las cadenas de 0's y 1's que terminan en 001.
7. $0^*1^*2^*$ denota las cadenas que están formadas por cierta cantidad de 0's, luego cierta cantidad de 1's y luego cierta cantidad de 2's. Observar que este es lenguaje del autómata definido en el ejemplo 2.11.

Veremos ahora como a partir de una expresión regular r podemos construir un autómata no determinístico M que defina el mismo lenguaje, es decir tal que $L(r) = L(M)$. En realidad lo que construiremos son autómatas no determinísticos con un sólo estado final. Para las expresiones regulares ε, $\emptyset$ o a con $a \in \Sigma$ los autómatas no determinísticos de la Fig. 3 definen el lenguaje correspondiente:

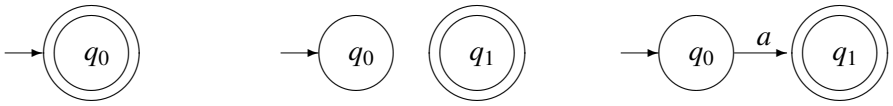


Figura 3:

A un autómata M con estado inicial q y estado final f lo dibujaremos de la siguiente manera (Fig. 4):

Ahora haremos lo siguiente: dadas expresiones regulares r_1 y r_2 a las cuales ya les hayamos asociado los autómatas M_1, M_2 correspondientes, como en la Fig. 5 encontraremos los autómatas correspondientes a $r_1 + r_2$, $r_1 r_2$ y r_1^*. Dicho de otra manera: si L_1 es el lenguaje de M_1 y L_2 es el lenguaje de M_2, entonces encontraremos autómatas correspondientes a $L_1 \cup L_2$, $L_1 L_2$ y L_1^*. El autómata correspondiente a $r_1 + r_2$ será el de la Fig. 6.

El autómata correspondiente a $r_1 r_2$ será el de la Fig. 7.

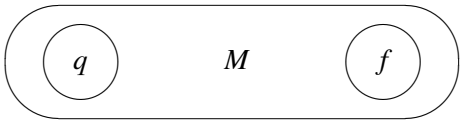


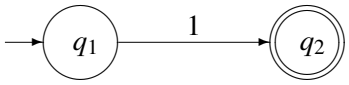
Figura 4:

Figura 5: Los autómatas de r_1 y r_2.

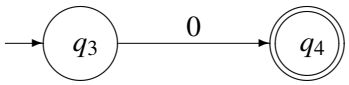
El autómata correspondiente a r_1^* será el de la Fig. 8
De lo anterior se deduce

Teorema 2.2. *Sea L una lenguaje denotado por la expresión regular r. Entonces existe M un DFA con* ε*-mov tal que* $L = L(M)$.

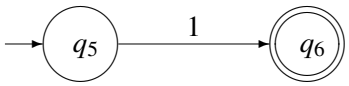
Ejemplo 2.21. Construyamos un autómata para la expresión regular $01^* + 1$. Por lo dicho anteriormente (respecto a los paréntesis), esta expresión es en realidad $((0(1^*)) + 1)$, es decir es de la forma $r_1 + r_2$, donde $r_1 = 01^*$ y $r_2 = 1$. El autómata para r_2 es fácil



Podemos expresar r_1 como $r_3 r_4$, donde $r_3 = 0$ y $r_4 = 1^*$. El autómata de r_3, también es fácil:



Ahora bien, r_4 es r_5^*, con $r_5 = 1$. Un autómata para r_5 es



Observemos que para poder construir el autómata correspondiente a la expresión regular necesitamos distinguir los estados de r_2 y r_5, aunque representen la misma expresión. Basándonos en las explicaciones anteriores un autómata para r_4 será

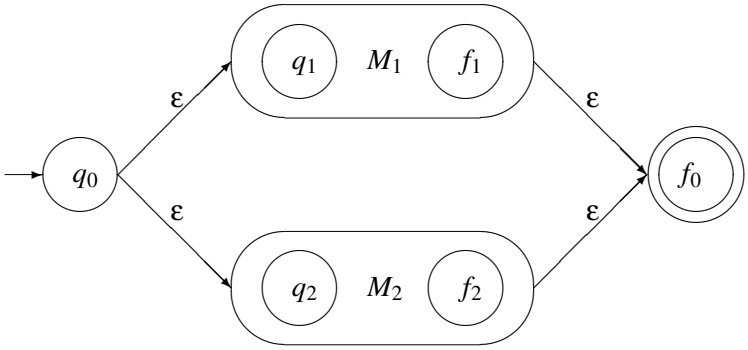


Figura 6: El autómata de $r_1 + r_2$.

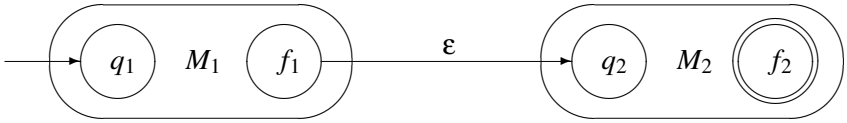
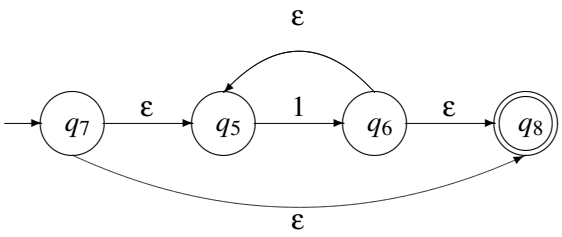
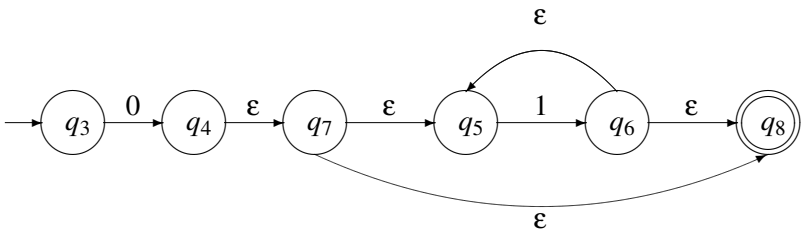


Figura 7: El autómata de $r_1 r_2$.



Luego el autómata correspondiente a la expresión regular $r_1 = 01^*$ será



Finalmente el autómata correspondiente a $01^* + 1$ será el de la Fig. 9.

2.5.1 Teorema de Kleene

Estudiemos el problema, un poco más complicado, de construir una expresión regular que denote el lenguaje de un autómata dado. Esta construcción se basa en la idea de encontrar, en forma algorítmica, una expresión regular que involucra resolver el problema para un autómata con menos estados. Repitiendo el proceso un número conveniente de veces lograremos la expresión regular buscada. Para

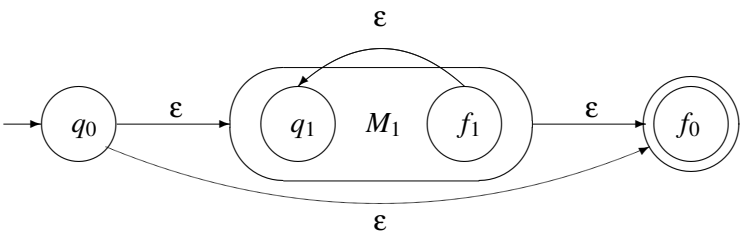


Figura 8: El autómata de r_1^*.

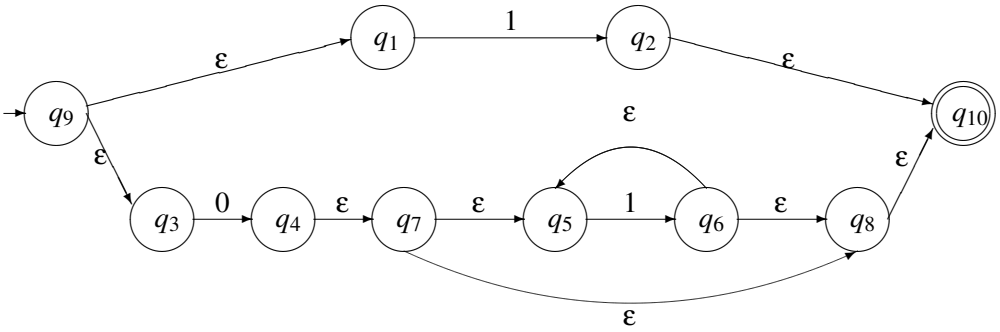


Figura 9: Un autómata correspondiente a la expresión regular $01^* + 1$.

describir el procedimiento, comenzaremos con el caso más simple en el cual nuestro autómata M tiene un solo estado final, q_n y su estado inicial es q_0. Es claro que $L(M) = L_{0n}$, el lenguaje de las cadenas que comienzan en q_0 y llegan a q_n. Llamemos I_0 al lenguaje formado por las cadenas que salen de q_0 y vuelven a q_0 *sin pasar nuevamente por él.* Si el estado inicial es el estado final, (es decir, $n = 0$) entonces claramente $L_{0n} = I_0^*$, puesto que con repetir caminos que salgan de q_0 y vuelvan a él obtengo todas las palabras aceptadas. Si $n \neq 0$, definamos F_{0n} como el lenguaje de las cadenas que salen de q_0 y llegan a q_n *sin pasar por* q_0. Entonces, por un razonamiento similar, $L_{0n} = I_0^* F_{0n}$.

Debemos ahora explicitar quiénes son I_0 y F_{0n}. Los elementos de I_0 son cadenas de dos formas:

1. $a\beta_0 b$, donde $a, b \in \Sigma \cup \{\varepsilon\}$ y β_0 es una cadena que parte de un estado p y tal que $q_0 \xrightarrow{a} q_i \xrightarrow{\beta_0} q_j \xrightarrow{b} q_0$ con p y q distintos de q_0 y además el camino que recorre β_0 no pasa por q_0; o bien
2. $c \in \Sigma \cup \{\varepsilon\}$, que haga un bucle de q_0 en sí mismo.

Si consideramos en el primer caso el autómata M' que se obtiene de M sacando q_0 y haciendo que q_i sea estado inicial y q_j estado final, entonces β_0 es una cadena aceptada por M'. En este caso, $L(M') = L_{ij}$, el lenguaje de las cadenas que salen de q_i y llegan a q_j sin pasar nunca por q_0, y obtenemos:

$$I_0 = aL_{ij}b + \cdots + c + \ldots$$

donde se suman todas las posibilidades para a, b, i, j tales que $q_0 \xrightarrow{a} q_i$, $q_j \xrightarrow{b} 0$, con q_0 distinto de q_i y q_j, los c tales que $q_0 \xrightarrow{c} q_0$, y en L_{ij} no consideramos q_0. Es decir, las cadenas que parten y llegan a q_0 se pueden ver como la concatenación de cadenas formadas por ciertos símbolos de entrada, con lenguajes aceptados por autómatas más chicos.

De igual modo, si q_0 no es el estado final, los elementos de F_{0n} son las cadenas que parten de q_0 y llegan al estado final q_n sin pasar por q_0. En este caso las cadenas son de la forma $a\beta$ con

$q_0 \xrightarrow{a} q_j \xrightarrow{\beta} q_n$, donde $a \in \Sigma \cup \{\varepsilon\}$. Luego esta cadena es la concatenación de un símbolo de entrada y una cadena aceptada por un autómata más chico:

$$F_{0n} = aL_{jn} + \ldots$$

donde a, j se mueven entre todas las opciones tales que $q_0 \xrightarrow{a} q_j \xrightarrow{\beta} q_n$, $j \neq 0$ y en L_{ij} (igual que arriba) no consideramos a q_0. Dado que el problema se va reduciendo a autómatas mas chicos, este algoritmo termina.

El caso de autómatas M con múltiples estados finales se trata similarmente, escribiendo

$$L(M) = aL_{jk} + \ldots,$$

donde a, j, k se mueven entre las opciones tales que $q_0 \xrightarrow{a} q_j \xrightarrow{\beta} q_k$ con q_k un estado final.

Ejemplo 2.22. Encontremos una expresión regular para el lenguaje aceptado por el autómata M descrito mediante el diagrama de transición de la Figura 10.

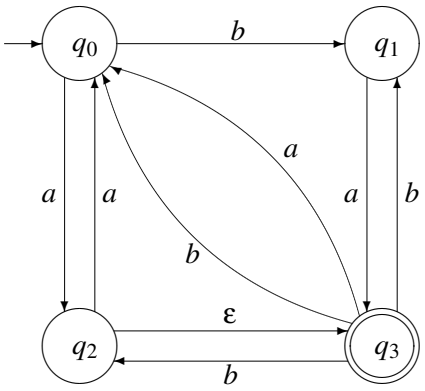


Figura 10: El autómata M

Es claro que $L(M) = L_{03}$, el lenguaje de las cadenas que comienzan en q_0 y llegan a q_3. Llamemos I_0 al lenguaje formado por las cadenas que salen de q_0 y vuelven a q_0 *sin pasar nuevamente por él* y F_{03} al lenguaje de las cadenas que salen de q_0 y llegan a q_3 *sin pasar por* q_0. Entonces, $L_{03} = I_0^* F_{03}$.

Analicemos ahora quiénes son I_0 y $F_{0,3}$.

Las formas de salir de q_0 y volver a sí mismo sin pasar nuevamente por él se pueden escribir así:

$$I_0 = bL_{12}a + bL_{13}a + bL_{13}b + aL_{23}a + aL_{23}b + aL_{22}a,$$

$$I_0 = b(L_{12}a + L_{13}(a+b)) + a(L_{23}(a+b) + L_{22}a),$$

donde L_{12} es el lenguaje aceptado por el autómata M_{12} que consta de los estados $\{q_1, q_2, q_3\}$ (hemos eliminado q_0), y que tiene por estado inicial a q_1 y final a q_2. De manera similar se definen $L_{13} = L(M_{13})$, $L_{23} = L(M_{23})$ (ver Figura 12) y L_{22} (ver Figura 11).

Las maneras de salir de q_0 y llegar al estado final q_3 sin pasar por q_0 son:

$$F_{03} = bL_{13} + aL_{23}.$$

Tenemos, como antes $L_{12} = I_1^* F_{12}$, salvo que ahora hemos eliminado q_0 de nuestras consideraciones. Es decir,

$$I_1 = aL_{33}b, \qquad F_{12} = aL_{33},$$

son las formas de ir de q_1 a q_1 y de q_1 a q_3, respectivamente, sin pasar nuevamente por q_1 (y sin utilizar q_0). Ahora, en nuestro L_{33}, no utilizaremos ni q_0 ni q_1.

> Como nuestro estado inicial es igual al estado final (q_3), tenemos
>
> $$L_{33} = I_3^*.$$
>
> En I_3 no usaremos ni q_0 ni q_1. Entonces obtenemos:
>
> $$I_3 = bL_{22}\varepsilon = b,$$
>
> con lo que
>
> $$L_{33} = b^*.$$

Ya podemos calcular los anteriores $I_1 = aL_{33}b = ab^*b$ y $F_{12} = aL_{33} = ab^*$. Luego obtenemos

$$L_{12} = (ab^*b)^*ab^*.$$

Ahora nos toca ver $L_{13} = I_1^* F_{13}$, eliminando q_0.

$$I_1 = ab^*b, \qquad F_{13} = aL_{33} = ab^*,$$

dado que, como antes, en L_{33} eliminamos q_0 y q_1. Entonces $L_{13} = (ab^*b)^*ab^*$.

Observemos que $L_{23} = I_2^* F_{23}$, eliminando q_0.

$$I_2 = \varepsilon L_{33}b = L_{33}b, \qquad F_{23} = \varepsilon L_{33} = L_{33},$$

donde en L_{33} se eliminan q_0 y q_2.

> Veamos L_{33} con q_0 y q_2 eliminados. Como antes (estado inicial es el final) tenemos
>
> $$L_{33} = I_3^*.$$
>
> En I_3 no usaremos ni q_0 ni q_2. Entonces obtenemos:
>
> $$I_3 = bL_{11}a = ba$$
>
> con lo que
>
> $$L_{33} = (ba)^*.$$

Obtenemos entonces

$$I_2 = (ba)^*b, \qquad F_{23} = (ba)^*,$$

y luego $L_{23} = ((ba)^*b)^*(ba)^*$.

Sólo nos falta $L_{22} = I_2^*$ (Figura 11). Tenemos

$$I_2 = \varepsilon L_{33} b = L_{33} b = (ba)^* b,$$

pues es el mismo L_{33} considerado anteriomente. Entonces

$$L_{22} = ((ba)^* b)^*.$$

Por último,

$$I_0 = b(L_{12}a + L_{13}(a+b)) + a(L_{23}(a+b) + L_{22}a),$$

$$I_0 = b((ab^*b)^* ab^* a + (ab^*b)^* ab^*(a+b)) + a(((ba)^*b)^*(ba)^*(a+b) + ((ba)^*b)^* a).$$

$$F_{03} = bL_{13} + aL_{23},$$

$$F_{03} = b(ab^*b)^* ab^* + a((ba)^*b)^*(ba)^*,$$

y nuestro resultado final es

$$L = L_{03} = I_0^*\, F_{03}.$$

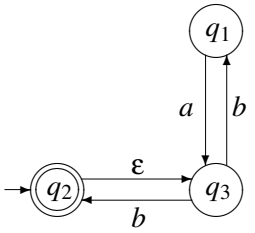


Figura 11: El autómata M_{22}

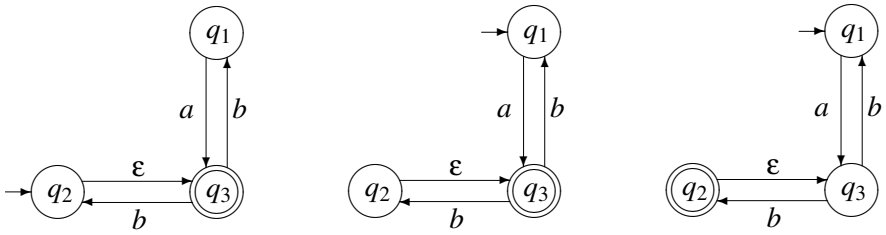


Figura 12: Los autómatas M_{23}, M_{13} y M_{12}, respectivamente

Teorema 2.3 (Teorema de Kleene)**.** *Sea M un NFA con ε-mov, entonces, existe una expresión regular r tal que $L(M)$ es el lenguaje denotado por r.*

Probaremos este teorema viendo que, dado un NFA M, el algoritmo descripto anteriormente devuelve una expresión regular que caracteriza el lenguaje aceptado por M. Para ello será necesario formalizar las definiciones de L_{nm}, I_n, F_{nm} y el proceso de "eliminar estados".

Supongamos que $M = (Q, \Sigma, \delta, q_0, F)$. Supondremos que existe un único estado final: $F = \{q_f\}$, con $0 \leq f \leq r$. Sea $Q = \{q_0, \ldots, q_r\}$ y $\Sigma = \{c_1, \ldots, c_u\}$.

Dados $0 \leq n, m \leq r$, $R \subseteq Q$, definamos recursivamente las siguientes expresiones regulares:

$$\begin{array}{rl} L_{nm}(R) & = \emptyset \qquad \text{si } n \text{ ó } m \text{ no están en } R. \\ L_{nn}(R) & = I_n(R)^* \\ L_{nm}(R) & = I_n(R)^* F_{nm}(R) \qquad \text{si } n \neq m. \\ I_n(R) & = \sum_{q_n \xrightarrow{a} q_t,\, q_s \xrightarrow{b} q_n} a L_{ts}(R \setminus \{q_n\}) b \;\; + \sum_{q_n \xrightarrow{c} q_n} c \\ F_{nm}(R) & = \sum_{q_n \xrightarrow{a} q_t} a L_{tm}(R \setminus \{q_n\}), \end{array}$$

donde $a, b, c \in \Sigma \cup \{\varepsilon\}$. En particular,

$$L_{nn}(\{q_n\}) = I_n(\{q_n\})^* = \left(\sum_{q_n \xrightarrow{c} q_n} c \right)^*.$$

En esencia, en las expresiones $L_{nm}(R)$ sólo consideramos "habilitados" los estados del conjunto R.

Por otro lado, sea $M_{nm}(R) = (R, \Sigma, \delta \mid R, q_n, \{q_m\})$ el autómata que resulta de M luego eliminar todos los estados que no están en R y sus transiciones, y estipulando que el estado inicial es q_n y el único estado final es q_m. En caso de que q_n ó q_m no estén en R, $M_{nm}(R)$ será un autómata vacío con lenguaje $\emptyset$.

Lema 2.4. *Toda cadena α representada por las expresiones regulares de arriba son aceptadas por los respectivos autómatas. Es decir: para todo $R \subseteq Q$, $L_{nm}(R) \subseteq L(M_{nm}(R))$.*

Demostración. Tenemos dos casos a probar, correspondientes a la definición de $L_{nm}(R)$:

(1) Si $q_n \in R$ y $\alpha \in I_n(R)^k$ entonces $\alpha \in L(M_{nn}(R))$.
(2) Si $n \neq m$, $q_n, q_m \in R$ y $\alpha \in I_n(R)^k F_{nm}(R)$ entonces $\alpha \in L(M_{nm}(R))$.

Procederemos simultáneamente por inducción en k y en $|\, R \,|$.

El caso base para R es el conjunto vacío, para el cual los antecedentes de (1) y (2) son falsos. Más aún, por las definiciones tenemos que si alguno de n ó m falta en R, $L_{nm}(R) = L(M_{nm}(R)) = \emptyset$. Luego, podemos considerar de ahora en adelante que $n, m \in R$.

$\boxed{k=0}$ En el caso (1) debe ser $\alpha = \varepsilon$ y claramente $\varepsilon \in L(M_{nn}(R))$ al ser q_n estado inicial y final.
En el caso (2), $\alpha \in F_{nm}(R) = \sum_{q_n \xrightarrow{a} q_t} a L_{tm}(R \setminus \{q_n\})$. Luego existen a, t, α' tales que

$$q_n \xrightarrow{a} q_t, \quad \alpha' \in L_{tm}(R \setminus \{q_n\}), \quad \alpha = a\alpha'.$$

Por hipótesis inductiva (eliminamos un estado) $\alpha' \in L(M_{tm}(R \setminus \{q_n\}))$ y luego existe una sucesión

$$q_t = s_1 \xrightarrow{a_1} s_2 \xrightarrow{a_2} \cdots \xrightarrow{a_l} s_{l+1} = q_m$$

con $s_j \in R \setminus \{q_n\}$ y $\alpha' = a_1 a_2 \ldots a_l$. Como $q_n \xrightarrow{a} q_t$ entonces $\alpha \in L(M_{nm}(R))$.

$\boxed{k+1}$ Caso (1): es muy similar al (2), lo dejamos como ejercicio.
Caso (2): Supongamos que $\alpha \in I_n(R)^{k+1} F_{nm}(R)$. Luego, hay dos opciones (para cada uno de los tipos de sumandos en la definición de I_n):

- $\alpha = a_0\alpha'\beta'$ con $q_n \xrightarrow{a_0} q_n$, $\alpha' \in I_n(R)^k$ y $\beta' \in F_{nm}(R)$. Por hipótesis inductiva (disminuimos k), $\alpha'\beta' \in L(M_{nm}(R))$. Como antes, hay una sucesión

$$q_n = s_1 \xrightarrow{a_1} s_2 \xrightarrow{a_2} \cdots \xrightarrow{a_l} s_{l+1} = q_m$$

tal que $\alpha'\beta' = a_1 a_2 \dots a_l$ y luego

$$q_n \xrightarrow{a_0} q_n \xrightarrow{a_1} s_2 \xrightarrow{a_2} \cdots \xrightarrow{a_l} s_{l+1} = q_m$$

muestra que $\alpha \in L(M_{nm}(R))$.

- $\alpha = a_0\gamma b_0\alpha'\beta'$ con $q_n \xrightarrow{a_0} s_0$, $t_0 \xrightarrow{b_0} q_n$, $\gamma \in L_{s_0 t_0}(R \setminus \{q_n\})$, $\alpha' \in I_n(R)^k$ y $\beta' \in F_{nm}(R)$. Por hipótesis inductiva $\alpha'\beta' \in L(M_{nm}(R))$ (disminuimos k) y $\gamma \in L(M_{s_0 t_0}(R \setminus \{q_n\}))$ (borramos q_n). Es decir

$$s_0 \xrightarrow{\gamma} t_0, \quad q_n \xrightarrow{\alpha'\beta'} q_m,$$

y luego

$$q_n \xrightarrow{a_0} s_0 \xrightarrow{\gamma} t_0 \xrightarrow{b_0} q_n \xrightarrow{\alpha'\beta'} q_m$$

muestra que $\alpha \in L(M_{nm}(R))$.

□

Lema 2.5. *Toda cadena α aceptada por $M_{nm}(R)$ está en el lenguaje representado por la expresión regular dada por el algoritmo. Es decir: para todo $R \subseteq Q$, $L(M_{nm}(R)) \subseteq L_{nm}(R)$.*

Demostración. En esta prueba es más conveniente por motivos técnicos considerar las sucesiones de transiciones que dan lugar a las cadenas aceptadas, que considerar las cadenas solamente. Probaremos, para toda sucesión de transiciones en M

$$q_n = s_0 \xrightarrow{a_0} s_1 \xrightarrow{a_1} \cdots \xrightarrow{a_{l-1}} s_l = q_m$$

(donde $a_i \in \Sigma \cup \{\varepsilon\}$ y $s_i \in Q$), que:

si $s_i \in R$ para todo i entonces $a_1 \dots a_{l-1} \in L_{nm}(R)$,

por inducción en la longitud l de la sucesión. Llamemos $\alpha = a_1 \dots a_{l-1}$. Consideraremos un "caso (1)" cuando $n = m$ y un "caso (2)" cuando no sean iguales.

$\boxed{l = 0}$ Caso (1): si la cantidad de transiciones es cero, no nos movemos de q_n. La cadena representada es la vacía, y obviamente está en $I_n(R)^* = L_{nn}(R)$.

Caso (2): al ser $n \neq m$, l tiene que ser mayor a cero, así que este caso base es trivial.

$\boxed{l+1}$ Caso (1): si todos los s_i son iguales a q_n, entonces claramente

$$\alpha \in \left(\sum_{q_n \xrightarrow{c} q_n} c \right)^* \subseteq I_n(R)^* = L_{nn}(R)$$

Sino, tomemos $0 \leq j \leq h \leq l$ tales que s_i es distinto de q_n para todo i entre j y h, pero $s_{j-1} = s_{h+1} = q_n$. Luego existen α', β y γ (eventualmente vacías) tales que:

$$q_n = s_0 \xrightarrow{\alpha'} s_{j-1} \xrightarrow{a_{j-1}} s_j \xrightarrow{\beta} s_h \xrightarrow{a_h} s_{h+1} \xrightarrow{\gamma} s_l = q_n$$

Por hipótesis inductiva, tenemos $\alpha', \gamma \in L_{nn}(R) = I_n(R)^*$ y $\beta \in L_{jh}(R \setminus \{q_n\})$.[1] Luego

$$\alpha \in I_n(R)^* \left(a_{j-1} L_{jh}(R \setminus \{q_n\}) a_h\right) I_n(R)^* \subseteq I_n(R)^* I_n(R) I_n(R)^* = I_n(R)^*$$

Caso (2): sea s_h es el último s_j que es igual a q_n (notar que h debe ser menor que l puesto que $n \neq m$). Luego

$$q_n = s_0 \xrightarrow{\alpha'} s_h \xrightarrow{a_h} s_{h+1} \xrightarrow{\beta'} s_l = q_m$$

con $\alpha = \alpha' a_h \beta'$ con α' y β' eventualmente vacíos. Ahora, la sucesión de transiciones correspondiente a β' no contiene a q_n; por hipótesis inductiva tenemos que $\beta' \in L_{km}(R \setminus \{q_n\})$, donde suponemos sin pérdida de generalidad que $s_{h+1} = q_k$. Y también por hipótesis inductiva $\alpha' \in L_{nn}(R) = I_n(R)^*$. Entonces

$$\alpha \in I_n(R)^* a_h L_{km}(R \setminus \{q_n\}) \subseteq I_n(R)^* F_{nm}(R) = L_{nm}(R).$$

□

Prueba del Teorema de Kleene. Supongamos $M = (Q, \Sigma, \delta, q_0, F)$ donde $Q = \{q_0, \ldots, q_r\}$ y $F = \{q_k, \ldots, q_m\}$ con $0 \leq k \leq m \leq r$. Está claro que

$$L(M) = \bigcup_{k \leq i \leq m} L(M_{0i}(Q)),$$

dado que cada cadena aceptada termina obviamente en un solo estado final. Pero ahora, por los lemas anteriores tenemos

$$L(M) = \bigcup_{k \leq i \leq m} L_{0i}(Q)$$

y por último

$$L(M) = \sum_{k \leq i \leq m} L_{0i}(Q)$$

es la expresión regular buscada. □

2.6 Pumping Lemma

En esta sección veremos que existen lenguajes que no son regulares. Demostraremos esto usando un resultado conocido como el Pumping Lemma o "Lema del Inflado".

Proposición 2.6 (Pumping Lemma)**.** *Sea L un lenguaje regular. Entonces existe un número $k > 0$ tal que para toda cadena $\alpha \in L$ con $| \alpha | \geq k$, existen cadenas β_1, β_2, γ con $| \beta_1 \gamma | \leq k$, $| \gamma | > 0$ que satisfacen*

1. $\alpha = \beta_1 \gamma \beta_2$,
2. $\beta_1 \gamma^n \beta_2 \in L$ *para todo* $n \geq 1$.

Demostración. Debido a que L es un lenguaje regular, existe M un DFA tal que $L = L(M)$. Sea k el número de estados de M y α una cadena en L con $| \alpha | \geq k$. Entonces $\alpha = a_{i_1} \cdots a_{i_t}$ donde a_{i_j} símbolo de input ($1 \leq j \leq t$) y $t \geq k$. Sean $q_{i_0}, q_{i_1}, \ldots, q_{i_t}$ estados tal que, $q_{i_0} = q_0$ el estado inicial, q_{i_t} un estado final y

$$q_{i_0} \xrightarrow{a_{i_1}} q_{i_1} \xrightarrow{a_{i_2}} \cdots \xrightarrow{a_{i_{t-1}}} q_{i_{t-1}} \xrightarrow{a_{i_t}} q_{i_t}.$$

[1] Aquí suponemos, por comodidad notacional, que $s_j = q_j$ y $s_h = q_h$.

Es claro que, como $q_{i_0}, \ldots, q_{i_k}$ es una lista de $k+1$ estados y como el autómata tiene k estados, hay al menos dos estados que se repiten en la lista, digamos q_{i_r} y q_{i_s}, con $r < s \leq k$. Sean entonces $\beta_1 = a_{i_1} \cdots a_{i_{r-1}}$, $\gamma = a_{i_r} \cdots a_{i_{s-1}}$ y $\beta_2 = a_{i_s} \cdots a_{i_t}$. Luego, $\alpha = \beta_1\gamma\beta_2$, $|\,\gamma\,| > 0$ y $|\,\beta_1\gamma\,| \leq k$. Observemos que

$$q_{i_0} \xrightarrow{\beta_1} q_{i_r} \xrightarrow{\gamma} q_{i_r} \xrightarrow{\beta_2} q_{i_t},$$

en particular tenemos que $q_{i_r} \xrightarrow{\gamma} q_{i_r}$. Por lo tanto, se cumple que para $n > 0$,

$$q_{i_r} \xrightarrow{\gamma^n} q_{i_r} \quad \text{o equivalentemente} \quad q_{i_r} \xrightarrow{\gamma} q_{i_r} \xrightarrow{\gamma} \cdots \xrightarrow{\gamma} q_{i_r} \ (n\text{-veces}).$$

Por consiguiente, también vale

$$q_{i_0} \xrightarrow{\beta_1} q_{i_r} \xrightarrow{\gamma^n} q_{i_r} \xrightarrow{\beta_2} q_{i_t},$$

lo cual implica que $\beta_1\gamma^n\beta_2$ es una cadena aceptada por M y por lo tanto que pertenece a L. Esto prueba el Pumping Lemma. □

Como mencionamos arriba el Pumping Lemma es útil para demostrar que un lenguaje no es regular.

Ejemplo 2.23. Probemos que el lenguaje $L = \{a^n b^n : n > 1\}$ no es regular.

Demostración. Hagamos la prueba por el absurdo suponiendo que L es regular. Sea k como en el Pumping Lemma. Debemos elegir $\alpha \in L$ de tal forma de obtener una contradicción que nos lleve al absurdo. Elegimos $\alpha = a^k b^k$, por el Pumping Lemma existen cadenas β_1, β_2, γ con $|\,\beta_1\gamma\,| \leq k$, $|\,\gamma\,| > 0$ que satisfacen $\alpha = \beta_1\gamma\beta_2$ y $\beta_1\gamma^n\beta_2 \in L$ para todo $n \geq 1$. Como $|\,\beta_1\gamma\,| \leq k$, γ es de la forma a^s con $1 \leq s$, luego $\alpha = a^r a^s a^t b^k$ con $r+s+t = k$ y $s > 0$. Además $a^r a^{ns} a^t b^k \in L$ para $n \geq 1$. Esto es una contradicción pues como $r+ns+t > k$ para $n > 1$, entonces $a^r a^{ns} a^t b^k \notin L$ para $n > 1$. □

El Pumping Lemma es utilizado para detectar si un lenguaje no es regular, con la siguiente estrategia:

1. Seleccionar el lenguaje L que usted desea ver que no es regular.
2. Suponer que es regular y que por lo tanto existe un k como en el Pumping Lemma.
3. Seleccionar una cadena α en L de longitud mayor que k. La elección de la cadena no es arbitraria y depende del lenguaje dado.
4. Descomponer la cadena de acuerdo al Pumping Lemma y generar nuevas cadenas en L haciendo "inflación" en el centro (es decir las cadenas $\beta_1\gamma^n\beta_2$).
5. Verificar que las cadenas "infladas" no pertenecen a L. Esto genera una contradicción que vino de suponer que L era regular.

Observar que la estrategia anterior podría no servir, pero el hecho de que no podamos aplicar el Pumping Lemma a un lenguaje L no implica que este sea regular.

Ejemplo 2.24. Demostrar que el lenguaje $L = \{0^n 001^n \mid n \in \mathbb{N}\}$ no es un lenguaje regular.

Demostración. Supongamos que L es un lenguaje regular y sea k como en el Pumping Lemma. La cadena $\alpha = 0^k001^k$ es de L y por lo tanto es aceptada por el autómata. Por el Pumping Lemma $\alpha = \beta_1\gamma\beta_2$, con $|\gamma| > 0$ y $|\beta_1\gamma| \leq k$ y tal que $\beta_1\gamma^n\beta_2$ es aceptado por el autómata para todo $n > 0$. Como $|\beta_1\gamma| \leq k$, es claro que $\beta_1 = 0^r$, $\gamma = 0^s$ y $s \geq 1$. Es decir $\beta_1\gamma^n\beta_2 = 0^{k+1+(n-1)s}001^{k+1}$. Por lo tanto, $0^{k+1}0^{(n-1)s}001^{k+1}$ es una cadena de L para todo $n > 0$. Lo cual es un absurdo, pues esa cadena no pertenece a L para $n > 1$. □

Observar que en el ejemplo anterior hay cadenas de L que son de longitud mayor que k (por ejemplo $0^{k/2}001^{k/2}$, si k es par) que no sirven para demostrar, usando el Pumping Lemma, que el lenguaje no es regular. Es decir, se debe tener mucho cuidado en la elección de la cadena, pues la elección de una cadena incorrecta hará que la utilización del Pumping Lemma no nos sirva para demostrar que el lenguaje no es regular.

2.7 Ejercicios

1. Trace los diagramas de transición de los DFA dados por las siguientes reglas de transición.

   (a) Estados $\{q_0, q_1, q_2\}$; símbolos de input $\{a, b\}$, estado inicial q_0 y estado final q_0 también.

	a	b
q_0	q_1	q_0
q_1	q_2	q_0
q_2	q_0	q_2

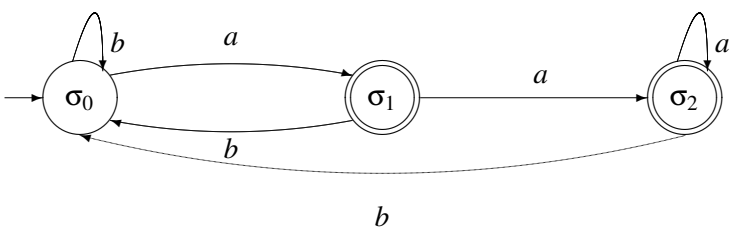
   (b) Estados $\{q_0, q_1, q_2\}$, símbolos de input $\{a, b\}$, estado inicial q_0 y estados finales q_0, q_2.

	a	b
q_0	q_1	q_1
q_1	q_0	q_2
q_2	q_0	q_1

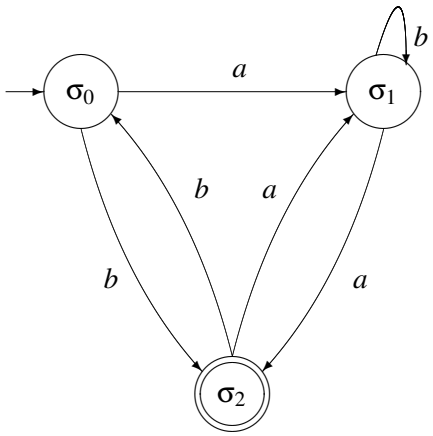
   (c) Estados $\{q_0, q_1, q_2, q_3\}$, símbolos de input $\{a, b, c\}$, estado inicial q_0 y estados finales q_1, q_2.

	a	b	c
q_0	q_1	q_0	q_2
q_1	q_0	q_3	q_0
q_2	q_3	q_2	q_0
q_3	q_1	q_0	q_1

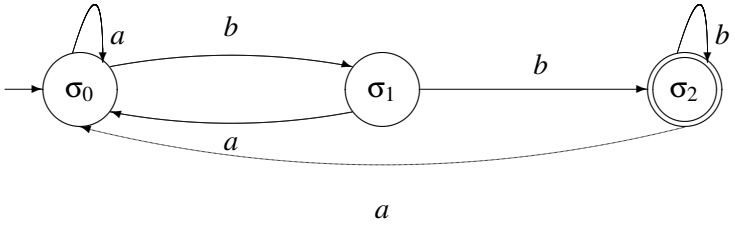
2. Determine si la cadena *abbaa* es aceptada por el DFA definido por el siguiente diagrama



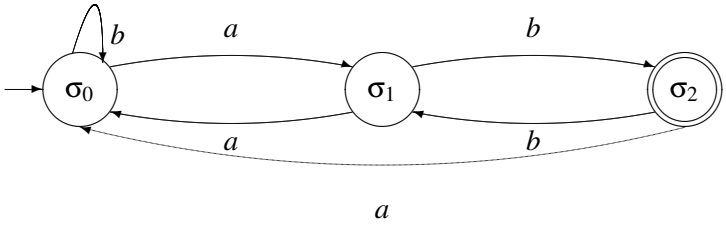
3. Determine si la cadena *abbaa* es aceptada por el DFA definido por el siguiente diagrama



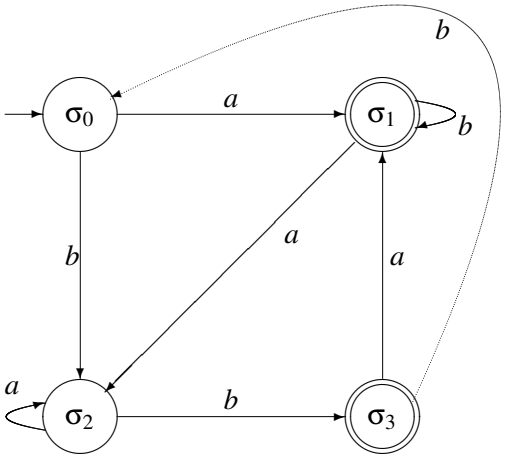
4. Determine si la cadena *aabaabb* es aceptada por el DFA definido por el siguiente diagrama



5. Determine si la cadena *aaabbbaab* es aceptada por el DFA definido por el diagrama:



6. Determine si la cadena *aaababbab* es aceptada por el DFA definido por el diagrama:



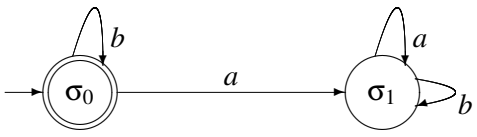
7. Pruebe que una cadena α en el alfabeto $\{a,b\}$ es aceptada por el autómata del Ejercicio 2 si y sólo si α termina en a.

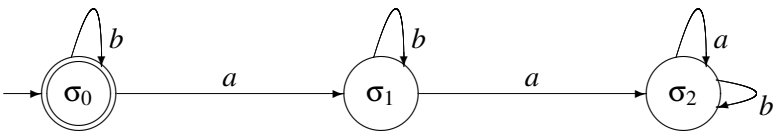
8. Pruebe que una cadena α en el alfabeto $\{a,b\}$ es aceptada por el autómata del Ejercicio 4 si y sólo si α termina en bb.

9. Trace el diagrama de transición del DFA que acepte el conjunto de cadenas en al alfabeto $\{a,b\}$ dado en cada uno de los siguientes items.

   (a) Cadenas con un número par de letras a.
   (b) Cadenas con exactamente una letra b.
   (c) Cadenas con al menos una letra b.
   (d) Cadenas con exactamente dos letras a.
   (e) Cadenas con al menos dos letras a.
   (f) Cadenas que contengan m letras a, donde m es un múltiplo de 3.
   (g) Cadenas que empiecen con baa.
   (h) Cadenas que contengan $abba$.
   (i) Cadenas donde toda letra b esté seguida de una letra a.
   (j) Cadenas que terminen con aba
   (k) Cadenas que empiecen con ab y terminen con aba

10. Demuestre que los siguientes DFA son equivalentes.





11. Sea L un conjunto finito de cadenas no vacías sobre $\{a,b\}$. Demuestre que hay un DFA que acepta L.

12. Hallar un autómata finito determinístico que acepte exactamente el lenguaje de las cadenas sobre el alfabeto $\Sigma = \{0,1\}$ que tienen una cantidad de 1's que es múltiplo de 3 y un número par de ceros.

13. Dado el DFA con alfabeto $\Sigma = \{0,\ 1\}$, estado inicial q_0 y estados finales q_1 y q_f y con la siguiente tabla de transición:

	0	1
q_0	q_2	q_1
q_1	q_0	q_f
q_2	q_f	q_2
q_f	q_0	q_1

    (a) Hacer el diagrama de transición correspondiente,

    (b) Verificar si las cadenas $w_1 = 01011$ y $w_2 = 01000$ son aceptadas o no por el autómata. Describa paso a paso.

14. Dado el DFA con alfabeto $\Sigma = \{a,\ b\}$, estado inicial q_0 y estados finales q_1 y q_f y con la siguiente tabla de transición:

	a	b
q_0	q_1	q_2
q_1	q_2	q_1
q_2	q_f	q_0
q_f	q_f	q_2

    (a) Hacer el diagrama de transición correspondiente,

    (b) Verificar si las cadenas $w_1 = baaba$ y $w_2 = aabaa$ son aceptadas o no por el autómata. Describa paso a paso.

15. Construir un autómata finito determinístico con alfabeto $\Sigma = \{a,\ b\}$ que acepte cadenas que empiecen con *ab* y terminen con *ba*.

16. Hallar un autómata finito determinístico que acepte exactamente el lenguaje de las cadenas sobre el alfabeto $\Sigma = \{0,1\}$ que tienen una cantidad par de 1's y el número de 0's es múltiplo de 3.

17. Hallar un autómata finito determinístico que acepte exactamente el lenguaje de las cadenas sobre el alfabeto $\Sigma = \{0,1\}$ que tiene un número par de ceros y terminan con dos (o más) unos.

18. Hallar un autómata finito determinístico que acepte exactamente el lenguaje de las cadenas sobre el alfabeto $\Sigma = \{0,1\}$ que no empiezan y terminan con el mismo símbolo.

19. Trace los diagramas de transición de los autómatas no determinísticos dados por las siguientes reglas de transición.

    (a) Estados $\{q_0, q_1, q_2\}$; símbolos de input $\{a,b\}$, estado inicial q_0 y estado final q_0 también y reglas de transición dadas por la siguiente tabla.

	a	b
q_0	$\emptyset$	$\{q_1, q_2\}$
q_1	$\{q_2\}$	$\{q_0, q_1\}$
q_2	$\{q_0\}$	$\emptyset$

    (b) Estados $\{q_0, q_1, q_2\}$, símbolos de input $\{a,b\}$, estado inicial q_0 y estados finales q_0, q_1 y reglas de transición dadas por la siguiente tabla.

	a	b
q_0	$\{q_1\}$	$\{q_0, q_1\}$
q_1	$\emptyset$	$\{q_2\}$
q_2	$\{q_1\}$	$\emptyset$

    (c) Estados $\{q_0, q_1, q_2, q_3\}$, símbolos de input $\{a,b\}$, estado inicial q_0 y estado final q_1 y reglas de transición dadas por la siguiente tabla.

	a	b
q_0	$\emptyset$	$\{q_3\}$
q_1	$\{q_1, q_2\}$	$\{q_3\}$
q_2	$\emptyset$	$\{q_0, q_1, q_3\}$
q_3	$\emptyset$	$\emptyset$

    (d) Estados $\{q_0, q_1, q_2\}$, símbolos de input $\{a,b,c\}$, estado inicial q_0 y estados finales q_0, q_1 y reglas de transición dadas por la siguiente tabla.

	a	b	c
q_0	$\{q_1\}$	$\emptyset$	$\emptyset$
q_1	$\{q_0\}$	$\{q_2\}$	$\{q_0, q_2\}$
q_2	$\{q_0, q_1, q_2\}$	$\{q_0\}$	$\{q_0\}$

    (e) Estados $\{q_0, q_1, q_2, q_3\}$, símbolos de input $\{a,b,c\}$, estado inicial q_0 y estados finales q_0, q_3 y reglas de transición dadas por la siguiente tabla.

	a	b	c
q_0	$\emptyset$	$\{q_3\}$	$\emptyset$
q_1	$\{q_1, q_2\}$	$\{q_3\}$	$\emptyset$
q_2	$\emptyset$	$\{q_0, q_1, q_3\}$	$\{q_0, q_2\}$
q_3	$\emptyset$	$\emptyset$	$\{q_0\}$

20. Sea M_1 es autómata correspondiente al del Ejercicio 2 y M_2 el autómata correspondiente al del Ejercicio 4. Hallar diagramas de transición de autómatas cuyos lenguajes sea $L(M_1) \cup L(M_2)$ y $L(M_1) \cap L(M_2)$ respectivamente.

21. Para cada autómata que se muestra en las Figuras 13, 14, 15, 16 y 17 establezca el conjunto de estados Q, el conjunto de símbolos de input Σ, el estado inicial q_0, el conjunto de estados finales $\mathcal{F}$.

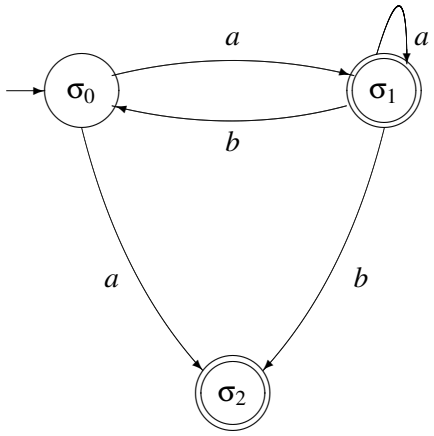


Figura 13:

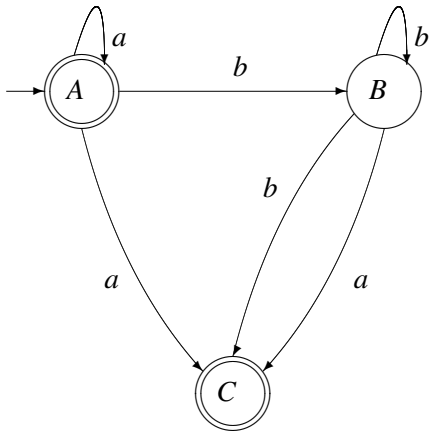


Figura 14:

22. ¿Las cadenas *bbabbb* y *bbabab* son aceptadas por el autómata de la Fig. 18? Pruebe sus respuestas.

23. Demuestre que una cadena α es aceptada por el autómata de la Fig. 18 si y sólo si α tiene una sola letra *a* y termina en *b*.

24. ¿Las cadenas *aaabba* y *aaaab* son aceptadas por el autómata de la Fig. 19? Pruebe sus respuestas.

25. Determine el lenguaje aceptado por los autómatas de las Fig. 20, 21 y 22. Encuentre un autómata determinístico con el mismo lenguaje en cada caso.

26. Caracterice los arreglos aceptados por el autómata de la Fig. 19.

27. Verifique que las cadenas aceptadas por el autómata de la Fig. 15 son las cadenas en $\{a,b\}$ que terminan en *bab*.

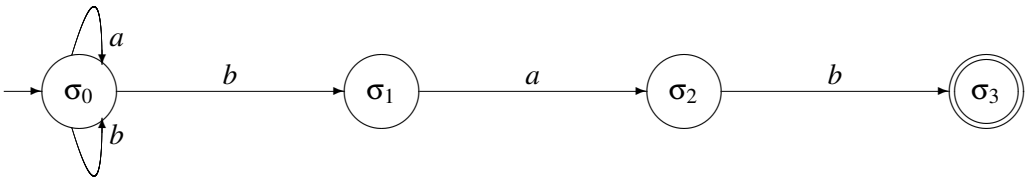


Figura 15:

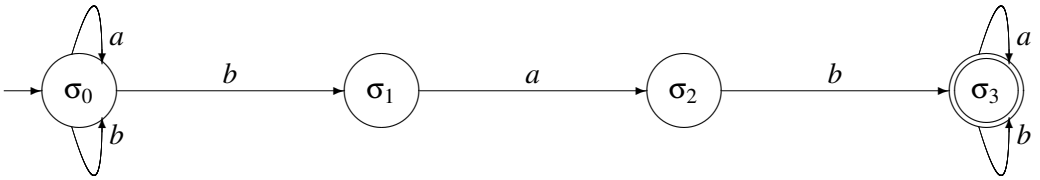


Figura 16:

28. Caracterice las cadenas aceptadas por los autómatas del Ejercicio 19.

29. Caracterice las cadenas aceptadas por los autómatas de las Figuras 13, 14, 16y 17.

30. Diseñe autómatas no determinísticos que acepten las cadenas no nulas sobre $\{a,b\}$ y que tengan las siguientes propiedades.

    (a) Comienzan con *abb* o con *ba*.

    (b) Terminan con *abb* o con *ba*.

    (c) Contienen *abb* o *ba*.

    (d) Contienen *bab* y *bb*.

    (e) Toda *b* se encuentra entre dos *a*.

    (f) Comienzan con *abb* y terminan con *ab*.

    (g) Comienzan con *ab* pero no terminan con *ab*.

    (h) No contienen *ba* o *bbb*.

    (i) No contienen *abba* o *bbb*

31. Caracterice el lenguaje aceptado por cada uno de los autómatas del Ejercicio 1.

32. Caracterice el lenguaje aceptado por los autómatas definidos en los Ejercicios 5 y 6.

33. Describir en palabras los conjuntos denotados por las siguientes expresiones regulares.

    (a) $(11+0)^*(00+1)^*$

    (b) $(1+01+001)^*(\varepsilon+0+00)$

34. Encontrar expresiones regulares en el alfabeto $\{a,b\}$ que describan los siguientes conjuntos:

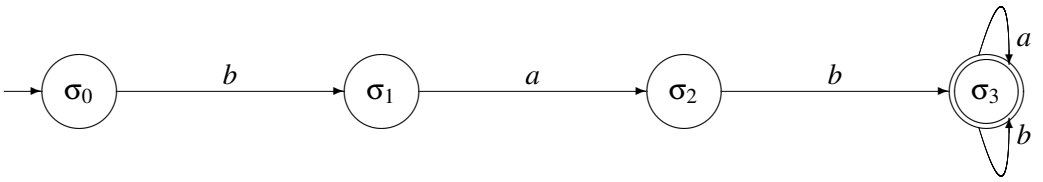


Figura 17:

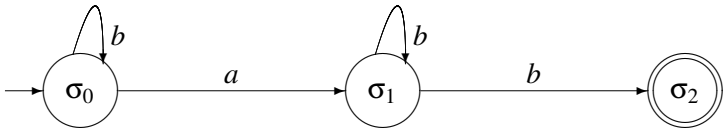


Figura 18:

(a) Cadenas con un número par de letras a.

(b) Cadenas con exactamente una letra b.

(c) Cadenas con al menos una letra b.

(d) Cadenas con exactamente dos letras a.

(e) Cadenas con al menos dos letras a.

(f) Cadenas que contengan m letras a, donde m es un múltiplo de 3.

(g) Cadenas que empiecen con baa.

(h) Cadenas que contengan $abba$.

(i) Cadenas donde toda letra b esté seguida de una letra a.

(j) Cadenas que terminen con aba

(k) Cadenas que empiecen con ab y terminen con aba

35. Construir autómatas finitos cuyo lenguaje sea dado por las siguientes expresiones regulares.

(a) $10+(0+11)0^*1$

(b) $01[((10)^*+111)^*+0]^*1$

(c) $((0+1)(0+1))^*+((0+1)(0+1)(0+1))^*$

36. Encontrar expresiones regulares en el alfabeto $\{a,b\}$ que describan los siguientes conjuntos:

(a) Comienzan con abb o con ba.

(b) Terminan con abb o con ba.

(c) Contienen abb o ba.

(d) Contienen bab y bb.

(e) Toda b se encuentra entre dos a.

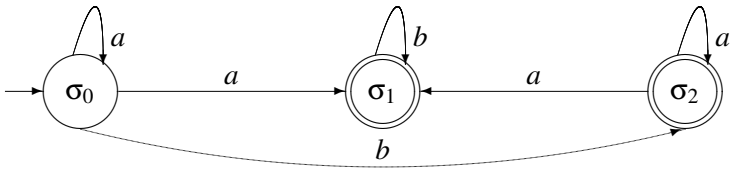


Figura 19:

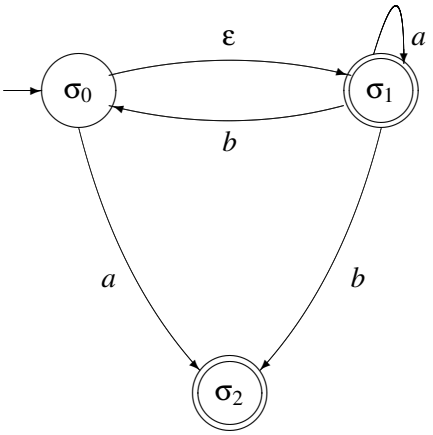


Figura 20:

(f) Comienzan con *abb* y terminan con *ab*.

(g) Comienzan con *ab* pero no terminan con *ab*.

(h) No contienen *ba* o *bbb*.

(i) No contienen *abba* o *bbb*

37. (a) Dibuje un autómata finito determinístico que acepte exactamente el lenguaje de las cadenas de 0's y 1's que no tienen más de tres ceros consecutivos.

(b) Escriba la expresión regular correspondiente.

38. Verificar si los siguientes lenguajes son o no regulares:

(a) $L_1 = \{0^n 1^n 1^m 0^m \mid n \in \mathbb{N}\}$.

(b) $L_2 = \{1^n 0^n 1^m 0^m \mid n,\ m \in \mathbb{N}\}$.

(c) $L_3 = \{0^n 110^n, n \geq 0\}$.

(d) $L_4 = \{ab^{n+2}baa^n\}$.

(e) $L_5 = \{110^n 110^n, n \geq 0\}$,

3 Gramáticas y lenguajes

En este capítulo introduciremos las gramáticas libres de contexto y regulares y los lenguajes que ellas describen (los lenguajes libres de contexto y regulares). Los lenguajes libres de contexto son de gran importancia en la definición de los lenguajes de programación, entre otra aplicaciones. Como

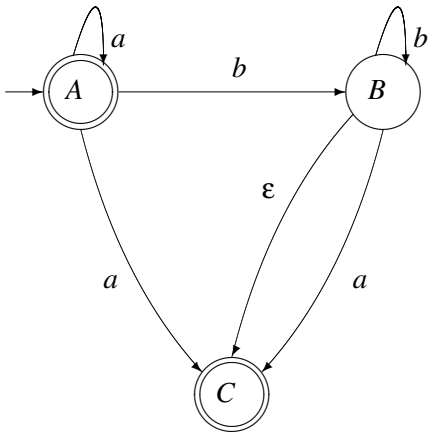


Figura 21:

un ejemplo, los lenguajes libres de contextos son útiles para describir expresiones aritméticas con paréntesis balanceados y para describir estructuras de bloque en los lenguajes de programación.

3.1 Definiciones básicas y ejemplos

Una *gramática libre de contexto* es un conjunto finito de *variables* (también llamadas *no terminales* o *categorías sintácticas*), un conjunto de símbolos llamados *terminales* y un conjunto de reglas llamadas *producciones* que transforman una variable en una cadena formada por variables y terminales.

La motivación original de las gramáticas libres de contexto fue la descripción de los lenguajes naturales. Podemos escribir reglas como:

$$\begin{aligned}
< \text{oración} > &\rightarrow < \text{sujeto} >< \text{predicado} > \\
< \text{sujeto} > &\rightarrow < \text{sujeto} >< \text{adjetivo} > \\
< \text{sujeto} > &\rightarrow \text{el perro} \\
< \text{adjetivo} > &\rightarrow \text{pequeño} \\
< \text{adjetivo} > &\rightarrow \text{grande} \\
< \text{adjetivo} > &\rightarrow \text{bueno} \\
< \text{predicado} > &\rightarrow < \text{verbo} >< \text{adjetivo} > \\
< \text{verbo} > &\rightarrow \text{es}
\end{aligned}$$

donde las categorías sintácticas son denotadas por los $<, >$ y los terminales son “el perro", “pequeño", “grande", “bueno" y “es". Ahora deseamos ver que frases se pueden armar con terminales partiendo de la variable $<$ oración $>$ y usando las reglas de producción. Por ejemplo

$$\begin{aligned}
< \text{oración} > &\Rightarrow < \text{sujeto} >< \text{predicado} > \\
&\Rightarrow < \text{sujeto} >< \text{adjetivo} >< \text{predicado} > \\
&\Rightarrow \text{el perro} < \text{adjetivo} >< \text{predicado} > \\
&\Rightarrow \text{el perro} < \text{adjetivo} >< \text{verbo} >< \text{adjetivo} > \\
&\Rightarrow \text{el perro bueno} < \text{verbo} >< \text{adjetivo} > \\
&\Rightarrow \text{el perro bueno es} < \text{adjetivo} > \\
&\Rightarrow \text{el perro bueno es grande}
\end{aligned}$$

El símbolo $\Rightarrow$ denota la acción de derivar, es decir, reemplazar una variable por el lado derecho de una producción para la variable.

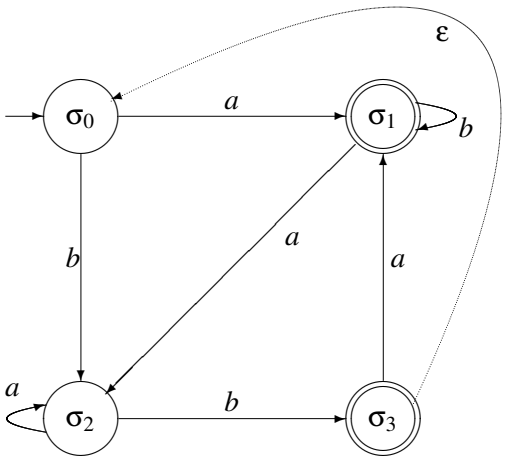


Figura 22:

Pese a su origen, las gramáticas libres de contexto no son adecuadas para la descripción de lenguajes naturales, una razón importante de este hecho es que es necesaria información semántica, además de la sintáctica, para poder construir frases correctas en castellano. Por ejemplo, aunque las reglas de producción anteriores son aparentemente “naturales" también podemos armar la siguiente frase “el perro grande es pequeño".

Ahora, formalizaremos los conceptos expresados previamente.

Definición 3.1. Una *gramática libre de contexto (CFG)* es una 5-upla $G = (V,T,P,S)$ tal que

1. V es un conjunto finito. Los elementos de V son llamados *variables* o *no terminales* o *categorías sintácticas*.

2. T es un conjunto finito disjunto con V. Los elementos de T son llamados *terminales*.

3. S es una variable especial que llamaremos el *símbolo inicial*.

4. P es un conjunto finito P, cuyos elementos son llamados *producciones*, tal que cada producción es de la forma $A \to \alpha$, donde A es una variable y α es una cadena formada por variables y terminales (que puede ser vacía), es decir $\alpha \in (V \cup T)^*$.

En el futuro usaremos la siguiente notación: si $A \to \alpha_1, A \to \alpha_2, \ldots, A \to \alpha_n$ son producciones para la variable A en alguna gramática, podemos expresar esto como

$$A \to \alpha_1 \mid \alpha_2 \mid \cdots \mid \alpha_n.$$

Definición 3.2. Sea $G = (V,T,P,S)$ una gramática. Si $A \to \alpha$ es una producción y xAy es una cadena formada por variables y terminales, se dice que $x\alpha y$ *se deriva directamente* de xAy y se escribe

$$xAy \Rightarrow a\alpha y.$$

Si $\alpha_1, \ldots, \alpha_n$ cadenas y $\alpha_1 \Rightarrow \alpha_2, \alpha_2 \Rightarrow \alpha_3, \ldots, \alpha_{n-1} \Rightarrow \alpha_n$ derivaciones directas, entonces decimos que α_n se deriva de α_1 y se escribe

$$\alpha_1 \Rightarrow \alpha_n.$$

La secuencia

$$\alpha_1 \Rightarrow \alpha_2 \Rightarrow \alpha_3 \Rightarrow \cdots \Rightarrow \alpha_{n-1} \Rightarrow \alpha_n$$

se llama *derivación* de α_1 a α_n. Por convención, cualquier cadena se deriva de si misma.

Finalmente, el *lenguaje generado* por G consiste de todas las cadenas de elementos de T que se derivan de S. Se denota $L(G)$. Los lenguajes generados por las gramáticas libres de contexto se llaman *lenguajes libres de contexto*.

Ejemplo 3.1. Consideremos la gramática con una variable E, símbolos terminales $+$, $*$, $($, $)$ y id, y producciones

$$E \to E+E \mid E*E \mid (E) \mid id \quad .$$

Entonces $(id+id)*id$ se deriva de E, pues

$$\begin{aligned} E &\Rightarrow E*E \\ &\Rightarrow (E)*E \\ &\Rightarrow (E)*id \\ &\Rightarrow (E+E)*id \\ &\Rightarrow (E+id)*id \\ &\Rightarrow (id+id)*id \end{aligned}$$

La primera línea se obtiene usando la producción $E \to E*E$, la segunda se obtiene reemplazando la primera E por el lado derecho de la producción $E \to (E)$. Las restantes líneas se obtienen de aplicar sucesivamente las producciones $E \to id$, $E \to E+E$, $E \to id$, y $E \to id$.

Ejemplo 3.2. Consideremos la gramática con una variable S , símbolos terminales a y b, y producciones $S \to aSb \mid ab$. Veamos que el lenguaje generado por esta gramática es el de todas las cadenas en el alfabeto $\{a,b\}$ que son de la forma $a^n b^n$ con $n > 0$.

Demostración. Hay esencialmente una única forma de obtener una cadena con símbolos terminales: primero aplicar repetidas veces la producción $S \to aSb$ y luego terminar con la producción $S \to ab$. Es decir que las derivaciones son del tipo

$$S \Rightarrow aSb \Rightarrow a^2Sb^2 \Rightarrow \cdots \Rightarrow a^{n-1}Sb^{n-1} \Rightarrow a^n b^n.$$

Entonces es claro que las palabras generadas por el lenguaje son de la forma $a^n b^n$ y obviamente toda cadena de la forma $a^n b^n$ se obtiene haciendo la derivación escrita más arriba. □

Ejemplo 3.3. Consideremos la gramática con una variable S, símbolos terminales a y b, y producciones $S \to aSa \mid bSb \mid a \mid b \mid \varepsilon$. Veamos que el lenguaje generado por esta gramática es el de todas las cadenas en el alfabeto $\{a,b\}$ que son capicúa.

Demostración. Por definición, una palabra capicúa es o bien de la forma $x_1x_2\ldots x_nx_n\ldots x_2x_1$ o de la forma $x_1x_2\ldots x_{n-1}x_nx_{n-1}\ldots x_2x_1$, con x_i igual a a o b. Usando sucesivamente las producciones $S \to x_1Sx_1$, $S \to x_2Sx_2$, …, $S \to x_{n-1}Sx_{n-1}$, obtenemos la derivación

$$S \Rightarrow x_1Sx_1 \Rightarrow x_1x_2Sx_2x_1 \Rightarrow \cdots \Rightarrow x_1x_2\ldots x_{n-1}Sx_{n-1}\ldots x_2x_1,$$

si ahora usamos las producción $S \to x_n S x_n$ y luego $S \to \varepsilon$ obtenemos $S \Rightarrow x_1 x_2 \ldots x_n x_n \ldots x_2 x_1$. Por otro lado si aplicamos la producción $S \to x_n$ a $x_1 x_2 \ldots x_{n-1} S x_{n-1} \ldots x_2 x_1$ obtenemos la derivación $S \Rightarrow x_1 x_2 \ldots x_{n-1} S x_{n-1} \ldots x_2 x_1 \Rightarrow x_1 x_2 \ldots x_{n-1} x_n x_{n-1} \ldots x_2 x_1$.

Veamos por inducción la recíproca, es decir que una cadena del lenguaje generado por la gramática es capicúa: la hipótesis inductiva es $S \Rightarrow \alpha S \beta$, entonces $\alpha = x_1 x_2 \ldots x_k$ y $\beta = x_k \ldots x_2 x_1$, con x_i igual a a o b. La inducción se hace sobre la longitud de α. Si la longitud de α es 1 el resultado es trivial. Supongamos que tenemos probado el resultado para cadenas de longitud $k-1$. Sea $S \Rightarrow \alpha S \beta$, con $|\alpha| = k$. Sea $\alpha = x_1 x_2 \ldots x_k$, es claro que la última producción que se usó es $S \to x_k S x_k$, luego tenemos $S \Rightarrow \alpha' S \beta' \Rightarrow \alpha' x_k S x_k \beta' = \alpha S \beta$. Claramente $\alpha' = x_1 x_2 \ldots x_{k-1}$ tiene longitud $k-1$, y entonces por hipótesis inductiva tenemos que $\beta' = x_{k-1} \ldots x_2 x_1$ y por consiguiente $\beta = x_k \ldots x_2 x_1$. Sea ahora una cadena α de símbolos terminales, tal que $S \Rightarrow \alpha$, entonces la derivación es $S \Rightarrow \alpha_1 S \alpha_2 \Rightarrow \alpha_1 x \alpha_2 = \alpha$, con x igual a a, b o ε. Por lo visto más arriba $\alpha_1 x \alpha_2 = \alpha$ es capicúa. □

Una manera alternativa de enunciar las producciones de una gramática es por el empleo de la *forma normal de Backus-Naur* o *BNF*. En una BNF los símbolos no terminales empiezan con "<" y terminan con ">". Las producción $A \to \alpha$ se expresa $A ::= \alpha$.

Ejemplo 3.4. Un entero es una cadena que consiste de un símbolo opcional ($+$ o bien $-$) seguido por un entero o cadena de dígitos (del 0 al 9). La siguiente gramática (escrita en notación BNF) genera todos los enteros.

$$\begin{aligned}
< \text{dígito} > &::= 0 \mid 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9 \\
< \text{entero} > &::= < \text{entero con signo} > \mid < \text{entero sin signo} > \\
< \text{entero con signo} > &::= + < \text{entero sin signo} > \mid - < \text{entero sin signo} > \\
< \text{entero sin signo} > &::= < \text{dígito} > \mid < \text{dígito} > < \text{entero sin signo} >
\end{aligned}$$

El símbolo inicial es $<$ entero $>$, las otras variables son $<$ entero con signo $>$, $<$ entero sin signo $>$, $<$ dígito $>$, los símbolos terminales son $0,1,2,3,4,5,6,7,8,9,+,-$.

Por ejemplo la derivación del entero -452 es

$$\begin{aligned}
< \text{entero} > &\Rightarrow < \text{entero con signo} > \\
&\Rightarrow - < \text{entero sin signo} > \\
&\Rightarrow - < \text{dígito} > < \text{entero sin signo} > \\
&\Rightarrow - < \text{dígito} > < \text{dígito} > < \text{entero sin signo} > \\
&\Rightarrow - < \text{dígito} > < \text{dígito} > < \text{dígito} > \\
&\Rightarrow -4 < \text{dígito} > < \text{dígito} > \\
&\Rightarrow -45 < \text{dígito} > \\
&\Rightarrow -452
\end{aligned}$$

Claramente el lenguaje asociado a esta gramática es el de todas las cadenas que empiezan con $+$ o $-$ y sigue una sucesión de dígitos o las cadenas que son una sucesión de dígitos.

La sintaxis de algunos lenguajes de computación de alto nivel, como Pascal, C o FORTRAN, pueden expresarse en BNF.

3.2 Formas normales de Chomsky y Greibach

Dada una gramática libre de contexto G es posible hallar gramáticas con producciones de cierto tipo tal que el lenguaje generado por estas gramáticas sea del mismo que el generado por G.

Definición 3.3. Sea $G = (V,T,P,S)$ una CFG. Diremos que G está en la *forma normal de Chomsky* si todas las producciones son de la forma

$$A \to a \qquad \text{o} \qquad A \to BC,$$

donde $A,B,C \in V$ y $a \in T$. Es decir, si el lado derecho de todas las producciones es o bien un símbolo terminal o dos variables. Diremos que G está en la *forma normal de Greibach* si todas las producciones son de la forma

$$A \to a\alpha,$$

donde $A \in V$, $a \in T$ y $\alpha \in V^*$. Es decir, si el lado derecho de todas las producciones es un símbolo terminal seguido de ninguna o varias variables.

Dada una gramática G es posible hallar gramáticas G' y G'' en forma normal de Chomsky y Greibach, respetivamente, tal que el lenguaje generado por G sea el mismo que el generado por G' o G''.

En lo que resta de la sección veremos como dada una gramática G libre de contexto podremos encontrar en forma algorítmica una gramática equivalente en la forma de Chomsky. El procedimiento consta de los siguientes pasos:

1. Eliminar ε-producciones ($A \to \varepsilon$).
2. Eliminar variables que no llevan a cadenas de símbolos terminales.
3. Eliminar símbolos (variables y terminales) que no pueden ser alcanzados.
4. Eliminar producciones unitarias ($A \to B$).
5. Forma normal de Chomsky.

Comencemos a hacer el procedimiento:

(1) Eliminar ε-producciones. Con este procedimiento obtendremos una gramática con lenguaje $L(G) - \{\varepsilon\}$. Si el lenguaje original contenía a ε, entonces al final agregaremos la producción $S \to \varepsilon$. El procedimiento es como sigue. Primero sea $N = \emptyset$ y ahora:

(a) si $A \to \varepsilon$ es producción, entonces agregar A a N.

(b) Iteramos el siguiente paso hasta que N se estabilice: si $A \to \alpha$ es producción con $\alpha \in N^+$, entonces agregar A a N.

Es claro que el proceso (b) termina debido a que $N \subset V$.

El nuevo conjunto P' de producciones es el siguiente: si $A \to X_1X_2\ldots X_k$ es una producción en P donde $X_i \in V \cup T$, entonces $A \to \alpha_1\alpha_2\ldots\alpha_k$ está en P' si

1. $\alpha_i = X_i$ si $X_i \notin N$,
2. $\alpha_i = X_i$ o $\alpha_i = \varepsilon$ si $X_i \in N$,
3. $\alpha_1\alpha_2\ldots\alpha_k \neq \varepsilon$.

Es decir, si partimos de $P' = \emptyset$, para toda $A \to \alpha$ en P, se agregan a P' las producciones $A \to \alpha'$ donde $\alpha' \neq \varepsilon$ y α' es cualquier combinación que resulte de eliminar algunas variables en N de α . Observar que si S está en N, entonces ε es aceptado por el lenguaje original.

(2) Eliminar variables que no llevan a cadenas de símbolos terminales. Debemos reemplazar el conjunto de variables originales por el conjunto V' que se construye con cierto número de iteraciones: el primer V' se obtiene de agregar todas las variables que figuren del lado izquierdo de producciones del tipo $A \to \beta$ donde $\beta \in T^+$. Luego se itera el siguiente procedimiento hasta que se estabilice V'

(a) el nuevo V' se obtiene agregando todas las variables que figuren del lado izquierdo de producciones del tipo $A \to \beta$ donde $\beta \in$, es decir cuando β es una cadena donde las variables que aparecen son de V'. En pseudocódigo sería

$$V' := V' \cup \{A : A \to \beta, \beta \in (T \cup V')^+\}$$

Terminada la iteración nuestro nuevo conjunto de variables V será igual al último V'. Si S no está en V', entonces es fácil ver que el lenguaje generado por G es vacío.

(3) Eliminar símbolos que no pueden ser alcanzados. Sea $V' = \{S\}$ y $T' = \emptyset$. Iterar de la siguiente forma hasta que V' y T' se estabilicen:

(a) si $A \to \alpha$ es una producción con $A \in V'$, entonces agregar a V' todas las variables que se encuentren en α y agregar a T' todos los símbolos no terminales que se encuentren en α. En pseudocódigo sería

$$V' := V' \cup \{B \in V : A \to \alpha B \beta,\ A \in V',\ \alpha, \beta \in (T \cup V)^*\}$$
$$T' := T' \cup \{a \in T : A \to \alpha a \beta,\ A \in V',\ \alpha, \beta \in (T \cup V)^*\}$$

La iteración finaliza puesto que $V' \subset V$ y $T' \subset T$. Nuestros nuevos V y T serán V' y T', respectivamente, obtenidos por las iteraciones de más arriba.

(4) Eliminar producciones unitarias. Las producciones unitarias son aquellas de la forma $A \to B$ con $A, B \in V$. El proceso de eliminar producciones unitarias es sencillo: recorremos el conjunto de producciones y cada vez que encontramos una producción del tipo $A \to B$ la eliminamos y todas las producciones $B \to \gamma$ las cambiamos por $A \to \gamma$.

(5) Forma normal de Chomsky. En este caso suponemos que hemos completado los procedimientos (1)…(4). Si $A \to \alpha$ es una producción la *longitud de* $A \to \alpha$ se define como la longitud de la cadena α. Para hacer la forma normal de Chomsky procederemos de la siguiente manera:

(a) por cada símbolo terminal a se agrega una nueva variable A y una producción $A \to a$. Ahora, en cada producción de longitud mayor que 1 se reemplaza cada símbolo terminal a por la nueva variable A. Después de hacer esto todas las producciones son del tipo $A \to a$, con a terminal o del tipo $A \to A_1 \ldots A_n$ con $A_1, \ldots, A_n$ variables.

(b) En cada producción de longitud mayor que dos $A \to A_1 A_2 \ldots A_n$ se reemplaza $A_1 A_2$ por una nueva variable B y se agrega la producción $B \to A_1 A_2$. Este procedimiento se itera hasta que no haya producciones de longitud mayor que 2. Es claro que la iteración termina debido as que en cada paso se disminuye la longitud máxima que pueden tener las producciones.

Ejemplo 3.5. Sea $G = (V, T, P, S)$ con $V = \{S, A\}$, $T = \{a, b\}$ y producciones

$$S \to aAS \mid a \qquad A \to SbA \mid SS \mid ba$$

Ejemplo 3.6. Sea $G = (V,T,P,S)$ con $V = \{S,A\}$, $T = \{a,b\}$ y producciones

$$S \to aAS \mid a \qquad A \to SbA \mid SS \mid ba$$

Encontrar una gramática en la forma normal de Chomsky que genere el lenguaje $L(G)$.

No es difícil verificar que esta gramática no necesita que realicemos los procedimientos (1), (2), (3) y (4). Debemos entonces aplicar el procedimiento (5):

(a) Primero agregamos una variable por cada símbolo terminal, X por a e Y por b. También agregamos las producciones $X \to a$ e $Y \to b$. En las producciones de longitud mayor que 1 reemplazamos a por X y b por Y. Obtenemos entonces la gramática $G = (V,T,P,S)$ con $V = \{S,A,X,Y\}$, $T = \{a,b\}$ y producciones

$$\begin{array}{lll} S \to XAS \mid a & \qquad & A \to SYX \mid SS \mid YX \\ X \to a & & T \to b \end{array}$$

(b) Reemplazamos en $S \to XAS$ a XA por U (una nueva variable) y agregamos la producción $U \to XA$. Reemplazamos en $A \to SYX$ a SY por W (una nueva variable) y agregamos la producción $W \to SY$. Obtenemos la gramática $G = (V,T,P,S)$ con $V = \{S,A,X,Y,U,W\}$, $T = \{a,b\}$ y producciones

$$\begin{array}{lll} S \to US \mid a & \qquad & A \to WX \mid SS \mid YX \\ X \to a & & T \to b \\ U \to XA & & W \to SY \end{array}$$

La formal normal de Greibach se puede obtener a partir de la forma normal de Chomsky, pero es mucho más complicado hacerlo y queda fuera de los alcances de este escrito.

3.3 Gramáticas regulares

Los lenguajes definidos por autómatas finitos o, lo que es lo mismo, por expresiones regulares, también pueden ser vistos como los lenguajes que generan ciertas gramáticas, llamadas gramáticas regulares.

Definición 3.4. Sea G una gramática tal que toda producción es de la forma

$$A \to aB \quad \text{o bien} \quad A \to \varepsilon,$$

donde A, B son variables y a es terminal. Entonces diremos que G es una *gramática regular*.

El lenguaje generado por un una gramática regular será llamado *lenguaje regular*.

Observemos primero que podemos considerar permitidas, en las gramáticas regulares, las producciones del tipo $A \to a$ pues se pueden obtener por composición de las permitidas en la definición. Una observación importante es que cualquier cadena con símbolos terminales y no terminales que se obtiene por derivación a partir del símbolo inicial, tiene a lo sumo una variable y ésta se ubica en el extremo derecho de la cadena. Esto implica que cuando se dice que se usa ciertas producciones para realizar una derivación, no haya ambigüedad en la forma en que hay que aplicarla.

Ejemplo 3.7. Encontremos una gramática que genere el lenguaje asociado a la expresión regular a^*b^*. Es claro que el lenguaje asociado a a^*b^* es $L = \{a^n b^m : 0 \leq n, m\}$. Sea G gramática con una variable S, símbolos terminales a y b y las siguientes producciones:

$$\begin{aligned} S &\to aS \mid aT \mid bT \mid \varepsilon, \\ T &\to bT \mid \varepsilon \end{aligned}$$

entonces el lenguaje que genera G es L.

Demostración. Primero veamos que $L \subset L(G)$: sea $a^n b^m$ en L, si $n, m > 0$ usando la producción $S \to aS$, $(n-1)$-veces tenemos $S \Rightarrow a^{n-1}S$, luego usando la producción $S \to aT$ obtenemos $S \Rightarrow a^n T$, después usamos $S \to bT$ m-veces y obtenemos $S \Rightarrow a^n b^m T$, finalmente usando la producción $T \to \varepsilon$ tenemos la derivación $S \Rightarrow a^n b^m$. En el caso que $n = 0$ y $m > 0$ se hace $S \Rightarrow bT \Rightarrow b^m T \Rightarrow b^m$. El caso $n > 0$ y $m = 0$ es similar. Cuando $n = m = 0$, hacemos $S \Rightarrow \varepsilon$ usando la producción $S \to \varepsilon$.

Veamos ahora que $L(G) \subset L$: observemos que si usamos una derivación en esta gramática que agrega un b, entonces no se puede agregar más un a, como (por la definición de gramáticas regulares) los agregados de símbolos terminales sólo se pueden hacer a la derecha, es claro que las a's estarán todas a la izquierda de cualquier b.

□

Ejemplo 3.8. Encontremos una gramática regular G, cuyo lenguaje sea el lenguaje asociado a la expresión regular $r = (0+1)^*00$. Primero observemos que el lenguaje asociado a r es el de todas las cadenas de 0's y 1's que terminan en 00. Una gramática que genera el lenguaje $L(r)$ es la siguiente: los símbolos terminales serán $0, 1$, las variables serán S, T y las producciones serán

$$S \to 0S \mid 1S \mid 0T, \qquad T \to 0U, \qquad U \to \varepsilon.$$

Demostración. Veamos primero que $L(r) \subset L(G)$: sea $x = a_1 a_2 \cdots a_k 00$ una cadena en $L(r)$, es decir que los a_i son 0 o 1 (en forma arbitraria). Si a_1 es 0, hacemos la derivación $S \Rightarrow 0S$, si es 1 hacemos $S \Rightarrow 1S$, es decir que tenemos $S \Rightarrow a_1 S$, en forma análoga obtenemos la derivación

$$S \Rightarrow a_1 S \Rightarrow a_1 a_2 S \Rightarrow \cdots \Rightarrow a_1 a_2 \cdots a_k S.$$

Si a $a_1 a_2 \cdots a_k S$ le aplicamos sucesivamente las producciones $S \to 0T$ y $T \to 0$, obtenemos $S \Rightarrow a_1 a_2 \cdots a_k S \Rightarrow a_1 a_2 \cdots a_k 0T \Rightarrow a_1 a_2 \cdots a_k 00$.

Veamos ahora que $L(G) \subset L(r)$: es claro que si $S \Rightarrow x$ una derivación con x compuesta de símbolos terminales, entonces la última producción que se usó fue $T \to 0$, y entonces la penúltima fue $S \to 0T$. Como las variables siempre están en el extremo derecho, la derivación es $S \Rightarrow x'S \Rightarrow x'0T \Rightarrow x'00 = x$. Es decir $x \in L(r)$.

□

Los ejemplos anteriores se pueden generalizar con el siguiente teorema, cuyo enunciado es un tanto complicado, pero que puede entenderse bien en casos concretos.

Teorema 3.1. *Sea r una expresión regular sobre una alfabeto Σ, entonces existe H una gramática regular con símbolos terminales en Σ tal que $L(r) = L(H)$. Más explícitamente: sea Σ un alfabeto.*

(1) $L(\emptyset) = \emptyset$ es $L(G)$ donde G es una gramática sin producciones.

(2) $L(\varepsilon) = \{\varepsilon\}$es $L(G)$ donde G tiene una única producción $S \to \varepsilon$.

(3) Para cada a en Σ, $L(a) = \{a\}$ *es* $L(G)$ *donde* G *tiene producciones* $S \to Ua$ *y* $T \to \varepsilon$.

Supongamos que r *y* r' *son expresiones regulares tales que* $L(r) = L(G)$, $L(r') = L(G')$*con* $G = (V,\Sigma,P,S)$ *y* $G' = (V',\Sigma,P',S')$. *Supongamos además, sin perdida de generalidad, que* $V \cap V' = \emptyset$, *es decir que no tienen variables comunes. Debido a la definición de gramática regular, el conjunto* P *es de la forma:*

$$\{C_k \to c_k D_k\} \cup \{U_t \to \varepsilon\},$$

para $k = 1,\ldots,m$, $t = 1,\ldots,s$. *Entonces:*

(4) $L(r+r')$ *es* $L(H)$, *donde* H *tiene como variables* $V = V \cup V'$, *la variable inicial igual a* S *y con producciones* $P_H = P \cup P' \cup \{S \to \alpha : \text{ si } S' \to \alpha\}$.

(5) $L(rr') = L(H)$, *donde* H *es la gramática regular con variables* $V_H = V \cup V'$, *variable inicial igual a* S *y con producciones* $P_H = P^{(1)} \cup P'$, *donde* $P^{(1)}$ *es igual a:*

$$\{C_k \to c_k D_k\} \cup \{C_k \to c_k S' : \text{si } D_k \to \varepsilon \in P\}$$

para $k = 1,\ldots,m$.

(6) $L(r^*) = L(H)$, *donde* H *es la gramática regular con variables* $V_H = V$,*variable inicial* S *y producciones:*

$$P_H = P \cup \{S \to \varepsilon\} \cup \{C_k \to c_k S : \text{si } D_k \to \varepsilon \in P\}$$

para $k = 1,\ldots,m$.

Demostración. La demostración es bastante directa, pero a su vez es tediosa. La dejamos como ejercicio para el lector. □

Ejercicio 3.1. Repetir los Ejemplos 3.7 y 3.8 usando el método del Teorema 3.1.

Como ya mencionamos, la recíproca del Teorema 3.1 también es verdadera, es decir, dado un lenguaje generado por una gramática regular G , existe una expresión regular r que denota el mismo lenguaje. En realidad esto lo probaremos de manera indirecta: es sencillo construir a partir de G un autómata no determinístico M con el mismo lenguaje que M. Luego par obtener r usamos el teorema de Kleene.

Proposición 3.2. *Sea* $G = (V,\Sigma,P,S)$ *una gramática regular, entonces existe* $M = (Q,\Sigma,\delta,q_0,F)$ *un NFA (sin* ε*-mov) tal que* $L(M) = L(G)$. *Explícitamente,* $Q = V$, $q_0 = S$, $B \in \delta(A,a)$ *sii* $A \to aB$ *y* $F = \{A \in V : A \to \varepsilon\}$.

La demostración es muy sencilla y se deja a cargo del lector.

4 Autómatas con pila

Hemos visto que los lenguajes que generan las expresiones regulares o, equivalentemente, los lenguajes regulares se pueden obtener a partir de autómatas finitos, y viceversa. En forma análoga, a los lenguajes libres de contexto le corresponden los autómatas con pila. En esta sección veremos la definición de los autómatas con pila, los lenguajes generados por ellos y algunos ejemplos. No probaremos, por estar fuera de los alcances de este texto, la equivalencia entre los lenguajes libres de contexto y los lenguajes aceptados por los autómatas con pila.

Un autómata con pila (PDA) es esencialmente un autómata finito que posee control sobre una pila, es decir una lista de la cual solo se puede "leer", "poner" o "sacar" el primer elemento. Dado el estado actual del autómata y el primer elemento de la pila, un símbolo de input nos llevará (posiblemente en forma no determinística) el estado siguiente y a la modificación que se debe hacer en el primer elemento de la pila. Diremos que una cadena es aceptada por *pila vacía* por el PDA si cuando la aplicamos obtenemos una pila vacía. Diremos que una cadena es aceptada por *estado final* por el PDA si lleva el estado inicial a uno final. Los lenguajes aceptados por los autómatas con pila, tantos los aceptados por pila vacía o por estado final, son los mismos que los aceptados por las gramáticas libres de contexto e incluyen estrictamente a los lenguajes regulares.

Ejemplo 4.1. El lenguaje $L = \{wcw^R : w \in (0+1)^* \text{ y } c \text{ símbolo } \}$ es un lenguaje no regular. Esto es fácil de ver usando el Pumping Lemma. Sin embargo, es un lenguaje libre de contexto generado por la gramática $S \to 0S0 \mid 1S1 \mid c$. Mostraremos a continuación un autómata con pila cuyo lenguaje por pila vacía es L. Los símbolos de input serán, obviamente, 0, 1 y c. Consideremos dos estados q_1 y q_2 y que en la pila se pueden "apilar" tres tipos de objetos rojos (R), verdes (V) y azules (A). La idea para definir el autómata es la siguiente: la pila comenzará con un solo elemento R para indicar, justamente, el comienzo de la pila. Ahora, pensemos a la pila como una "memoria" donde guardaremos la forma de la primera porción de la palabra hasta c: haremos que cada vez que el input sea 0, se agregue a la pila una A y cada vez que sea 1 se agregue una V. Estos inputs no cambian el estado inicial. Cuando ingresa el input c, cambiamos de estado para indicar que tenemos que empezar a leer el final de la palabra. Ahora deberemos desapilar convenientemente, de tal forma que si después de c viene w^R, la pila quede vacía. Por ejemplo, supongamos que w termina en 1, entonces, después de aplicar c, estamos en el segundo estado y la pila tiene en la parte superior una V. Esto indica que la última letra de w es un 1 y por lo tanto, para tener esperanza de que la palabra sea aceptada, la primera letra después de c debería ser un 1. Por lo tanto decimos que si el input es 1 y la pila muestra V, se retira V. Análogamente si en la parte superior hay un A y el input es 0, se retira A. Siguiendo así, si el sufijo de c es w^R, llegaremos a una pila con un solo elemento, el R. El último movimiento, un ε-movimiento, se define de la siguiente manera: si estamos en el segundo estado y la pila muestra el R, se saca el R. Resumiendo: el autómata tendrá las siguientes reglas:

1. Comenzamos en el estado q_1 y con R en la pila.

2. En el caso en que el estado es q_1 el autómata actúa de la siguiente manera. Si se ingresa el símbolo de input 0, agregamos a la pila una A. Si el símbolo de input es 1 agregamos una V. En ambos caso el estado permanece q_1. Si la entrada es c pasamos al estado q_2 y la pila no cambia.

3. En el caso en que el estado es q_2 el autómata actúa de la siguiente manera. Si se ingresa el símbolo de input 0 y el primero de la pila es A, se retira el primer elemento de la pila (es decir la A). Si el símbolo de input es 1 y el primero de la pila es V, se retira el primer elemento de la pila. Si el primer elemento de la pila es R, se lo retira sin esperar input. En todos los casos el estado continúa siendo q_2.

4. En las situaciones no contempladas en los items anteriores, el autómata no hace nada.

Haciendo algunos ejemplos con cadenas particulares es fácil convencerse que las únicas cadenas aceptadas por pila vacía son las de la forma wcw^R.

Definición 4.1. Un *autómata con pila* es una 7-upla $M = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$, en donde

1. Q es un conjunto finito de *estados*;

2. Σ es un alfabeto, el *alfabeto de entrada*;

3. Γ es un alfabeto, el *alfabeto de la pila*;

4. $q_0 \in Q$, el *estado inicial*;

5. $Z_0 \in \Gamma$, el *símbolo inicial* de la pila;

6. $F \subset Q$, los *estados finales*;

7. $\delta : Q \times (\Sigma \cup \{\varepsilon\}) \times \Gamma \to \mathcal{P}_f(Q \times \Gamma^*)$, la *función de transición*, donde $\mathcal{P}_f(Q \times \Gamma^*)$ indica los subconjuntos finitos de $Q \times \Gamma^*$.

Usaremos PDA (por sus siglas en inglés) como sinónimo de "autómata con pila".

En general haremos uso de letras minúsculas del comienzo del alfabeto $(a, b, c, \ldots)$ para denotar los símbolos del alfabeto de entrada (es decir de Σ) y usaremos letras minúsculas, pero del final del alfabeto $(\ldots, x, y, z)$, para denotar cadenas en Σ. Las letras mayúsculas $(A, B, C, \ldots, X, Y, Z)$, denotarán símbolos de la pila (es decir elementos de Γ) y las letras griegas minúsculas $(\alpha, \beta, \gamma, \ldots)$ denotarán elementos de Γ^*.

Observación 4.1. Debemos hacer algunos comentarios acerca de como se debe interpretar δ. Si tenemos, por ejemplo, que $\delta(q, a, Z) = \{(p, \gamma)\}$ esto lo debemos interpretar de la siguiente manera: si q estado y Z cima de la pila, al aplicarle a del alfabeto de entrada, el estado del autómata cambia a p y en la pila se produce el reemplazo de Z por γ, quedando el primer símbolo de γ como cima de la pila. Por ejemplo si estamos en un estado q y la pila es $Z_1Z_2Z_2$, entonces aplicar a usando la regla $\delta(q, a, Z_1) = \{(p, Z_2Z_1Z_2)\}$, hace que pasemos al estado p y que la pila pase a ser $Z_2Z_1Z_2Z_2Z_2$.

Los autómatas con pila permiten acciones no determinísticas, pues

$$\delta(q, a, Z) = \{(p_1, \gamma_1), \ldots, (p_m, \gamma_m)\}$$

indica la posibilidad de hacer diferentes acciones aún con el mismo input. Por otro lado, también están permitidos los ε-movimientos, es decir

$$\delta(q, \varepsilon, Z) = \{(p_1, \gamma_1), \ldots, (p_m, \gamma_m)\}$$

está permitido y permite, sin ningún input, cambiar el estado y la pila.

Notemos que los DFA y NFA definidos en capítulos anteriores son casos especiales de autómatas con pila: basta "olvidarse" de la pila en la definición.

Ejemplo 4.2. Describamos en lenguaje formal el autómata con pila del Ejemplo 4.1: $M = (\{q_1, q_2\}, \{0, 1, c\}, \{A, V, R\}, \delta, q_1, R, \emptyset)$. Observemos que hemos determinado que el conjunto de estados finales es vacío. Esto se debe a que en este caso estamos interesados en el lenguaje aceptado por pila vacía y por lo tanto los estados finales son irrelevantes. La descripción de δ es

$$\begin{array}{lll} \delta(q_1, 0, X) = \{(q_1, AX)\}, & \delta(q_1, 1, X) = \{(q_1, VX)\}, & \delta(q_1, c, X) = \{(q_2, X)\}, \\ \delta(q_2, 0, A) = \{(q_2, \varepsilon)\}, & \delta(q_2, 1, V) = \{(q_2, \varepsilon)\}, & \delta(q_2, \varepsilon, R) = \{(q_2, \varepsilon)\}, \end{array}$$

donde X es un símbolo arbitrario de la pila. Las transiciones no descritas son $\delta(q, a, Z) = \emptyset$.

Dada una cadena del alfabeto de entrada queremos describir formalmente la situación del autómata con pila después de haberse aplicado parte de la cadena. Para ello debe registrarse, sin duda, el estado actual del autómata y el contenido de la pila. Además, registraremos la parte de la cadena que todavía no ha sido aplicada. Formalmente:

Definición 4.2. Sea $M = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ un PDA. Una *descripción instantánea (ID)* es un triple (q, w, γ), donde $q \in Q$, $w \in \Sigma^*$ y $\gamma \in \Gamma^*$.

Si $(q, aw, Z\gamma)$ y $(p, w, \beta\gamma)$ son dos ID, denotaremos

$$(q, aw, Z\gamma) \vdash (p, w, \beta\gamma)$$

si $(p, \beta) \in \delta(q, a, Z)$. Dadas (q, w, γ) y (q', w', γ') descripciones instantáneas, denotaremos

$$(q, w, \gamma) \vdash^* (q', w', \gamma')$$

si existen $(q_1, w_1, \gamma_1), \ldots, (q_m, w_m, \gamma_m)$ descripciones instantáneas tales que

$$(q, w, \gamma) \vdash (q_1, w_1, \gamma_1) \vdash \cdots \vdash (q_m, w_m, \gamma_m) \vdash (q', w', \gamma').$$

Por convención, será aceptado $(q, w, \gamma) \vdash^* (q, w, \gamma)$.

Las descripciones instantáneas serán útiles para definir el lenguaje aceptado por un PDA.

Definición 4.3. Sea $M = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ un PDA. Entonces, definimos $L(M)$ el *lenguaje de M por estado final*, como

$$L(M) = \{w \in \Sigma^* : (q_0, w, Z_0) \vdash^* (p, \varepsilon, \gamma) \text{ para algún } p \in F, \gamma \in \Gamma^*\}.$$

El *lenguaje de M por pila vacía* es:

$$N(M) = \{w \in \Sigma^* : (q_0, w, Z_0) \vdash^* (p, \varepsilon, \varepsilon) \text{ para algún } p \in Q\}.$$

Como ya hemos mencionado el conjunto de los lenguajes aceptados por estado final es equivalente al conjunto de los lenguajes aceptados por pila vacía. La demostración de este hecho sigue el espíritu de las demostraciones que hemos visto en teoría de lenguajes, dado un lenguaje definido de cierta manera, construimos en forma algorítmica un autómata que acepta ese lenguaje.

Recordemos que si M es un NFA, puede asociarse trivialmente M' un PDA, que esencialmente es el mismo M con una pila que no se usa. Es claro entonces, por las respectivas definiciones, que el lenguaje de M coincide con el lenguaje de M' por estado final.

Ejemplo 4.3. $L = \{aa^R : a \in \{0,1\}^*\}$

En la sección 3.2 hemos visto que toda gramática se puede reducir a una gramática equivalente en la forma normal de Greibach. La siguiente proposición probará el "implica" de la equivalencia entre lenguajes generados por gramáticas libres de contexto y lenguajes aceptados por autómatas con pila.

Proposición 4.1. *Sea G una CFG en la forma normal de Greibach. Entonces existe M un PDA tal que* $L(G) = N(M)$.

Demostración. Denotemos $L = L(G)$ y hagamos la demostración en el caso en que ε no esté en L. La demostración en el otro caso es similar y se deja a cargo del lector. G está en la forma normal de Greibach, es decir toda producción es de la forma $A \to a\gamma$ con $\gamma \in V^*$. Definamos el PDA

$$M = (\{q\}, T, V, \delta, q, S, \{\emptyset\}),$$

donde $(q, \gamma) \in \delta(q, a, A)$ si y sólo si $A \to a\gamma$ está en P. Observemos que en este caso los estados del autómata no tiene importancia (siempre estamos en el mismo estado) y lo único que importa es la

pila. Las operaciones elementales que hace δ sobre la pila claramente simulan las producciones y una composición de estas operaciones simula una derivación a izquierda. De manera formal debemos demostrar que

$$S \Rightarrow w \qquad \text{si y sólo si} \qquad (q, w, S) \vdash^* (q, \varepsilon, \varepsilon),$$

para $w \in T^*$. En realidad es más sencillo demostrar

$$S \Rightarrow w\gamma \qquad \text{si y sólo si} \qquad (q, w, S) \vdash^* (q, \varepsilon, \gamma), \tag{5}$$

para $w \in T^*$ y $\gamma \in V^*$ y se deja como ejercicio para el lector. Claramente, esto implica lo anterior haciendo $\gamma = \varepsilon$. □

Finalizaremos la sección dando la definición de autómatas con pila determinísticos.

Definición 4.4. Un *autómata con pila determinístico* es una 7-upla $M = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$, en donde

1. Q es un conjunto finito de *estados*;
2. Σ es un alfabeto, el *alfabeto de entrada*;
3. Γ es un alfabeto, el *alfabeto de la pila*;
4. $q_0 \in Q$, el *estado inicial*;
5. $Z_0 \in \Gamma$, el *símbolo inicial* de la pila;
6. $F \subset Q$, los *estados finales*;
7. $\delta : Q \times (\Sigma \cup \{\varepsilon\}) \times \Gamma \to \{\emptyset\} \cup (Q \times \Gamma^*)$, la *función de transición*, tal que si $\delta(q, \varepsilon, Z) \in Q \times \Gamma^*$, entonces $\delta(q, a, Z) = \emptyset$ para todo $a \in \Sigma$.

Usaremos DPDA (por sus siglas en inglés) como sinónimo de "autómata con pila determinístico".

Observemos que las transiciones de un DPDA tienen las siguientes restricciones:

1. $| \delta(q, a, Z) | = 0$ o 1.
2. Si $| \delta(q, \varepsilon, Z) | > 0$, entonces $| \delta(q, a, Z) | > 0$ para todo $a \in \Sigma$.

Haremos algunos comentarios respecto a los DPDA

1. los lenguajes aceptados por pila vacía correspondientes a DPDA son lenguajes que tienen la siguiente propiedad: si x y y son cadenas aceptadas y $x \neq y$, entonces x no puede ser prefijo de y. En particular, hay lenguajes regulares que no son aceptados por DPDA por pila vacía (por ejemplo 0^*).
2. Consideremos los lenguajes aceptados por DPDA por estado final, en este caso este conjunto de lenguajes contiene estrictamente a los lenguajes regulares y está contenido estrictamente en los lenguajes aceptados por PDA. Por ejemplo, el lenguaje $L = \{aa^R : a \in \{0,1\}^*\}$ es un lenguaje aceptado por un PDA, pero no es posible obtenerlo como lenguaje de un DPDA.

4.1 Ejercicios

1. Demostrar que los enteros se pueden generar con una gramática regular.
2. Encontrar gramáticas libres de contexto que generen los siguientes conjuntos
   (a) Todas las cadenas distintas de ε definidas sobre $\{a,b\}$.
   (b) Cadenas definidas sobre $\{a,b\}$ que empiecen con a.
   (c) Cadenas definidas sobre $\{a,b\}$ que terminen en ba.
   (d) Cadenas definidas sobre $\{a,b\}$ que contengan ba.
   (e) Cadenas definidas sobre $\{a,b\}$ que no terminen en ab.
   (f) Enteros que no empiecen con 0 (hacerlo con BNF).
   (g) Números con punto flotante (como 0.294, 89.0, 45.895).
   (h) Números exponenciales (que incluyan a los números con punto flotante y a otros como 6.9E4, 5E23, 7.5E-3, 4E-5).
3. Sea G la gramática con símbolo inicial S y derivaciones

$$S \to bS \mid aA \mid a, \qquad A \to aS \mid bB, \qquad B \to bA \mid aS \mid b$$

   (donde a y b son los símbolos terminales).
   (a) Demuestre, proporcionando la derivación correspondiente, que las siguientes cadenas pertenecen a $L(G)$

$$aaabb, \qquad bbbaaaaa, \qquad abaaabbabbaa.$$

   (b) Probar que $L(G)$ es el conjunto de todas las cadenas con un número impar de símbolos a.
4. Sea G la gramática regular definida por las producciones

$$S \to bS \mid aA \mid b, \qquad A \to aS \mid bA \mid a$$

   (donde a y b son los símbolos terminales). Demuestre que $\alpha \in L(G)$ sii $\alpha \neq \varepsilon$ y contiene un número par de símbolos a.
5. Demuestre que el lenguaje

$$\{a^n b^n c^k \mid n,k \in \mathbb{N}\}$$

   es libre de contexto, pero no es regular.
6. Obtener un autómata finito no determinístico que acepte únicamente las cadenas generadas por la siguiente gramática regular G: las variables son S y C, con S como variable inicial; las constantes son a,b y las producciones son

$$S \to bS, \quad S \to aC, \quad C \to bC, \quad C \to b.$$

7. Sea G la gramática regular definida por las producciones

$$S \to bS \mid aA \mid b, \qquad A \to aS \mid bA \mid a$$

   (donde a y b son los símbolos terminales). Obtener un autómata finito no determinístico$M$ tal que $L(M)$ es el lenguaje generado por G.

8. Sea L_1 (respectivamente L_2) el lenguaje generado por la gramática del Ejercicio 3 (respectivamente, Ejercicio 4). Encuentre una gramática regular que genere el lenguaje L_1L_2.

9. Demuestre que el conjunto L de cadenas sobre a,b, definido por

$$L = \{x_1 \cdots x_n \mid x_1 \cdots x_n = x_n \cdots x_1\}$$

(es decir las cadenas capicúas), no es un lenguaje regular.

10. Dada la expresión regular

$$b(a^* + b)^* bb^* a$$

construir

(a) Un autómata finito que acepte exactamente el lenguaje que denota la expresión regular.

(b) Una gramática que genera exactamente el lenguaje que denota la expresión regular.

11. Dada la expresión regular

$$(a + bb)^*(b^* aa + b)^*$$

construir:

(a) Un autómata finito que acepte exactamente el lenguaje que denota la expresión regular.

(b) Una gramática que genera exactamente el lenguaje que denota la expresión regular.

12. Dada la expresión regular

$$(ab)^*(ab^* aa + b)^*$$

construir:

(a) Un autómata finito que acepte exactamente el lenguaje que denota la expresión regular.

(b) Una gramática que genera exactamente el lenguaje que denota la expresión regular.

13. Dada la expresión regular

$$0(0^* + 1)^* 00^* 1$$

construir

(a) Un autómata finito que acepte exactamente el lenguaje que denota la expresión regular.

(b) Una gramática que genera exactamente el lenguaje que denota la expresión regular.

14. Probar que el lenguaje $L = \{0^n 1^n 1^m 0^m \mid n \in \mathbb{N}\}$ es libre de contexto pero no es regular.

15. Probar que el lenguaje $L = \{1^n 0^n 1^m 0^m \mid n,\ m \in \mathbb{N}\}$ es libre de contexto pero no es regular.

16. Dado el lenguaje $L = \{0^n 110^n, n \geq 0\}$,

(a) Probar que es libre de contexto.

(b) Encontrar la forma normal de Chomsky.

(c) Probar que no es regular.

17. Considere el lenguaje $L = \{ab^{n+2}baa^n\}$.

(a) Pruebe que L es libre de contexto.

(b) Encontrar la forma normal de Chomsky.

(c) Determine si L es regular o no. Justifique su respuesta.

18. Dado el lenguaje $L = \{110^n 110^n, n \geq 0\}$,

(a) Probar que es libre de contexto.

(b) Encontrar la forma normal de Chomsky.

(c) Probar que no es regular.