Benemérita Universidad Autónoma de Puebla

FACULTAD DE CIENCIAS FÍSICO MATEMÁTICAS
Posgrado en Ciencias Matemáticas

Que para obtener el grado de:
Maestría en Ciencias Matemáticas

Autómatas Arbóreos

Presenta:
Patricia Aguilar Rangel

Director de tesis:
Dr. Carlos Guillén Galván

Puebla, Puebla, 20 de abril del 2022

DR. SEVERINO MUÑOZ AGUIRRE
SECRETARIO DE INVESTIGACIÓN Y
ESTUDIOS DE POSGRADO, FCFM-BUAP
P R E S E N T E:

Por este medio le informo que la C:

PATRICIA AGUILAR RANGEL

estudiante de la Maestría en Ciencias (Matemáticas), ha cumplido con las indicaciones que el Jurado le señaló en el Coloquio que se realizó el día 6 de abril de 2022, con la tesis titulada:

Autómatas arbóreos

Por lo que se le autoriza a proceder con los trámites y realizar el examen de grado en la fecha que se le asigne.

A T E N T A M E N T E.
H. Puebla de Z. a 7 de abril de 2022

DRA. PATRICIA DOMÍNGUEZ SOTO
COORDINADORA DEL POSGRADO
EN MATEMÁTICAS.

Facultad
de Ciencias
Físico Matemáticas

Av. San Claudio y 18 Sur, edif. FM1
Ciudad Universitaria, Col. San
Manuel, Puebla, Pue. C.P. 72570
01 (222) 229 55 00 Ext. 7550 y 7552

Dedicatoria

Dedico este trabajo de tesis a las personas que son muy importantes en mi vida:

A mis padres, la señõra Cesarea Rangel Morales y el señõr Romoaldo Sebastian Aguilar García, quienes con su amor y esfuerzo siempre me apoyaron en todo lo que necesitara a pesar de los problemas. Siempre estarán en mi corazón.

A mis hermosos hijos Julio César y Eduardo, quienes han sido mi mayor motivación para cada día seguir adelante.

A mi esposo Froylan Sergio, quien siempre ha sido un gran apoyo en todo mi camino, por creer en mí y por darme tanta felicidad.

A mi suegra María Elena quien siempre me ha apoyado cuando más lo he necesitado.

A mis hermanos Tomasita, Rosa María y César quienes me han apoyado en todo momento, son los mejores hermanos.

Agradecimientos

Este trabajo de tesis ha sido posible gracias a la colaboración de varias personas e instituciones, las cuales serán mencionadas a continuación y merecen reconocimiento especial.

Quiero agradecer principalmente a mi director de tesis el Dr. Carlos Guillén Galván por haber confiado en mí, por darme la valiosa oportunidad de realizar este trabajo, sin su apoyo este trabajo no habría sido posible.

Agradecer al Dr. Juan Alberto Escamilla Reyna, al Dr. César Bautista Ramos, al Dr. Rafael Lemuz López y al Dr. Francisco Javier Mendoza, por aceptar formar parte de mi jurado sinodal y también ser parte de mi comité tutorial, por sus observaciones, las cuales hicieron de este un mejor trabajo.

Agradecer a la Facultad de Ciencias Físico Matemáticas por una vez más permitirme ser parte de esta gran institución.

Quiero agradecer al Consejo Nacional de Ciencia y Tecnología por su valioso apoyo económico que aportó durante mis estudios de maestría, gracias a su apoyo este trabajo llego a un buen termino.

Agradecer a mi familia por todo el apoyo que me brindaron para que pudiera dedicar tiempo a mis estudios.

Introducción

Los Autómatas de estado finito que procesan cadenas y árboles enraizados de símbolos apoyan una metodología parámetro fijo tratable (PFT) importante. Esta metodología es basada en la siguiente estrategia algorítmica de dos pasos; el primer paso consiste en calcular una representación para el grafo de ancho arbóreo acotado, como un árbol binario etiquetado, denominado árbol de análisis, para el objeto. En el segundo, se utiliza un autómata arbóreo de estado finito para reconocer con precisión los árboles de análisis que representan los objetos (por ejemplo, grafos de ancho arbóreo acotado) que tienen la propiedad de interés, por ejemplo, tener un ciclo Hamiltoniano [3].

Como sabemos, la Teoría de la Complejidad clásica analiza y clasifica problemas de acuerdo a la cantidad de recursos necesarios para resolver un problema, comúnmente los recursos que se consideran son el tiempo y el espacio.

La *Teoría de la Complejidad Parametrizada* no sólo considera la complejidad con respecto al tamaño de las entradas, también considera la complejidad en términos de un *parámetro*, el cual es un valor numérico dependiente de la entrada en una forma arbitraria. La noción central de esta teoría es la *tratabilidad de parámetro fijo*.

Un resultado cuya demostración se basa en los autómatas arbóreos es el teorema de Courcelle [3], este resultado establece las condiciones para la existencia de algoritmos parámetro fijo tratables para problemas sobre grafos que pueden ser expresables en lógica monádica de segundo orden (MS_2) con ancho árboreo acotado.

El Teorema de Courcelle es un meta-teorema basado en

lógica para establecer, junto con el Teorema de Bounlander [5], que varias propiedades teóricas sobre grafos son PFT decidibles en tiempo lineal, cuando el parámetro es de entrada el ancho árboreo del grafo. Un meta-teorema es un teorema sobre teoremas, por ejemplo, el Teorema de la deducción es un meta-teorema [10]. En términos informales el Teorema de Courcelle tiene la forma:

Si la propiedad de interés es expresable en lógica MS_2, entonces, parametrizando por el ancho árboreo de la entrada, puede ser determinado en PFT tiempo lineal, en tanto el grafo tenga la propiedad.

Muchas actividades en todas las áreas del conocimiento para su desarrollo requieren del apoyo de recursos tecnológicos, dentro de estos recursos, las computadoras y los sistemas de software son intrumentos de considerable valor. En el desarrollo de sistemas de software es común contar con algoritmos que realizan una tarea en determinado tiempo y espacio de memoria. Particularmente existen muchos algoritmos que realizan una tarea específica en un tiempo considerablemente grande, estos algoritmos pueden resultar muy incomodos para un usuario que necesita obtener una respuesta en tiempo real, por lo que es imprescindible buscar métodos que nos conduzcan a la "eficiencia" de los algoritmos.

El término de eficiencia esta ligado a los conceptos de complejidad en tiempo y espacio. En muchos casos el espacio que ocupan los algoritmos puede no ser problema ya que cada vez contamos con dispositivos con gran poder de almacenamiento. En consecuencia la investigación sobre la eficiencia computacional se centra principalmente en mejorar complejidad en tiempo de los algoritmos y en la clasificación de los problemas que pueden ser tratados de manera eficiente.

El presente documento abarca desde el estudio y presentación de los autómatas en su versión clásica hasta su versión

arbórea, así como los resultados principales que nos conducen a los Teoremas de Courcelle y Bounlander.

En el capítulo 1 se describen algunos conceptos de la teoría de grafos, los cuales serán fundamentales para el desarrollo de este trabajo. También se presentan algunos conceptos básicos de la teoría de hipergrafos.

En el capítulo 2 se presentan los conceptos y resultados principales al rededor de las clases de complejidad para problemas de decisión, problemas de conteo y problemas de optimización. También se desarrolla la teoría de complejidad parametrizada cuya noción central es la tratabilidad de parámetro fijo.

El capítulo 3 es dedicado a la teoría básica de los autómatas clásicos, así como a los conceptos de lenguajes regulares y gramáticas formales.

En el capítulo 4 se generaliza el concepto de autómata clasico a autómata arbóreo. Se desarrolla el concepto de gramática arbórea y finalmente se presenta el teorema de Myhill-Nerode para árboles.

Símbolos

$\mathbb{N}$	Conjunto de los números naturales.
G	Grafo.
$V(G)$	Conjunto de vértices.
$E(G)$	Conjunto de aristas.
$\|G\|$	Número de vértices de G.
$\|\|G\|\|$	Número de aristas de G.
H	Hipergrafo.
Σ	Alfabeto.
Σ^*	Conjunto de cadenas finitas formadas en el alfabeto Σ.
Σ^{**}	Conjunto de árboles enraizados con etiquetas en el alfabeto Σ.
MT	Máquina de Turing.
$L(M)$	Conjunto de cadenas aceptado por el autómata M.
$\mathcal{L}$	Conjunto de lenguajes aceptados por autómatas finitos.
PFT	Parámetro fijo tratable.
P	Clase de complejidad **P**.
NP	Clase de complejidad **NP**.
SAT	Conjunto de fórmulas sastisfactibles.
$AAFD$	Aceptación por autómata finito determinista.
$AAFND$	Aceptación por autómata finito no determinista.
$EXP(\Sigma)$	Conjunto de expresiones regulares sobre Σ.

Lista de figuras

Índice

Capítulo 1

Preliminares

En esta sección se introducen los conceptos básicos de la teoría de grafos, los cuales son necesarios para el desarrollo de este trabajo. La mayoría de los conceptos que aquí se presentan, se pueden encontrar en [4].

1.1. Grafos

Dado un conjunto A no vacío, denotaremos con $[A]^2$ al conjunto de todos los subconjuntos binarios de A.

El concepto de grafo surge de manera natural en un grupo de objetos y una relación que existe entre los objetos. Para ser más precisos, se tiene la siguiente definición.

Definición 1.1. *Un **grafo** es un par de conjuntos $G = (V, E)$ tales que $E \subset [V]^2$, donde los elementos de V son los vértices o nodos del grafo G y los elementos de E son las aristas del grafo G.*

Si $G = (V, E)$ es un grafo con conjunto de vértices V, se dice que es un grafo sobre V. Los símbolos $V(G)$ y $E(G)$ denotan los conjuntos de vértices y el conjunto de aristas del grafo G, respectivamente. El número de vértices de el grafo G es su orden, lo denotamos por $|G|$ y el número de aristas es

denotado por $||G||$.

A continuación veamos la representación de un grafo. En esta representación, para cada vértice asignamos un punto y unimos dos puntos con una línea si los dos vértices correspondientes forman una arista.

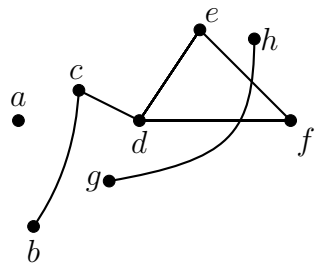


Figura 1.1: El grafo sobre $V = \{a, b, ..., h\}$ con conjunto de aristas $E = \{\{b, c\}, \{c, d\}, \{d, e\}, \{d, f\}, \{e, f\}, \{g, h\}\}$.

En la Figura 1.1, podemos observar que el orden de G es, $|G| = 8$; y el número de aristas es $||G|| = 6$.

Dado un grafo $G = (V, E)$, el **grafo vacío** $G = (\emptyset, \emptyset)$, se denota por $\emptyset$. Si el orden de G es 0 o 1, G se llama el **grafo trivial**. Decimos que el vértice v es **incidente** en la arista e si $v \in e$. Los vértives v, w incidentes con una arista e son sus **extremos** y la arista e une esos extremos. Una arista $\{x, y\}$ usualmente se denotará por xy o yx. Dos vértices x e y son **adyacentes** o vecinos si $\{x, y\}$ es una arista de G y dos aristas $e \neq f$ son **adyacentes** si estas tienen un punto en común. Decimos que un grafo G es **completo** si todos sus vértices son adyacentes dos a dos. Un grafo completo de n vértices es denotado por K^n; un K^3 es llamado un **triángulo**, véase la Figura 1.2.

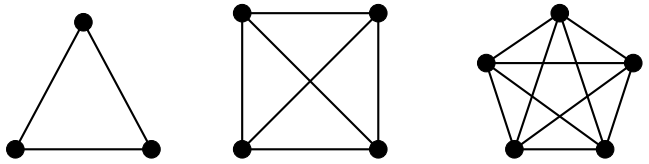


Figura 1.2: Los grafos completos; K^3, K^4 y K^5, respectivamente.

Sean $G = (V, E)$ y $G' = (V', E')$ dos grafos. Definimos la **unión** e **intersección** de los grafos G y G', respectivamente por:

$$G \cup G' := (V \cup V', E \cup E') \text{ y } G \cap G' := (V \cap V', E \cap E').$$

Si $G \cap G' = \emptyset$, entonces G y G' son disjuntos. Si $V' \subseteq V$ y $E' \subseteq E$, entonces G' es un **subgrafo** de G (y G es un **supergrafo** de G'), escrito como $G' \subseteq G$. Menos formalmente, decimos que G contiene a G'. Si $G' \subseteq G$ y $G' \neq G$, entonces G' es un subgrafo propio de G y se denota por $G' \subset G$, véase la Figura 1.3.

Si $G' \subseteq G$ y G' contiene todas sus aristas $xy \in E$ con $x, y \in V'$, entonces G' es un **subgrafo inducido** de G; decimos que V' induce o amplia G' en G, y escribimos $G' =: G[V']$. Así, si $U \subseteq V$ es cualquier conjunto de vértices, entonces $G[U]$ denota el grafo sobre U cuyas aristas son precisamente las aristas de G con ambos extremos en U.

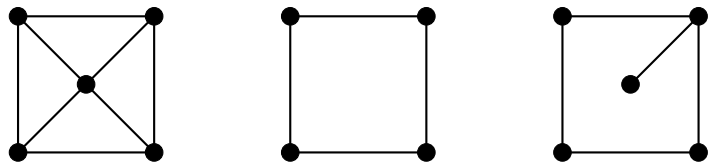


Figura 1.3: Un grafo G con subgrafos G' y G''.

Observemos que en la Figura 1.3, el grafo G' es un subgrafo inducido de G, pero G'' no lo es.

Un **camino** es un grafo no vacío $P = (V, E)$ de la forma:

$$V = \{x_0, x_1, ..., x_k\} \quad \text{y} \quad E = \{x_0x_1, x_1x_2, ..., x_{k-1}x_k\},$$

donde x_i son todos distintos. Los vértices x_0 y x_k están ligados por P y son llamados sus **vértices extremos** o **extremos**. Los vértices $x_1, ..., x_{k-1}$ son llamados vértices **internos** de P, véase la Figura 1.4. El número de aristas de una camino es su **longitud**, y el camino de longitud k es denotado por P^k. Se considera a K^1 como el camino de longitud cero y se denota por P^0.

Se hace referencia a un camino por la sucesión natural de sus vértices. Más precisamente, alguna de las dos sucesiones naturales: $x_0...x_k$ y $x_k...x_0$ denotan el mismo camino.

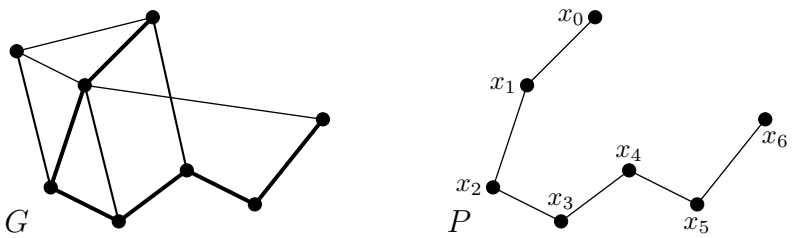


Figura 1.4: Un camino $P = P^6$ en el grafo G.

Si $P = x_0...x_{k-1}$ es una camino y $k \geq 3$, entonces el grafo $C := P + x_{k-1}x_0$ es llamado un **ciclo**. De manera análoga que en los caminos, denotamos un ciclo por su sucesión de vértices, así el ciclo C puede ser escrito como $x_0...x_{k-1}x_0$. La **longitud** de un ciclo es su número de aristas (o vértices). Un ciclo de longitud k es llamado un k-**ciclo** y es denotado por C^k.

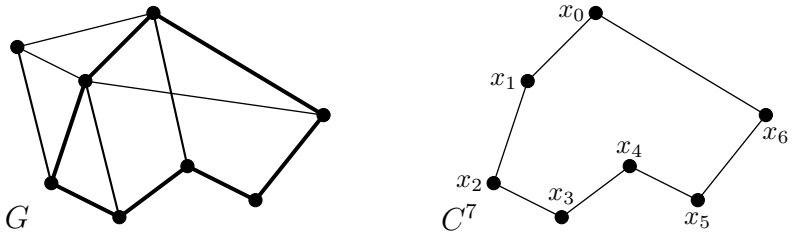


Figura 1.5: El ciclo $C^7 := P^6 + x_6x_0$ en el grafo G.

La **distancia** $d_G(x, y)$ entre dos vértices x, y en un grafo G es la longitud del camino más corto; si no existe tal camino, se establece que $d_G(x, y) := \infty$.

Un grafo G es llamado **conexo** si es no vacío y cualquier par de sus vértices están unidos por un camino en G. Si $U \subset V(G)$ y $G[U]$ es conexo, diremos que U es conexo en G. Si un grafo no es conexo, diremos que es **disconexo**.

Como podemos observar en la Figura 1.5, el ciclo $C = C^7$, es un ciclo de longitud 7 en G. Además, el grafo G es conexo.

Sea $G = (V, E)$ un grafo. Un subgrafo conexo maximal de G es una **componente** de G. Claramente las componentes son subgrafos inducidos y sus vértices forman una partición de V.

Un grafo **acíclico**, uno que no contenga ciclos, es llamado un **bosque**. Un bosque conectado es llamado un **árbol**. Así, un bosque es un grafo cuyas componentes son árboles, véase la Figura 1.6. Los vértices de grado 1 en un árbol son sus **hojas**, los otros son sus **vértices internos**.

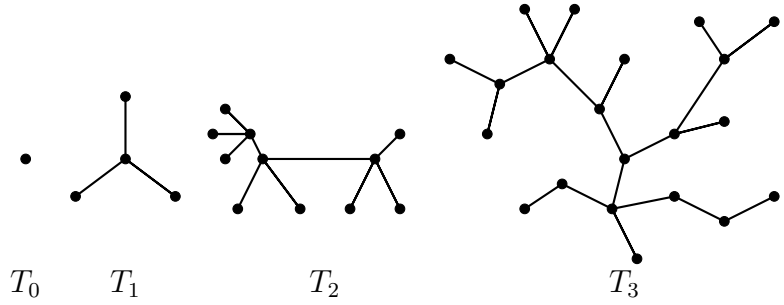


Figura 1.6: El bosque F, cuyas componentes son los árboles T_0, T_1, T_2 y T_3.

Algunas veces es conveniente considerar un vértice de un árbol como especial; tal vértice se denomina la **raíz** de este árbol. Un árbol T con una raíz fija r es un **árbol enraizado**. Los vértices a una distancia k de la raíz tienen una **altura** k y forman el $k-$ésimo **nivel** de T. Diremos que la **altura de un árbol** T será el mayor nivel de este, véase la Figura 1.7.

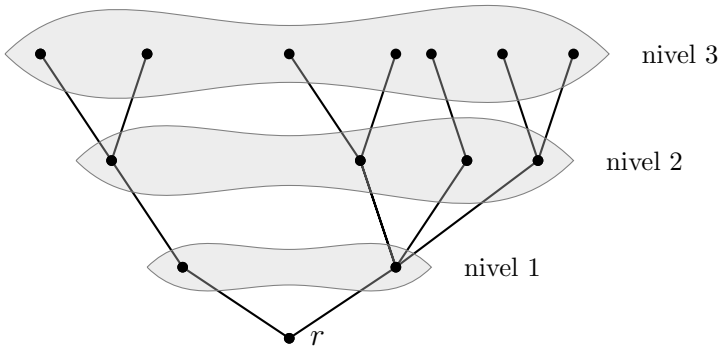


Figura 1.7: El árbol T con raíz r y altura 3.

Un **grafo dirigido** (o **digrafo**) es un par (V, E) de conjuntos disjuntos (de vértices y aristas) junto con dos mapeos $init\colon E \to V$ y $ter\colon E \to V$ que asigna a cada arista e un **vértice inicial** $init(e)$ y un **vértice terminal** $ter(e)$. La arista e se dice que es **dirigida de** $init(e)$ **a** $ter(e)$.

Note que un grafo dirigido puede tener diferentes aristas entre los mismos dos vértices x, y. Tales aristas son llamadas **aristas múltiple**; si estas tienen la misma dirección (digamos de x a y, estas son **paralelas**). Si $init(e) = ter(e)$, la arista e es llamada un **bucle**.

Un **grafo ponderado** (o **arista etiquetado**) es una terna (V, E, ϕ), donde $\phi\colon E \to Y$ es un mapeo tal que $\phi(e) = y$. La imagen de cada arista es llamada el **peso**, véase la Figura 1.8.

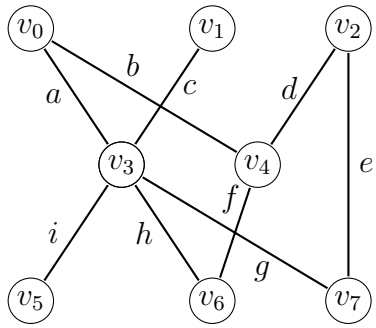


Figura 1.8: Un grafo ponderado.

1.2. Hipergrafos

En esta sección se introduce a los conceptos básicos de la teoría de hipergrafos. Si se desea estudiar más a profundidad este tema, se sugiere consultar [2]. La mayor parte de los conceptos que aquí se presentan, son tomados de esta referencia.

Los hipergrafos son sistemas de conjuntos finitos y forman probablemente, el concepto más general en matemáticas discretas. La idea básica consiste en considerar los conjuntos como aristas generalizadas y luego llamar hipergrafo a la familia de estas aristas (hiperaristas). Los hipergrafos son una generalización de grafos, por lo que muchas definiciones de grafos se refieren textualmente a los hipergrafos, vea [2].
A continuación se presenta la definición formal de hipergrafo.

Definición 1.2. *Un **hipergrafo** H denotado por $H = (V; E = (e_i)_{i\in I})$ sobre un conjunto finito V, es una familia $(e_i)_{i\in I}$ (donde I es un conjunto finito de índices) de subconjuntos de V, llamados hiperaristas. Algunas veces, denotamos a V por $V(H)$ y E por $E(H)$.*

Diremos que **el orden** del hipergrafo $H = (V; E)$ es la cardinalidad de V, es decir, $|V| = n$; su **tamaño** es la cardinalidad de E, es decir, $|E| = m$. Por definición, el **hipergrafo vacío** es $H = (\emptyset, \emptyset)$ y un **hipergrafo trivial** es $H = (V \neq \emptyset, \emptyset)$. En lo que sigue, a menos que se indique lo contrario, un hipergrafo tiene un conjunto no vacío de vértices, un conjunto no vacío de hiperaristas y no contiene hiperaristas vacías.

Sea $(e_j)_{j\in J}$, con $J \subseteq I$ una subfamilia de hiperaristas de $E = (e_i)_{i\in I}$, denotamos el conjunto de vértices pertenecientes a $\cup_{j\in J} e_j$ por $V(\cup_{j\in J} e_j)$, pero algunas veces usamos e para $V(e)$. Por ejemplo, a veces usamos $e \cap V'$ para $V(e) \cap V'$, con $V' \subseteq V$.

En un hipergrafo un vértice x es **aislado** si $x \in V - \bigcup_{i\in I} e_i$. Si $V = \bigcup_{i\in I} e_i$, el hipergrafo no tiene vértices aislados. Una hiperarista $e \in E$ tal que $|e| = 1$ es un **bucle**. Dos vértices en un hipergrafo son **adyacentes** si existe una arista que los contiene. En particular, si $\{x\}$ es una hiperarista, entonces x es adyacente a sí mismo. Dos hiperaristas en un hipergrafo son **incidentes** si tienen intersección no vacía.

Sea $H = (V; E = (e_i)_{i\in I})$ un hipergrafo:

- El **subhipergrafo inducido** $H(V')$ del hipergrafo H donde $V' \subseteq V$ es el hipergrafo $H(V') = (V'; E')$, donde

$$E' = \{V(e_i) \cap V' \neq \emptyset : e_i \in E \text{ y } e_i \text{ es un bucle o}$$

$$|V(e_i) \cap V'| \geq 2\}.$$

  La letra E' puede representar un conjunto multilpe.

- Dado un subconjunto $V' \subseteq V$, el **subhipergrafo** H' es el hipergrafo

$$H' = (V', E' = (e_j)_{j \in J}),$$

tal que para toda $e_j \in E'$, se tiene que $e_j \subseteq V'$.

- Un **hipergrafo parcial** H' del hipergrafo H generado por $J \subseteq I$, es un hipergrafo

$$H' = (V', (e_j)_{j \in J}).$$

Donde $\bigcup_{j \in J} e_j \subseteq V'$. Observemos que podemos tener $V' = V$.

La **estrella** $H(x)$ centrada en x es la familia de hiperaristas $(e_j)_{j \in J}$ que contiene a x; $d(x) = |J|$ es el grado de x a excepción de un bucle $\{x\}$, donde el grado $d(x) = 2$. Si el hipergrafo es sin hiperaristas repetidas, el grado se denota por $d(x) = |H(x)|$, excepto para un bucle $\{x\}$, donde el grado $d(x) = 2$. El máximo grado de un hipergrafo H es denotado por $\Delta(H)$. Si cada vértice tiene el mismo grado, decimos que el hipergrafo es **regular**, o $k-$**regular** si para cada $x \in V$, se tiene que $d(x) = k$. Si la familia de hiperaristas es un conjunto, esto es, si $i \neq j$ si y solo si $e_i \neq e_j$, decimos que H es sin hiperaristas repetidas. El **rango** $r(H)$ de H es la máxima cardinalidad de una hiperarista en el hipergrafo, esto es, $r(H) = \text{máx}_{i \in I}|e_i|$ y la mínima cardinalidad de una hiperarista es el **co-rango**, $cr(H) = \text{mín}_{i \in I}|e_i|$. Si $r(H) = cr(H) = k$, el hipergrafo es **uniforme** o $k-$**uniforme**.

El Hipergrafo H de la Figura 1.9 tiene orden: $|V| = 9$; tamaño: $||E|| = 6$; un bucle: e_6; un vértice aislado: v_8; los vértices v_4, v_5, v_6 son adyacentes; las aristas e_2, e_5 son incidentes. Si $H' = (V', E')$, donde $V' = \{v_2, v_3, v_4, v_5\}$ y $E' = \{e_2, e_3\}$, entonces H' es un subhipergrafo con un vértice aislado: v_5. Si $V' = \{v_1, v_3, v_4, v_5, v_9\}$, el supergrafo inducido $H(V')$ del hipergrafo H tiene hiperaristas:

$e'_1 = V' \cap e_1 = \{v_1\}$, $e'_2 = V' \cap e_2 = \{v_3\}$, $e'_3 = V' \cap e_3 = \{v_4\}$, $e'_4 = V' \cap e_4 = \{v_4, v_5\}$, $e'_5 = V' \cap e_5 = \{v_5\}$.
El hipergrafo parcial generado por $J = \{1, 2, 3, 4\}$ de H es $H' = (V' = \{v_1, v_2, v_3, v_4, v_9\}; E' = \{e_1, e_2, e_3, e_4\})$. La estrella $H(v_2) = \{e_2, e_3, e_4\}$. El grado de v_2 es 3. El rango $r(H) = 3$; el corango $cr(H) = 1$.

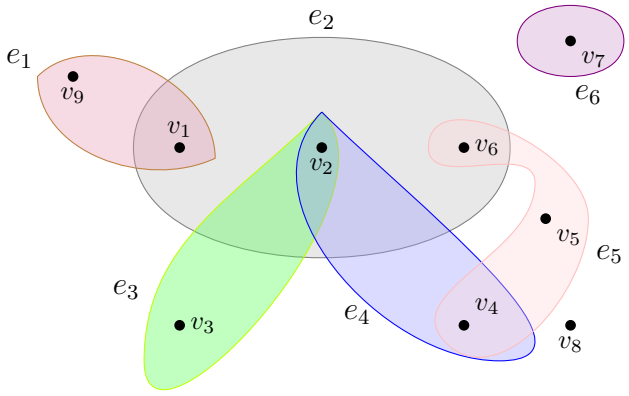


Figura 1.9: El Hipergrafo H.

Sea $H = (V; E)$ un hipergrafo sin vértices aislados. Un **camino** P en H de x a y es una sucesión alternante de vértices y aristas

$$x = x_1, e_1, x_2, ..., x_s, e_s, x_{s+1} = y$$

tales que

- $x_1, x_2, ..., x_s, x_{s+1}$ son vértices distintos con la posibilidad de que $x_1 = x_{s+1}$;
- $e_1, e_2, ..., e_s$ son hiperaristas distintas;
- $x_i, x_{i+1} \in e_i$, con $i = (1, 2, ..., s)$.
- Si $x = x_1 = x_{s+1} = y$, el camino es llamado un **ciclo**.

El entero s es la longitud del camino P. Observemos que si existe un camino de x a y, también existe un camino de y a x. En tal caso decimos que P conecta a x con y. Un hipergrafo es **conexo** si para cualquier par de vértices existe un camino que los conecta. Si no existe tal camino, decimos que lo vértices están desconectados. La **distancia** $d(x, y)$ entre dos vértices x, y es la mínima longitud de un camino que conecta a x con y. Si existe un par de vértices x, y sin camino de x a y (o de y a x), definimos $d(x, y) = \infty$.

Sea $H = (V; E)$ un hipergrafo, una **componente conexa** es un conjunto maximal de vértices $X \subseteq V$ tal que para todo $x, y \in X$, $d(x, y) \neq \infty$. El **diámetro** $d(H)$ de H es definido por

$$d(H) = \text{máx}\{d(x, y) \colon x, y \in V\}.$$

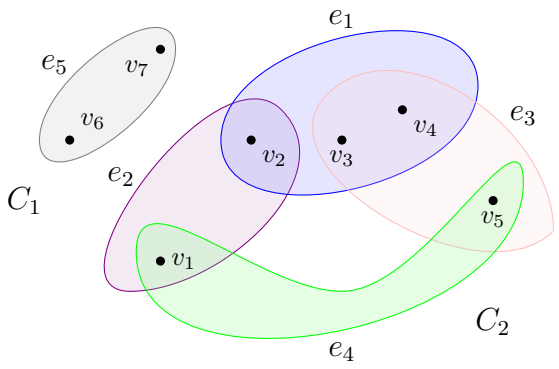


Figura 1.10: El Hipergrafo H'.

En la Figura 1.10, podemos observar que $P = v_1, e_4, v_5$ y $P' = v_1, e_2, v_2, e_1, v_4 e_3, v_5$ son caminos de v_1 a v_5 en el hipergrafo H'. La distancia de v_1 a v_5 es : $d(v_1, v_2) = 2$. Además, $P' + e_4, v_1$ es un ciclo en H'. El hipergrafo H' tiene dos componentes: C_1, C_2.

Un hipergrafo H es **completo** si $H = (V; E = P(V) - \{\emptyset\})$. Para $n = |V|$, un **hipergrafo completo k-uniforme** sobre $n \leq k \leq 2$ vértices es un hipergrafo el cual tiene todos los k subconjuntos de V como hiperaristas, es decir, $E = P_k(V)$, donde $P_k(V)$ es el conjunto de todos los $k-$ subconjuntos de V; es denotado por K_n^k.

Sea $H = (V; E)$ un hipergrafo, con $V = \{v_1, v_2, ..., v_n\}$ y $E = (e_1, e_2, ..., e_m)$ con

$$\bigcup_{i \in I} e_i = V,$$

(sin vértices aislados). Entonces H tiene una **matriz de incidencia** $n \times m$, $A = (a_{ij})$, donde:

$$a_{ij} = \begin{cases} 1 & \text{si } v_i \in e; \\ \\ 0 & \text{en otro caso.} \end{cases}$$

Esta matriz también puede escribirse como una matriz $m \times n$.

Capítulo 2

Clases de complejidad

En todas las áreas del conocimiento existe una cantidad considerable de problemas que son tratados con la ayuda de la computadora, lo que conlleva al desarrollo de software y a la necesidad de clasificar estos problemas de acuerdo a los recursos computacionales empleados para su solución.

Cualquier problema a tratar computacionalmente se puede clasificar principalmente dentro de tres grandes clases de problemas que son: problemas de decisión, problemas de conteo y problemas de optimización.

En cada clase de problemas existen subclasificaciones de acuerdo al tiempo y el espacio que ocupan los algoritmos empleados para su solución dentro de un modelo computacional.

Aquí se presentan los conceptos y resultados principales alrededor de las principales clases de complejidad de problemas en el modelo clásico computacional de las maquinas de Turing.

2.1. Clases de complejidad

Cuando se habla de complejidad es importante considerar que se hace referencia a una medida o parámetro que dá pauta a la clasificación de problemas, es decir, establecer una medida de complejidad para los problemas permite determinar las clases de complejidad.

Se utiliza como medida de complejidad el tiempo y el espacio. En este documento centramos nuestra atención al tiempo, considerando como modelo computacional las máquinas de Turing.

2.1.1. Clases de complejidad para problemas de decisión

- La clase **P** consiste de todos los problemas resolubles en tiempo polinomial por una MT de Turing determinista, vea [1]. Podemos decir que en esta clase se encuentran todos los problemas computacionalmente tratables.

  Formalmente la clase **P** se define de la siguiente manera:

  $$\mathbf{P} = \bigcup_{k \geq 0} DTIME[n^k].$$

  Donde $DTIME[n^k]$ es la clase de lenguajes reconocidos por una MT determinista en tiempo $O(n^k)$.

- La clase **NP** incluye problemas que son más "difíciles" que los que podemos hallar en la clase **P**. Podemos decir que la clase **NP** es una extensión de la clase **P** en el cual se reemplazan los algoritmos deterministas con algoritmos no deterministas.

  Formalmente la clase **NP** se define de la siguiente manera

  $$\mathbf{NP} = \bigcup_{k \geq 0} NTIME[n^k].$$

  Donde $NTIME[n^k]$ es la clase de lenguajes reconocidos por una MT no determinista en tiempo $O(n^k)$.

- La clase **NP**−difícil es la clase de todos los problemas tales que se puede transformar polinomialmente cualquier problema de **NP**, en uno de esta clase, es decir, dado

un problema H en **NP**-difícil se cumple que cualquier problema en **NP** se puede transformar polinomialmente en H.

- La clase **NP**-completo es la clase de todos los problemas que cumplen que pertenecen a la clase **NP** y todo problema en la clase **NP** se puede reducir a cualquier elemento de esta clase.

2.1.2. Clases de complejidad para problemas de conteo

- La clase **FP** denota la clase de funciones f tales que $f\colon \Sigma^* \to \mathbb{N}$ computables en tiempo polinomial determinista.

- La clase **#P** se define como:

$$\#\mathbf{P} = \{\varphi_M \colon M \in \mathcal{M}\}$$

donde $\mathcal{M}$ denota la clase de todas las MT no deterministas en tiempo polinomial y φ_M es el número de caminos computacionales de aceptación de M.

- Diremos que un problema de conteo $\#L$ pertenece a la clase $\#P-$difícil si todo problema en la clase $\#P$ se puede reducir a $\#L$, además si $\#L$ está en la clase $\#P$, entonces diremos que $\#L$ pertenece a la clase $\#P-$completo.

2.1.3. Complejidad parametrizada

Como sabemos, la teoría de la complejidad clásica analiza y clasifica problemas de acuerdo a la cantidad de recursos necesarios para resolver un problema, comúnmente los recursos que se consideran son el tiempo y el espacio. La teoría de la complejidad parametrizada no solo considera la complejidad

con respecto al tamaño de las entradas, también considera la complejidad en términos de un parámetro, el cual es un valor numérico dependiente de la entrada en una forma arbitraria. La noción central de esta teoría es la tratabilidad de parámetro fijo.

Comúnmente en teoría de la complejidad los problemas de decisión son considerados lenguajes sobre alfabetos finitos; con la intensión de distinguir estos problemas de los problemas parametrizados, llamaremos problemas clásicos a los conjuntos $Q \subset \Sigma^*$ de cadenas sobre Σ. Asumiremos que Σ es no vacío.

Definición 2.1. *Sea Σ un alfabeto finito.*

- *Una **parametrización** de Σ^* es un mapeo $\kappa\colon \Sigma^* \to \mathbb{N}$ que es computable en tiempo polinomial.*
- *Un **problema parametrizado** sobre Σ es una pareja (Q, κ) que consiste de un conjunto $Q \subset \Sigma^*$ de cadenas sobre Σ y una parametrización κ de Σ^*.*
- *Si (Q, κ) es un problema parametrizado sobre Σ, entonces las cadenas $x \in \Sigma^*$ seran llamadas **instancias** de Q, y los números $\kappa(x)$ se llamarán **parámetros**.*
- *A los lenguajes L del producto cartesiano $\Sigma^* \times \mathbb{N}$, se les llama **lenguajes parametrizados**.*
- *Sea $\langle x, k\rangle \in L$, donde L es un lenguaje parametrizado, llamaremos parámetro a la k que está en la segunda entrada de esta pareja.*

Ejemplo 2.2. *Sea SAT el conjunto de todas las fórmulas proposicionales satisfactibles, donde las fórmulas proposicionales se codifican como cadenas sobre un alfabeto finito Σ.*
Sea $\kappa\colon \Sigma^* \to \mathbb{N}$ la parametrización definida por:

$$\kappa(x) := \begin{cases} \text{número de variables de } x, & \text{si } x \in \ PROP \\ 1, & \text{en otro caso} \end{cases},$$

para $x \in \Sigma^*$. Denotamos el problema parametrizado (SAT, κ) por $p - SAT$.

Usualmente representamos un problema parametrizado (Q, κ) en la forma:

Instancia: $x \in \Sigma^*$.

Parámetro: $\kappa(x)$.

Problema: Decide si $x \in Q$.

Ejemplo 2.3. *El problema $p - SAT$ podría ser representado de la siguiente manera:*

Instancia: *Una fórmula proposicional α.*

Parámetro: *Número de variables de α.*

Problema: *Decide si α es satisfacible.*

En la teoría de la complejidad parametrizada es posible definir un parámetro que haga, en algunos casos, de un problema difícil uno tratable. Por esta razón es importante introducir la siguiente definición. La longitud de una cadena $x \in \Sigma^*$ es denotado por $|x|$.

Definición 2.4. *Sea Σ un alfabeto finito y $\kappa \colon \Sigma^* \to \mathbb{N}$ una parametrización.*

- *Dado un algoritmo $\mathcal{A}$ con un alfabeto de entrada Σ, diremos que un algoritmo es* ***parámetro fijo tratable*** *(pft-algoritmo) con respecto a κ si existe una función computable $f \colon \mathbb{N} \to \mathbb{N}$ y un polinomio $p \in \mathbb{N}_0[x]$ tal que para cada $x \in \Sigma^*$, el tiempo en el que corre $\mathcal{A}$ en una entrada x es a lo sumo*

$$f(\kappa(x)) \cdot p(|x|).$$

- *Un problema parametrizado (Q, κ) es* ***parámetro fijo tratable*** *si existe algún pft-algoritmo con respecto a κ que decida Q. Denotamos con, PFT la clase de todos los problemas parámetro fijo tratable.*

Ejemplo 2.5. *El problema SAT, parametrizado como se describió anteriormente, es decir p-SAT, es parámetro fijo tratable. Es verificable que el algoritmo de busqueda más "obvio" decide si una fórmula α de tamaño m con k variables es satisfactible en tiempo $O(2^k \cdot m)$.*

Definición 2.6. *Sea (Q, k) un problema parametrizado.*

- *(Q, k) es* ***fuertemente uniforme tratable de parámetro fijo*** *si y solo si existe un algoritmo $\mathcal{A}$, una constante c y una función $f\colon \mathbb{N} \to \mathbb{N}$ tal que, para cada x, k, el algoritmo $\mathcal{A}(\langle x, k\rangle)$ corre en un tiempo a lo más $f(k)|x|^c$ y $\langle x, k\rangle \in (Q, k)$ si y solo si $\mathcal{A}(\langle x, k\rangle) = 1$.*

- *(Q, k) es* ***uniforme tratable de parámetro fijo*** *si existe un algoritmo $\mathcal{A}$, una constante c y una función $f\colon \mathbb{N} \to \mathbb{N}$ tal que el tiempo de corrimiento de $\mathcal{A}(\langle x, k\rangle)$ es a lo más de $f(k)|x|^c$.*

- *(Q, k) es* ***no uniformemente tratable de parámetro fijo*** *si existe una constante c, una función $f\colon \mathbb{N} \to \mathbb{N}$ y una colección de procedimientos $\{\mathcal{A}_k\colon k \in \mathbb{N}\}$ tal que para cada $k \in \mathbb{N}$ y el tiempo de corrimiento de $\mathcal{A}_k(\langle x, k\rangle)$ es $f(k)|x|^c$ y $(\langle x, k\rangle) \in (Q, k)$ si y solo si $\mathcal{A}_k(\langle x, k\rangle) = 1$.*

Los problemas que se presentan en el Ejemplo 2.7 exhiben alguna forma de tratabilidad parametrizada de la Definición 2.6.

Ejemplo 2.7. *Consideremos los siguientes problemas.*

1. *Cubierta de vértices:*

   Instancia: *Un grafo $G = (V, E)$.*

Parámetro: *Un entero positivo k.*

Problema: *¿G tiene una cubierta de vértices de tamaño $\leq k$? (Una cubierta de vértices de un grafo G es una colección de vértices V' de G tales que para todas las arista v_1v_2 de G, ya sea $v_1 \in V'$ o $v_2 \in V'$.)*

El problema de la cubierta de vértices parametrizado por k puede ser resuelto en tiempo $2^k|G|$, véa [5][Teo 3.2.1].

El único algoritmo conocido el cual trabaja para toda k, y más aún, podemos calcular la constante, y por lo tanto, el tiempo de ejecución para cada k. Este es el comportamiento que se dió en la definición 2.6 y lo llamamos fuertemente uniforme tratable de parámetro fijo.

2. *El género de un grafo:*

   Instancia: *Un grafo $G = (V, E)$*

   Parámetro: *Un entero positivo k*

   Problema: *¿G tiene género k?.(Esto es, ¿se puede incrustar G sin que las aristas se crucen sobre una superficie con k mangos?)*

   Aún tenemos un único algoritmo ϕ para todo k, pero esta vez no tenemos forma de calcular la constante en el tiempo de ejecución. Simplemente sabemos que para cada k, el tiempo de ejecución de ϕ en la entrada $< G, k >$ es $O(|G|^3)$. Este comportamiento se denomina uniforme tratable de parámetro fijo.

3. *Número de enlace de un grafo:*

   Instancia: *Un grafo $G = (V, E)$*

   Parámetro: *Un entero positivo k*

> ***Problema:*** *¿Puede G ser incrustado en un $3-$espacio tal que el máximo tamaño de una colección de ciclos disjuntos vinvulados topológicamente está limitado por k?)*
>
> *Todo lo que conocemos es el exponente del tiempo de ejecución de los algoritmos. Para cada k, tenemos un algoritmo diferente (y desconocido) ejecutandose en $O|G|^3$ con constantes desconocidas. Este compartamiento se denomina no uniformemente tratable de parámetro fijo.*

Si consideramos la versión clásica de los problemas del Ejemplo 2.7, donde k no es fijo, entonces todos son ***NP-****difícil.*

2.1.4. Problemas de optimización

Los problemas de optimización estan en el corazón de la teoría de la complejidad, en la práctica generalmente deseamos optimizar alguna función objetivo (por ejemplo, encontrar un recorrido de menor costo), en lugar de resolver un problema de decisión relacionado. Además, incluso si demostramos que algún problema no tiene solución, a menudo es aceptable encontrar una solución aproximada a alguna relación de rendimiento aceptada [7].

En general un problema de optimización consiste de un conjunto de instancias, las cuales toman un valor bien especificado. Cada intancia es asociada con un conjunto de soluciones tales que cada solución tiene un valor dada la instancia. Resolver el problema de optimización es encontrar para cada instancia dada una mejor (u optimal) solución, la cual debe tener ya sea el más grande o más pequeño valor asociado, dependiendo de la descripción del problema de optimización.

Definición 2.8. *Un* ***problema de optimización*** *Q es una $4-$tupla, $Q = \{I_Q, S_Q, f_Q, opt_Q\}$, donde:*

- *I_Q es el conjunto de* ***instancias de entrada****. I_Q es reconocible en tiempo polinomial.*

- $S_Q(x)$ *es el conjunto de* ***soluciones viables*** *para la entrada* $x \in I_Q$.
- $f_Q(x, y) \in \mathbb{N}$ *es la* ***función objetivo*** *tal que para cada par* $x \in I_Q$ *y* $y \in S_Q(x)$, $f_Q(x, y)$ *es un número real.*
- $opt_Q \in \{max, min\}$, *especifica que el problema es un problema máximo o problema mínimo.*

Una ***solución óptima*** *para una instancia de entrada* $x \in I_Q$ *es una solución viable* $y \in S_Q(x)$ *tal que*

$$f_Q(x, y) = opt_Q\{f_Q(x, z) \colon z \in S_Q(x)\}.$$

Escribimos:

$$opt_Q(x) = opt_Q\{f_Q(x, z) \colon z \in S_Q(x)\}$$

Veamos a cotinuación un problema de optimización. En la programación de rutas o en la comunicación de la red, frecuentemente necesitamos encontrar el camino más corto de una posición dada a otra posición específica. Este problema es formulado como el problema del camino más corto.

Ejemplo 2.9. *El problema del camino más corto.*

- I_Q*: El conjunto de todos los grafos ponderados* G *con dos vértices especificados* u *y* v *en* G.
- S_Q*:* $S_Q(G)$ *es el conjunto de todos los caminos que conectan a* u *y* v *en* G.
- f_Q*:* $f_Q(G, u, v, P)$ *es la longitud del camino* P *(medido por el peso de sus aristas) que conectan* u *y* v *en* G.
- opt_Q*:* $\min f_Q$.

Capítulo 3

Autómatas

Los autómatas finitos son estructuras matemáticas que representan modelos de cómputo abstractos diseñados para reconocimiento de palabras. Es útil considerar un autómata finito como una máquina virtual que consiste de una semicinta infinita dividida en celdas que almacenan un sólo símbolo de algún alfabeto dado y un sistema de control aunado a un cabezal que apunta hacia las celdas de la cinta, véa la Fig. 3.1.

Definición 3.1. *Un* ***autómata finito determinista*** *puede ser considerado como una quíntupla $M = (Q, \Sigma, \delta, q_0, F)$, donde Q es un conjunto finito, llamado conjunto de estados, Σ es un conjunto finito de símbolos, llamado alfabeto, q_0 representa el estado inicial y F es subconjunto de Q, llamado conjunto de aceptación y δ es una función con dominio $Q \times \Sigma$ y rango Q, llamada* ***función de transición****.*

La descripción del funcionamiento de M en términos de lo dicho anteriormente es como sigue:

I. Inicialización

1. La cadena $\sigma \in \Sigma^*$ se coloca en la cinta de M con un símbolo en cada celda.

2. El cabezal de la máquina se apunta a la celda que contiene el símbolo de más a la izquierda de la cadena σ.

3. El estado actual de M pasa a ser q_0.

II. Ejecución

1. Leer símbolo actual (el apuntado por el cabezal). Si el cabezal apunta a una celda vacía, la máquina M termina su ejecución aceptando la cadena σ en caso de que el estado actual sea un estado final y rechazando la cadena σ en caso contrario.

2. Se calcula el estado siguiente a partir del estado actual y del símbolo del cabezal mediante la función de transición, es decir, el estado siguiente es $\delta(edo\_actual, sim\_cabezal)$.

3. El cabezal de M se mueve hacia la derecha.

4. El estado siguiente se convierte en el estado actual y M vuelve al paso 1.

Los pasos 2 y 3 pueden ser descritos mediante la siguiente expresión. Para la cadena $\sigma \in \Sigma^*$ tal que $\sigma = a\gamma$ con $a \in \Sigma$ y $\gamma \in \Sigma^*$,

$$< q, a\gamma > \vdash < \delta(q, a), \gamma >$$

Formalmente M acepta la cadena σ si para algún estado $q \in F$ se cumple

$$< q_0, \sigma > \vdash^* < q, \lambda >$$

donde $\vdash^*$ es la cerradura transitiva de la relación $\vdash$ y λ es la cadena vacía. El conjunto de las cadenas $\sigma \in \Sigma^*$ aceptadas por el autómata M se denota por $L(M)$, esto es

$$L(M) = \{\sigma \in \Sigma^* : M \text{ acepta } \sigma\}.$$

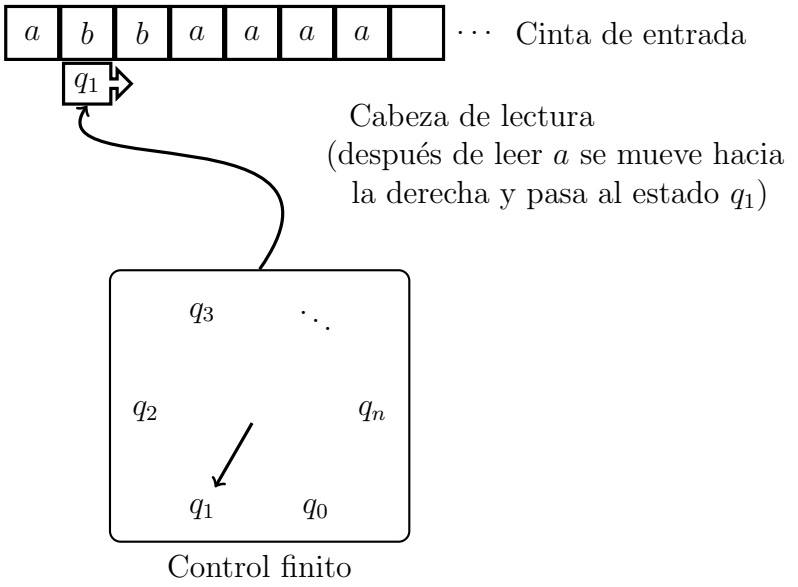


Figura 3.1: Representación análoga al mecanismo de MT para autómatas deterministas.

A continuación se presenta un ejemplo de un autómata finito determinista.

Ejemplo 3.2. *Sea* $M = \{Q, \Sigma, \delta, q_0, F\}$ *el autómata finito determinista el cual rechaza cadenas que no tienen exactamente tres 1's, donde* $Q = \{q_0, q_1, q_2, q_3, q_f\}$, $\Sigma = \{0, 1\}, F = \{q_f\}$ *y* δ *está dado por la siguiente tabla de transición.*

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

A continuación se muestran el diagrama de transición del autómata M.

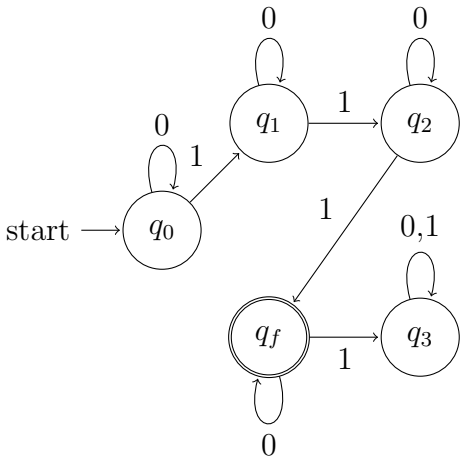


Figura 3.2: Diagrama de transición del autómata M.

Podemos observar que dado un autómata finito determinista M y una cadena σ el problema de determinar si $\sigma \in L(M)$ es (fuertemente uniformemente) parámetro fijo tratable, vea [5].

> ACEPTACIÓN POR AUTÓMATA FINITO DETERMINISTA (AAFD)
>
> *Entrada*: Una cadena $\sigma \in \Sigma^*$.
>
> *Parámetro*: M.
>
> *Pregunta*: ¿$\sigma \in L(M)$?.

La siguiente proposición, cuya demostración es trivial, completa la observación anterior.

Proposición 3.3. *El problema AAFD es soluble en tiempo $O(|\sigma|)$, y así es (fuertemente uniformemente) PFT.*

Los autómatas finitos no deterministas, a diferencia de los autómatas deterministas, cumplen que para cualquier posición en la que se encuentre el autómata hay uno o más posibles

caminos de cómputo, es decir, en un autómata finito no determinista se generaliza la relación de transición δ de una función a una multifunción.

Definición 3.4. *Un* ***autómata finito no determinista*** *es una quíntupla $M = (Q, \Sigma, \Delta, S, F)$, donde Q, F y Σ están definidos de la misma manera que para los autómatas finitos deterministas, S es llamado el conjunto de estados iniciales y $\Delta \subseteq Q \times \Sigma \times Q$ es una relación llamada* ***relación de transición****.*

Se puede interpretar la acción de una máquina M empezando en el estado q_i leyendo el símbola a, "pudiendo moverse a cualquiera de los estados q_k, con $< q_i, a, q_k > \in \Delta$". Formalmente escribimos esta acción mediante la siguiente expresión. Para la cadena $\sigma \in \Sigma^*$ tal que $\sigma = a\gamma$ con $a \in \Sigma$ y $\gamma \in \Sigma^*$,

$$< q_i, a\gamma > \vdash < q_k, \gamma > .$$

Tenemos que $\sigma \in L(M)$ si y solo si

$$< q_0, \sigma > \vdash^* < q, \lambda >,$$

para algún $q \in F$ y donde $\vdash^*$ denota la cerradura transitiva reflexiva de $\vdash$. La interpretación de la definición de $\vdash^*$ es que se acepta σ siempre que exista algún camino para terminar en un estado de aceptación, esto es, siempre que exista algún camino computable de M que acepte σ. Puede haber muchos caminos de computo posibles para la entrada σ, tal vez solo uno lleve a la aceptación. En este caso, se acepta σ.

A continuación se muestra un ejemplo de un autómata finito no determinista.

Ejemplo 3.5. *Sea $M = (Q, \Sigma, \Delta, S, F)$, donde el conjunto $Q = \{q_0, ..., q_4, q_f\}$, $\Sigma = \{0, 1\}$, $S = \{q_0\}$, $F = \{q_f\}$ y Δ es dado por la siguiente tabla de transiciones:*

	0	1
q_0	q_0	q_0, q_1
q_1	q_2	q_2
q_2	q_3	q_3
q_3	q_4	q_4
q_4	q_f	q_f

El diagrama de transiciones correspondiente es:

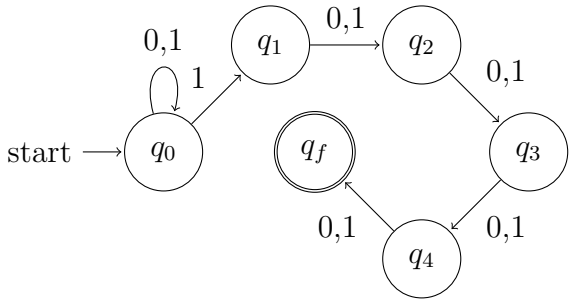


Figura 3.3: Diagrama de transición del autómata no determinista M.

Podemos observar que el autómata M acepta las cadenas de ceros y unos tales que su quinto símbolo de derecha a izquierda es 1.

Se puede pensar que el poder de computación de un autómata finito no determinista es mayor que el de un autómata finito determinista, sin embargo el siguiente Teorema debido a Rabin y Scott's [13], establece que ambos autómatas tienen el mismo poder de cómputo. Decimos que dos autómatas M_1 y M_2 son **equivalentes** si $L(M_1) = L(M_2)$.

Teorema 3.6. *Todo autómata finito no determinista es equivalente a uno determinista. Si M es una máquina no determinista con n estados, entonces podemos computar en $O(2^n)$ pasos una máquina determinista M' con a lo más 2^n estados. El orden O sólo depende de la cardinalidad de Σ.*

En el caso del autómata del ejemplo 3.5, su autómata determinístico correspondiente tendría almenos 2^5 estados. Por lo que en el siguiente ejemplo consideramos un autómata no determinista similar más simple.

Ejemplo 3.7. *Sea* $N = (Q, \Sigma, \Delta, S, F)$*, donde* $Q = \{q_0, q_1, q_f\}$*,* $\Sigma = \{0, 1\}$*,* $S = \{q_0\}$*,* $F = \{q_f\}$ *y* Δ *es dado por la siguiente tabla de transiciones:*

	0	1
q_0	q_0	q_0, q_1
q_1	q_f	q_f

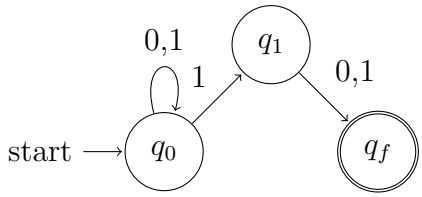


Figura 3.4: Diagrama de transiciones del autómata no determinista N.

Construimos el autómata determinista $N' = (Q', \Sigma', \delta, S', F')$ de la siguiente manera:

- $Q' = \mathcal{P}(Q)$,
- $\Sigma' = \Sigma$,
- $S' = \{\{q_0\}\}$
- $F' = \{\{q_f\}, \{q_0, q_f\}, \{q_1, q_f\}, \{q_0, q_1, q_f\}\}$,

y δ está dado por la siguiente tabla de transiciones

	0	1
$\emptyset$	$\emptyset$	$\emptyset$
$\{q_0\}$	$\{q_0\}$	$\{q_0, q_1\}$
$\{q_1\}$	$\{q_f\}$	$\{q_f\}$
$\{q_f\}_f$	$\emptyset$	$\emptyset$
$\{q_0, q_1\}$	$\{q_0, q_f\}$	$\{q_0, q_1, q_f\}$
$\{q_0, q_f\}_f$	$\{q_0\}$	$\{q_0, q_1\}$
$\{q_1, q_f\}_f$	$\{q_f\}$	$\{q_f\}$
$\{q_0, q_1, q_f\}_f$	$\{q_0, q_f\}$	$\{q_0, q_1, q_f\}$

Podemos simplificar la tabla de transiciones observando que los estados $\emptyset$, $\{q_f\}$, $\{q_1\}$ y $\{q_1, q_f\}$ son inalcanzables desde el estado $\{q_0\}$ siguiendo transiciones 0,1. La tabla simplificada queda de la siguiente manera:

	0	1
$\{q_0\}$	$\{q_0\}$	$\{q_0, q_1\}$
$\{q_0, q_1\}$	$\{q_0, q_f\}$	$\{q_0, q_1, q_f\}$
$\{q_0, q_f\}_f$	$\{q_0\}$	$\{q_0, q_1\}$
$\{q_0, q_1, q_f\}_f$	$\{q_0, q_f\}$	$\{q_0, q_1, q_f\}$

El diagrama de transiciones del autómata determinista N' finalmente queda expresado como sigue.

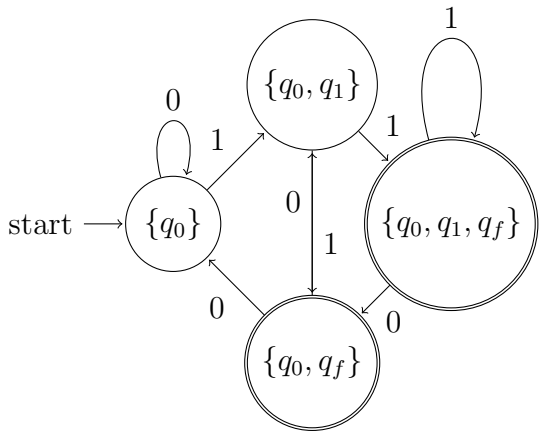


Figura 3.5: Diagrama de transiciones del autómata determinista N'.

La demostración que a continuación presentamos corresponde a la construcción del último autómata del ejemplo 3.7, ésta es conocida como la construcción de Thompson. Aquí se observa que no es necesario utilizar todos los estados de $\mathcal{P}(Q)$.

Demostración. **De Teorema 3.6**.
Dado $M = (Q, \Sigma, \Delta, q_0, F)$ un autómata finito no determinista, definimos el autómata determinista $M' = (Q', \Sigma, \delta, S, F')$ a partir de M, de la siguiente manera. El conjunto de estados Q' consiste de los conjuntos:

$$Q_0 = S = \{q_0\},$$

$$Q_1 = \{q \in Q : \exists a \in \Sigma, < q_0, a > \vdash < q, \lambda >\},$$

$$Q_2 = \bigcup_{q \in Q_1} \{r \in Q : \exists b \in \Sigma, < q, b > \vdash < r, \lambda >\},$$

$$\vdots \quad \vdots \qquad \vdots$$

$$Q_n = \bigcup_{q \in Q_{n-1}} \{s \in Q : \exists c \in \Sigma, < q, c > \vdash < s, \lambda >\}.$$

El procedimiento anterior se sigue hasta que ya no se generen nuevos estados.

Se observa que la longitud de los caminos está acotada por $n = |Q|$ y el proceso se detiene después de $O(2^n)$ pasos ya que a lo más se pueden generar 2^n estados distintos de $Q = \{q_0, q_1, ..., q_{n-1}\}$. Los estados de aceptación de M' son los subconjuntos de Q que contienen un estado aceptado de M, definido por $F' = \{Q_j : Q_j \cap F \neq \emptyset\}$, y, si $Q_i \subset Q$ y $a \in \Sigma$, definimos

$$\delta(Q_i, a) = Q_j, \text{ para } j = \{1, ..., n\}.$$

Veamos que $L(M) = L(M')$, esto es, que si $\sigma \in L(M)$ entonces $\sigma \in L(M')$. En efecto, si $< q_0, \sigma > \vdash^*_M < r, \lambda >$, entonces $r \in F$ y $< Q_0, \sigma > \vdash^*_{M'} < R, \lambda >$, para algún estado R que contenga al estado r de Q.

□

Como en el caso de los autómatas finitos deterministas tenemos el siguiente problema.

ACEPTACIÓN POR AUTÓMATA FINITO NO DETERMINISTA (AAFND)

Entrada: Una cadena $\sigma \in \Sigma^*$

Parámetro: M

Pregunta: ¿$\sigma \in L(M)$?

El siguiente corolario es consecuencia del teorema 3.6.

Corolario 3.8. *El problema AAFND es soluble en tiempo $O(n)$, y así es (fuertemente uniformemente) PFT.*

En ciertos casos es conveniente extender el modelo de un autómata finito no determinista a un autómata finito no determinista con $\lambda-$transiciones.

Definición 3.9. *Una $\lambda-$transición es una transición con etiqueta la cadena vacía λ.*

Un autómata finito no determinístico M con $\lambda-$transiciones puede pasar de un estado a otro sin leer un solo símbolo, como se muestra en el siguiente ejemplo.

Ejemplo 3.10. *El siguiente diagrama de transiciones corresponde a un autómata no determinista con $\lambda-$transiciones que acepta cadenas de $a's$ de longitud $3n$ o $5n$, con $n \in \mathbb{N}$.*

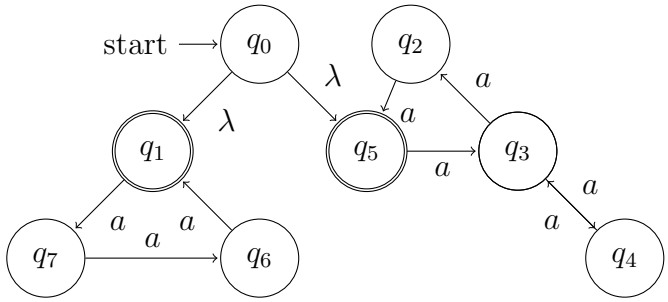


Figura 3.6: Autómata no determinista con $\lambda-$transiciones.

Como en el caso de los autómatas finitos no determinísticos versus los autómatas finitos deterministas, no se gana mayor poder computacional con estos autómatas. El siguiente teorema de McNaughton, Yamada, Rabin y Scott [13, 9] confirma nuestro comentario.

Teorema 3.11. *Cualquier autómata finito no determinista con $\lambda-$transiciones es equivalente a uno sin $\lambda-$transiciones. Además existe una construcción, que cuando se aplica a M, un autómata finito no determinista con $\lambda-$transiciones, produce un autómata no determinístico equivalente M' con $O(n)$ estados en $O(n)$ pasos.*

Para $A, B \subset \Sigma^*$, consideremos las siguientes definiciones:

$$A \cup B = \{x \colon x \in A \text{ o } x \in B\} \qquad \textbf{unión}.$$

$$A \cap B = \{x \colon x \in A \text{ y } x \in B\} \qquad \textbf{intersección}.$$

$$\sim A = \{x \in \Sigma^* \colon x \notin A\} \qquad \textbf{complemento}.$$

$$AB = \{xy \colon x \in A \text{ y } y \in B\} \qquad \textbf{concatenación}.$$

$$\begin{aligned} A^* &= \{x_1 x_2 \cdots x_n \colon n \geq 0 \text{ y } x_i \in A, 1 \leq i \leq n\} \\ &= A^0 \cup A^1 \cup A^2 \cup A^3 \cup \cdots \qquad \textbf{estrella}. \end{aligned}$$

Es importante no confundir la concatenación de conjuntos con la concatenación de cadenas. La operación concatenación toma dos cadenas x, y y crea una nueva cadena xy poniendolas juntas de un extremo a otro. La cadena xy es llamada la concatenación de x y y. Notemos que xy y yx son diferentes en general. Veamos algunas propiedades de la concatenación.

- La concatenación es asociativa: $(xy)z = x(yz)$;
- La cadena nula ϵ es una identidad para la concatenación: $\epsilon x = x\epsilon = x$;
- $|xy| = |x| + |y|$.

Un caso especial de la última ecuación es

$$a^m a^n = a^{m+n} \text{ para toda } m, n \geq 0.$$

De la teoría de Álgebra superior, sabemos que un monoide es cualquier estructura algebraica que consiste de un conjunto con una operación binaria asosiativa y una identidad para esa operación. De esta manera, el conjunto Σ^* con la concatenación de cadenas como operación binaria y ϵ como la identidad es un monoide.

- Un **prefijo** de una cadena x es una subcadena inicial de x; es decir, una cadena y para la que existe una cadena z tal que $x = yz$. Por ejemplo, *abaab* es un prefijo de *abaababa*. La cadena nula es el prefijo de cualquier cadena y cada cadena es es un prefijo de sí misma. Un prefijo y de x es un prefijo propio de x si $y \neq \epsilon$ y $y \neq x$.

- Considerando el conjunto concatenación AB, tenemos que $z \in AB$ si y solo si z puede ser escrito como una concatenación de dos cadenas x y y, donde $x \in A$ y $y \in B$. Por ejemplo, si $A = \{a, ab\}$ y $B = \{b, ba\}$, tenemos que $AB = \{ab, aba, abb, abba\}$. Al formar una concatenación de conjuntos, se incluyen todas las cadenas que pueden ser obtenidas de esta forma. Observemos que AB y BA son conjuntos diferentes en general. Así, considerando A y B del ejemplo anterior, tenemos que $BA = \{ba, bab, baa, baab\}$.

Veremos más adelante que la siguiente definición es importante para la eficiencia algorítmica.

Definición 3.12. *Si L es un conjunto de cadenas que reconoce un autómata finito, entonces decimos que L es de* ***estado finito****.*

Definición 3.13. *Sea $\mathcal{L}$ el conjunto de lenguajes aceptados por autómatas finitos. Si $L \in \mathcal{L}$, definimos L^* como el lenguaje formado de concatenaciones de cadenas en L. La operación $*\colon L \to L^*$ es llamada* ***operación estrella****.*

El siguiente teorema expresa algunas propiedades de los conjuntos de estado finito.

Teorema 3.14. *Sea $\mathcal{L}$ el conjunto de lenguajes aceptados por autómatas finitos. Entonces $\mathcal{L}$ es cerrado bajo las siguientes operaciones:*

(i) unión

(ii) complemento

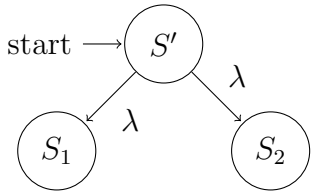
(iii) intersección

(iv) concatenación

(v) estrella .

Demostración. Sean $M_i = (Q_i, \Sigma_i, S_i, \Delta_i, F_i)$ para $i = 1, 2$, autómatas sobre Σ. Renombrando los estados podemos suponer que $Q_i \cap Q_j = \emptyset$.

(i) Se construye un autómata partiendo de M_1 y M_2, que se denota por $M_1 \cup M_2$, agregando un nuevo estado inicial S' con una $\lambda-$transición hacia cada uno de los estados iniciales de los autómatas M_1 y M_2. El conjunto de estados de aceptación de $M_1 \cup M_2$ es $F_1 \cup F_2$.

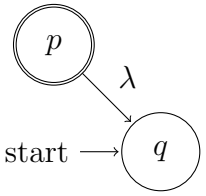


(ii) Veamos que $\overline{L(M_1)}$ es de estado finito. Para esto, basta con elegir $M' = (Q_1, \Sigma_1, S_1, \Delta_1, Q_1 - F_1)$. Claramente $\overline{L(M_1)} = L(M')$.

(iii) Se sigue de $L(M_1) \cap L(M_2) = \overline{\overline{L(M_1)} \cup \overline{L(M_2)}}$.

(iv) Aquí basta unir el autómata M_1 con el autómata M_2 con $\lambda-$transiciones entre el conjunto de estados de aceptación y el conjunto de estados iniciales del autómata M_2. En términos formales el autómata resultante se define por $M' = (Q_1 \cup Q_2, \Sigma_1 \cup \Sigma_2, S_1, \Delta', F_1 \cup F_2)$, donde

$$\Delta' = \Delta_1 \cup \Delta_2 \cup \{(p, \lambda, q) : p \in F_1 \wedge q \in S_2\}$$



(v) Si M_1 es el autómata que acepta el lenguaje L entonces

$$M' = (Q_1, \Sigma_1, S_1, \Delta', F_1),$$

donde

$$\Delta' = \Delta_1 \cup \{(p, \lambda, q) : p \in F_1 \wedge q \in S_1\}$$

es el autómata que reconoce L^*.

□

Veamos algunos ejemplos de la operaciones ya mencionadas:

Ejemplo 3.15. *Si* $L_1, L_2 \in \mathcal{L}$, *donde* $L_1 = \{11, 00, 10\}$ *y* $L_2 = \{1, 0\}$, *entonces:*

- ***unión:***
$$L_1 \cup L_2 = \{1, 0, 00, 10, 11\}.$$

- ***concatenación:***
$$L_1 L_2 = \{111, 110, 001, 000, 101, 100\}.$$

- ***estrella:*** $L^0 = \{\epsilon\}$, $L^1 = L$, $L^2 = LL$. *Si* $L = \{0, 11\}$, *entonces*
$$L^2 = \{00, 011, 110, 1111\}$$

3.1. Lenguajes Regulares

Otro método sintáctico bien conocido para los lenguajes de estado finito es a través de *patrones de coincidencia* y la clase de *lenguajes regulares*. Un lenguaje es regular si puede ser representado vía una cadena finita de símbolos tomados de Σ, junto con la unión, concatenación, estrella (estrella de Klein) y el vacío. Las siguientes definiciones formalizan el concepto de lenguaje regular.

Definición 3.16. *Una* ***expresión regular*** *sobre Σ es una expresión sobre el alfabeto Σ, $\cup$, $(,)$ y $\emptyset$ definida inductivamente como:*

1.- Son expresiones regulares el $\emptyset$ y todo elemento de Σ.

2.- Si α y β son expresiones regulares, entonces también lo son $(\alpha \cup \beta)$ y $(\alpha\beta)$.

3.- Si α es una expresión regular, entonces también lo es $(\alpha)^*$.

4.- α es una expresión regular si y solo si es obtenido de los casos anteriores.

Denotemos por $EXP(\Sigma)$ el conjunto de expresiones regulares sobre Σ.

Ejemplo 3.17. *Son elementos de $EXP(\{a, b\})$ las cadenas: a, b, $(a \cup b)$, (ab), $(a \cup b)(ab)$, $(ab)^*$, $(ab)^*(a \cup b)^*$, entre una infinidad.*

A cada expresión regular sobre un alfabeto podemos asociarle un lenguaje como se establece en la siguiente definición.

Definición 3.18. *Dada $\alpha \in EXP(\Sigma)$, definimos el lenguaje $L(\alpha)$ como sigue:*

1.- $L(\emptyset) = \emptyset$.

2.- $L(\alpha) = \{\alpha\}$ si $\alpha \in \Sigma$.

3.- $L(\alpha \cup \beta) = L(\alpha) \cup L(\beta)$, si $\alpha,\ \beta \in EXP(\Sigma)$.

4.- $L(\alpha\beta) = L(\alpha)L(\beta)$, si $\alpha,\ \beta \in EXP(\Sigma)$.

5.- $L(\alpha^*) = (L(\alpha))^*$, si $\alpha \in EXP(\Sigma)$.

Definición 3.19. *Un lenguaje L es **regular** si $L = L(\alpha)$ para algún $\alpha \in EXP(\Sigma)$.*

Ejemplo 3.20. *Si $\Sigma = \{a, b\}$, entonces:*

$$\begin{aligned} L(a(a \cup b)^*b) &= L(a)L((a \cup b)^*b) \\ &= \{a\}L(a \cup b)^*L(b) \\ &= \{a\}(L(a \cup b))^*\{b\} \\ &= \{a\}(L(a) \cup L(b))^*\{b\} \\ &= \{a\}((\{a\} \cup \{b\})^*\{b\} \\ &= \{a\}\{a, b\}^*\{b\}. \end{aligned}$$

Observe que $L(a(a \cup b)^*b)$ es el lenguaje que consiste de todas las cadenas sobre Σ que empiezan con el símbolo a y terminan con el símbolo b.

Teorema 3.21. *Todo lenguaje finito es regular.*

Demostración. Es claro por la operación de concatenación que toda cadena α es una expresión regular, por lo que $L_\alpha = \{\alpha\}$ es un lenguaje regular. Supongamos que L es un lenguaje tal que $L = \{\alpha_1, \ldots, \alpha_n\}$. Sea $L_i = \{\alpha_i\}$ para $i = 1, \ldots, n$, entonces de la operación unión se sigue que $L = L_1 \cup \cdots \cup L_n$ es un lenguaje regular. □

El siguiente teorema clásico debido a Kleene [8], establece que la clase de los lenguajes de estado finito es igual a la clase de los lenguajes regulares.

Teorema 3.22. *Un lenguaje L es de estado finito si y solo si L es un lenguaje regular.*

Demostración. Si L es un lenguaje regular, entonces es fácil construir un autómata M que acepte la cadena vacía $\emptyset$ y cualquier símbolo $a \in \Sigma$. Si L no es construido partiendo de cadenas de longitud ≤ 1, entonces éste se construye utilizando las condiciones de la 3 a la 5 de la definición 3.18.

Ahora de la definición 3.16, cualquier $\sigma \in L$ es obtenida mediante las operaciones de unión, concatenación y estrella. Luego por el Teorema 3.14, se sigue que L es de estado finito (por ejemplo, si existe un autómata M_1 que acepta todas las cadenas de L del tipo α y un autómata M_2 que acepta todas las cadenas de L del tipo β, entonces el autómata resultante de la combinación de los autómatas unión, concatenación y estrella reconoce a todas las expresiones γ de L que son derivables de las expresiones de los tipos α y β.

Para la condición necesaria, supongamos que L es un lenguaje de estado finito. Sea $M = (Q, \Sigma, \delta, S, F)$ un autómata finito determinista tal que $L = L(M)$. Veamos que existe un lenguaje regular R tal que $R = L(M)$.

Supongamos que $Q = \{q_1, q_2, \ldots, q_n\}$ y $S = \{q_1\}$. Se define $R(i, j, k)$, para $i, j \in \{1, \ldots, n\}$ y $k \in \{1, \ldots, n+1\}$, como el conjunto de cadenas en Σ^* derivables de M desde el estado q_i al estado q_j sin usar ningún estado q_m para $m \geq k$. Formalmente tenemos:

$$R(i,j,k) = \{\sigma \in L(M) : \langle q_i, \sigma\rangle \vdash^* \langle q_j, \lambda\rangle \wedge \forall\gamma[\langle q_i, \sigma\rangle \vdash^* \langle q_m, \gamma\rangle \Rightarrow (m < k \vee (\gamma = \lambda \wedge m = j) \vee (\gamma = \sigma \wedge m = i))]\}.$$

Observemos que

$$R(i,j,n+1) = \{\sigma \in L(M) : \langle q_i, \sigma\rangle \vdash^* \langle q_j, \lambda\rangle\}.$$

Entonces es claro que

$$L(M) = \bigcup_{i,j}\{R(i,j,n+1) : q_i \in S \wedge q_j \in F\}.$$

□

Si se demuestra que $R(i, j, n + 1)$ es un lenguaje regular, entonces $L(M)$ será un lenguaje regular porque del Teorema 3.14, la unión de lenguajes regulares es un lenguaje regular. El siguiente lema completa la demostración del Teorema.

Lema 3.23. *Para todo i, j, k, $R(i, j, k)$ es regular.*

Demostración. La demostración se realiza por inducción sobre k. Para $k = 1$, tenemos:

$$R(i,j,1) = \{\sigma \in L(M) : \langle q_i, \sigma\rangle \vdash^* \langle q_j, \lambda\rangle \wedge \forall\gamma[\langle q_i, \sigma\rangle \vdash^* \langle q_m, \gamma\rangle \Rightarrow (m < 1 \vee (\gamma = \lambda \wedge m = j) \vee (\gamma = \sigma \wedge m = i))]\}.$$

$$= \{\sigma \in L(M) : \langle q_i, \sigma\rangle \vdash^* \langle q_j, \lambda\rangle \wedge \forall\gamma[\langle q_i, \sigma\rangle \vdash^* \langle q_m, \gamma\rangle \Rightarrow \gamma = \lambda \wedge m = j]\}.$$

Esto es:

$$R(i,j,1) = \{\sigma : \langle q_i, \sigma\rangle \vdash \langle q_j, \lambda\rangle\} = \{\sigma : \delta(q_i, \sigma) = q_j\}.$$

Así $R(i, j, 1)$ es finito, luego por el Teorema 3.21 tenemos que $R(i, j, 1)$ es regular. Por otra parte, se afirma que:

$$R(i,j,k+1) = R(i,j,k) \cup R(i,k,k)R(k,k,k)^*R(k,j,k).$$

Si tal es el caso, por la hipótesis de inducción y las propiedades de unión, concatenación y estrella, la prueba habrá concluido. Para esto, observemos primero que $R(i, j, k + 1)$ es el conjunto de cadenas σ con $\langle q_i, \sigma\rangle$ moviendose a $\langle q_j, \lambda\rangle$ sin pasar por ningún estado q_m con $m \geq k + 1$. Ahora, para ir del estado q_i al estado q_j sin involucrar estados q_m con $m \geq k + 1$, tenemos las siguientes opciones.

1. Ir de q_i a q_j utilizando estados q_m con $m < k$. Esto es el conjunto $R(i, j, k)$.

2. En otro caso, ir de q_i a q_j utilizando q_k, y esto se puede hacer mediante las siguientes sub-opciones. Véase la Figura 3.7.

   (a) Ir de q_i a q_k usando q_m con $m < k$ (conjunto R(i,k,k)).

   (b) Ir de q_k a q_k varias veces usando estados q_m con $m < k$ (conjunto $R(k, k, k)^*$).

   (c) Ir de q_k a q_j utilizando estados q_m con $m < k$ (conjunto $R(k, j, k)$).

□

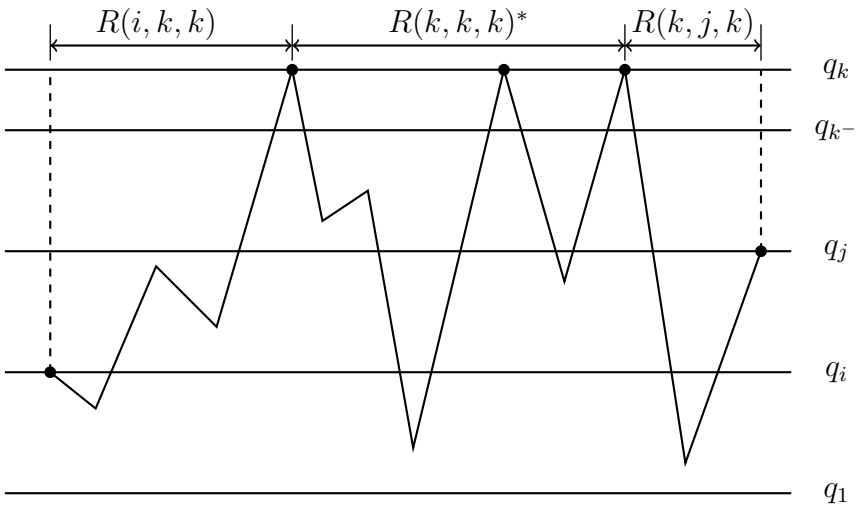


Figura 3.7: Caso 2 del Lema 3.23.

Definición 3.24. *Llamaremos a una relación R sobre Σ^* una* ***congruencia derecha*** *(con respecto a la concatenación) si y solo si:*

(a) R es una relación de equivalencia sobre Σ^*

(b) para todo $x, y \in \Sigma^*$ se cumple

$$xRy \Leftrightarrow (\forall z \in \Sigma^*, xzRyz)$$

Intuitivamente, las congruencias derechas deben coincidir en todas las extensiones. De manera similar podemos definir una congruencia izquierda, reemplazando $xzRyz$ en la Definición 3.24 inciso (b) por $zxRzy$.

Ejemplo 3.25. *Sea R la relación sobre Σ^* dada por xRy si y solo si x, y inician con la misma letra. Es claro que R es una relación de equivalencia y que cumple con (b) de la definición, por lo tanto R es una congruencia derecha. Observe que R no es una congruencia izquierda.*

Decimos que R es **una congruencia** si esta satisface ser una congruencia izquierda y una congruencia derecha.

Definición 3.26. *La **congruencia derecha canónica inducida** por un lenguaje L es la relación $\sim_L$ sobre Σ^* definida por*

$$x \sim_L y \Leftrightarrow (\forall z \in \Sigma^*)(xz \in L \Leftrightarrow yz \in L)$$

De manera análoga se define la congruencia izquierda canónica inducida por un lenguaje.

Veamos que la relación canónica efectivamente es una relación de equivalencia. Es claro que $x \sim_L x$, puesto que si $x \in \Sigma^*$, entonces $(\forall z \in \Sigma^*, xz \in L \Leftrightarrow xz \in L)$. Ahora si $x \sim_L y$ entonces $(\forall z \in \Sigma^*, xz \in L \Leftrightarrow yz \in L)$ es equivalente a $(\forall z \in \Sigma^*, yz \in L \Leftrightarrow xz \in L)$. Finalmente, si $x \sim_L y$ y $y \sim_L w$, entonces $(\forall z \in \Sigma^*, xz \in L \Leftrightarrow yz \in L)$ y $(\forall z \in \Sigma^*, yz \in L \Leftrightarrow wz \in L)$ implica $(\forall z \in \Sigma^*, xz \in L \Leftrightarrow yz \in L \Leftrightarrow wz \in L)$.

Definición 3.27. *Una relación de equivalencia sobre Σ^* es de **índice finito** si ésta solo tiene un número finito de clases de equivalencia sobre Σ^*.*

Ejemplo 3.28. *Sea $\Sigma = \{a\}$ y el lenguaje $L = \{a\}$, entonces $[a] = \{x \in \Sigma^* : x \sim_L a\}$ y como $\forall z \in \Sigma^*$, $xz \in L \Leftrightarrow az \in L$, entones $z = \lambda$, de donde $[a] = \{a\}$. Ahora sea $x_0 \in \Sigma^*$, con $x_0 \neq a$, entonces $[x_0] = \Sigma^* - \{a\}$. Esto es porque $xz \in L$ y $x_0 z \in L$ son simultáneamente falsos para todo $z \in \Sigma^*$. Aquí la relación $\sim_L$ es de índice finito.*

Ejemplo 3.29. *Sea $\Sigma = \{a, b\}$ y consideremos el lenguaje $L = \{x \in \Sigma^* : x$ tiene un número par de a's$\}$, entonces:*
$[a] = \{x \in \Sigma^* : x$ tiene un número impar de a's$\}$ y
$[aa] = \{x \in \Sigma^* : x$ tiene un número par de a's$\}$.
Es claro que la relación $\sim_L$ es de índice finito.

Definición 3.30. *Una relación de equivalencia R_2 es un **refinamiento** de una relación de equivalencia R_1 si y solo si las clases de equivalencia de R_1 son uniones de clases de equivalencia de R_2.*

El siguiente teorema debido a Nerode y Myhill [12, 11] es una caracterización algebraica de la noción computacional de tener un lenguaje aceptado por un autómata finito. El teorema es de gran importancia en términos de algoritmos prácticos para minimizar el número de estados en un autómata finito. Una de sus bondades es demostrar que lenguajes concretos son de estado finito.

Teorema 3.31. ***(Teorema de Myhill-Nerode)***

- *Las siguientes proposiciones sobre Σ^* son equivalentes:*

  (i) L es de estado finito;

  (ii) L es la unión de una colección de clases de equivalencia de una congruencia derecha de índice finito sobre Σ^*;

  (ii) $\sim_L$ es de índice finito.

- *Cualquier congruencia derecha que satisface (b) es un refinamiento de $\sim_L$.*

Demostración. $(a) \Rightarrow (b)$. Supongamos que L es un lenguaje regular que es aceptado por un autómata finito determinista $M = (Q, \Sigma, S, F, \delta)$, donde $S = \{q_0\}$. Se define la siguiente relación R_M sobre Σ^* de la siguiente manera:
xR_My si y solo si existe $q \in Q$ tal que

$$\langle q_0, x\rangle \vdash^* \langle q, \lambda\rangle \text{ y } \langle q_0, y\rangle \vdash^* \langle q, \lambda\rangle$$

Observemos que sobre la entrada x o y, terminamos en el estado q. Claramente para cada $z \in \Sigma^*$, se cumplen:

$$\langle q_0, xz\rangle \vdash^* \langle q, z\rangle \text{ y } \langle q_0, yz\rangle \vdash^* \langle q, z\rangle,$$

de donde R_M es una congruencia derecha. Como Q es finito, entonces R_M es de índice finito. Finalmente la implicación se sigue ya que:

$$L = \bigcup_{q\in F} \{\sigma \in \Sigma^* : \langle q_0, \sigma\rangle \vdash^* \langle q, \lambda\rangle\}$$

$(b) \Rightarrow (c)$. Supongamos que R es una congruencia derecha de índice finito tal que L es la unión de algunas de las clases de equivalencia de R. Veamos que $\sim_L$ es de índice finito, para esto basta ver que cada clase de equivalencia de $\sim_L$ es unión de algunas clases de equivalencia de R, ya que R es de índice finito. Para esto, se demuestra que para todo $y \in \Sigma^*$ se cumple

$$[y]_R \subseteq [y]_{\sim_L}, \tag{3.1}$$

de donde se cumpliría que para todo $y' \in [y]_{\sim_L}$,

$$[y']_R \subseteq [y']_{\sim_L} = [y]_{\sim_L},$$

de donde

$$[y]_{\sim_L} = \bigcup_{z\in[y]_{\sim_L}} [z]_R,$$

y como R es de índice finito, entonces $\sim_L$ deberá ser de índice finito. Para demostrar (3.1), supongamos que existen $x, y \in \Sigma^*$

tales que $x \in [y]_R$ y $x \notin [y]_{\sim_L}$, entonces existe $z \in \Sigma^*$ tal que xRy, $xz \in L$ y $yz \notin L$. Ahora como R es congruencia derecha se tiene que $xzRyz$, sea C la clase de equivalencia que contiene a xz y yz, entonces $C \subset L$ o $C \cap L = \emptyset$, en cualquier caso se obtiene una contradicción.

$(c) \Rightarrow (a)$. Veamos que si hay un número finito de clases de equivalencia bajo la relación $\sim_L$, entonces existe un autómata finito que acepta el lenguaje L. Para esto definimos el autómata $M = \langle Q, \Sigma, S, F, \delta \rangle$, donde

1. $Q = \{[x]_{\sim_L} : x \in \Sigma^*\}$,
2. $S = \{[\lambda]_{\sim_L}\}$,
3. $\delta([x], a) = [xa]$ para $x \in \Sigma^*$ y $a \in \Sigma$, y
4. $F = \{x \in \Sigma^* : [x]_{\sim_L} \subseteq L\}$.

Veamos que M está bien definido. Claramente las primeras dos especificaciones están bien definidas. Para demostrar que la tercera especificación está bien definida, supongamos que

$$\delta([x], a) = [xa] \text{ y } \delta([x'], a) = [x'a] \text{ con } x \sim_L x',$$

entonces debemos probar que $xa \sim_L x'a$. Para esto, supongamos por contradicción que $xa \not\sim_L x'a$, entonces existe $w \in \Sigma^*$ tal que $xaw \in L$ y $x'aw \notin L$ o $x'aw \in L$ y $xaw \notin L$, en el primer caso, como $x \sim_L x'$, entonces

$$x(aw) \in L \text{ sii } x'(aw) \in L,$$

como la concatenación es asociativa se llega a una contradicción con $x'aw \notin L$. El segundo caso se sigue por simetría.

Para demostrar que la cuarta especificación es bien definida, se afirma que para cada $x \in \Sigma^*$ se cumple que

$$[x]_{\sim_L} \cap L = \emptyset \vee [x]_{\sim_L} \subseteq L,$$

si esto no se cumpliera tendríamos que existen $x, y, z \in \Sigma^*$ tales que $y \in [x]_{\sim_L} \cap L$ y $z \in [x]_{\sim_L} - L$, de aquí se sigue que

(i) $y \sim_L x$ y $y \in L$,

(ii) $z \sim_L x$ y $z \notin L$.

De (i) obtenemos que $x \in L$ y de (ii) como $z \sim_L x$ tenemos que $z \in L$, lo que contradice (ii).

Ahora veamos que el autómata M reconoce el lenguaje L. Para esto, demostramos primero por inducción sobre la longitud de $y \in \Sigma^*$ que para todo $x \in \Sigma^*$ se cumple

$$\langle [x], y\rangle \vdash^* \langle [xy], \lambda\rangle. \tag{3.2}$$

Para $y = \lambda$ el resultado es claro porque $[x\lambda] = [x]$. Ahora supongamos el resultado para todo $x \in \Sigma^*$ y cadenas de longitud n, entonces si $y' = ay$, con $a \in \Sigma$ y $|y| = n$ se sigue que

$$\begin{array}{lll} \langle [x], y'\rangle = \langle [x], ay\rangle & \vdash \langle \delta([x], a), y\rangle & \text{(definición de } \vdash) \\ & = \langle [xa], y\rangle & \text{(definición de } \delta) \\ & \vdash^* \langle [xay], \lambda\rangle & \text{(hipótesis inductiva)} \\ & = \langle [xy'], \lambda\rangle & (y' = ay) \end{array}$$

Ahora de (3.2), para todo $x \in \Sigma^*$,

$$\text{si } \langle [\lambda], x\rangle \vdash^* \langle q, \lambda\rangle \text{ entonces } q = [x]_{\sim_L}.$$

Entonces de la especificación 4, o $x \in L$ y por tanto $[x]$ es un estado de aceptación o $x \notin L$ y $[x]$ no es un estado de aceptación.

□

Antes de ver un ejemplo del uso del Teorema de Myhill-Nerode, tenemos la definición de congruencia de Myhill.

Definición 3.32. *Decimos que $x \approx_L y$ si para todo $u, v \in \Sigma^*$ se cumple*

$$uxv \in L \text{ sii } uyv \in L.$$

Observe que $\approx_L$ es de índice finito, entonces $\sim_L$ es de índice finito. En efecto, si $\approx_L$ es de índice finito, como $[x]_{\approx_L} \subseteq [x]_{\sim_L}$, entonces $[x]_{\sim_L}$ es unión de clases de equivalencia con la relación $\approx_L$, de donde $\sim_L$ es de índice finito.

Como ejemplos de aplicación del Teorema de Myhill-Nerode demostremos los siguientes resultados clásicos de la teoría de autómatas.

Proposición 3.33. $L = \{a^n b^n : n \in \mathbf{N}\}$ *es no regular.*

Demostración. Para todo $n < m$ tenemos $a^n \not\sim_L a^m$ ya que $a^n b^n \in L$ y $a^m b^n \notin L$. Entonces existen un número infinito de clases $[a^k]$ y por tanto, $\sim_L$ es de índice infinito, del Teorema de Myhill-Nerode se sigue que L no es regular. □

Lema 3.34. ***Lema del bombeo (Bar-Hillel, Perles y Shamir).*** *Si L un lenguaje regular e infinito, entonces existen cadenas $x, y, z \in \Sigma^*$, $y \neq \lambda$ tales que para todo $n \geq 1$, $xy^n z \in L$.*

Demostración. Sean $y_0 = \lambda$, y_1 la menor cadena con el orden lexicográfico de longitud 1 que extiende a y_0 en L, inductivamente definimos y_{n+1} como la menor cadena con el orden lexicográfico de longitud $n+1$ que extiende a y_n en L. Por el Teorema de Mihill-Nerode, L es de índice finito (por ser regular), por lo tanto existe $m > n$ tal que $y_n \sim_L y_m$. Sea $x = y_n$ y $y \in \Sigma^*$ tal que $y_m = xy$. Como $\sim_L$ es una congruencia derecha, entonces $x \sim_L xy$ entonces $xy \sim_L xy^2$ y así por inducción tenemos conjuntos infinitos de palabras equivalentes

$$x \sim_L xy \sim_L xy^2 \sim_L xy^3 \sim_L \cdots$$

Por otro lado, como x tiene una extensión en L, existe z tal que $xz \in L$, de la cadena de equivalencias de arriba se infiere que para todo $n \geq 1$,

$$xy^n z \in L$$

□

3.2. Gramáticas Formales

De manera intuitiva una gramática es un conjunto de reglas que son utilizadas para construir un lenguaje $L \subseteq \Sigma^*$. Estas reglas permiten reemplazar símbolos o cadenas de símbolos con otros símbolos o cadenas de símbolos hasta que finalmente tenemos cadenas de símbolos de Σ que nos permiten formar un elemento del lenguaje. Las restricciones impuestas a las normas nos permiten construir diferentes lenguajes, en particular lenguajes con cualidades deseables.

Definición 3.35. *Una **gramática formal** es una cuádrupla $G = \langle N, \Sigma, S, P \rangle$, donde:*

- *N es un conjunto finito llamado conjunto de **símbolos no terminales**.*
- *Σ es un conjunto finito llamado conjunto de **símbolos terminales**.*
- *S es un elemento distinguido de N, llamado **símbolo inicial**.*
- *P es una relación finita sobre $(N \cup \Sigma)^*$ tal que para todo par $(w, u) \in P$ se cumple que la cadena w contiene un símbolo de N y al menos en un par $(w, u) \in P$ se cumple que $w = S$. Lamamos a P **conjunto de producciones** y $(\alpha, \beta) \in P$ es una **producción** de P.*

Los elementos $(w, u) \in P$, son denotados por $w \to u$. Si $W = uvw$ y $W' = uv'w$ son elementos de $(N \cup \Sigma)^*$ y $v \to v'$, entonces denotamos también $W \to W'$.

Si existen $W_1, \ldots, W_n$ para $n \geq 1$ tales $W_1 \to \cdots \to W_n$ denotamos $W_1 \to^* W_n$ y decimos que W_n es una **derivación** de W_1.

Definición 3.36. *El conjunto L de cadenas de símbolos de Σ que pueden ser generadas por el conjunto de producciones P de una gramática G es llamado **lenguaje generado** por la gramática G y es denotado por $G(L)$.*

Ejemplo 3.37. *Consideremos la gramática* $G = \langle N, S, \Sigma, P\rangle$ *donde:*

- $N = \{sum, A, B\}$,
- $\Sigma = \{+, \times, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$,
- $S = sum$ *y*
- $P = \{sum \to A + B, A \to A + B, B \to A \times B, A \to 0,$
  $A \to 1, \ldots, A \to 9, B \to 0, B \to 1, \ldots, B \to 9\}$.

Entonces tenemos la siguiente derivación

$$\begin{array}{lll} sum \to & A + B & \\ \to & A + A \times B & (B \to A \times B) \\ \to & A + A + B \times B & (A \to A + B) \\ \to & 3 + A + B \times B & (A \to 3) \\ \to & 3 + 4 + B \times B & (A \to 4) \\ \to & 3 + 4 + 5 \times B & (B \to 5) \\ \to & 3 + 4 + 5 \times 6 & (B \to 6). \end{array}$$

Observemos que el lenguaje generado por la gramática G es el conjunto de todas las expresiones aritméticas con números enteros $0 \leq n \leq 9$.

Definición 3.38. *Una gramática donde todas sus producciones son de la forma* $A \to W$*, donde* A *es un símbolo no terminal es llamada* ***gramática libre de contexto****. Los lenguajes generados por una gramática libre de contexto son llamados* ***lenguajes libres de contexto****. Una gramática donde sus producciones tienen la forma* $aAb \to W$*, donde* A *es un símbolo no terminal y* $ab \neq \lambda$ *es llamada* ***gramática sensitiva al contexto****.*

Otro ejemplo más abstracto de una gramática libre de contexto es el siguiente.

Ejemplo 3.39. *Sea* $G = \langle N, \Sigma, S, P\rangle$ *donde* $N = \{S, A\}$, $\Sigma = \{a, b\}$ *y* $P = \{S \to aSb, S \to ab\}$*. Claramente el lenguaje generado por esta gramática es* $L = \{a^n b^n : n \in \mathbb{N}\}$.

Capítulo 4

Autómatas arbóreos

En esta capítulo se generaliza la clase de objetos aceptados por un autómata, siendo estos objetos árboles enraizados etiquetados. Un árbol enraizado etiquetado es un árbol etiquetado con un nodo distinguido. Para tal objetivo se generaliza el concepto de autómata y posteriormente se observa que los resultados de la teoría de autómatas son muy similares.

En lo que sigue, por un *árbol* entenderemos que se trata de un árbol enraizado con etiquetas en un alfabeto Σ y donde cada nodo tiene una etiqueta acotada por una constante f. La colección de tales árboles sobre Σ se denota por Σ^{**}. Con r_T denotamos la raíz de $T \in \Sigma^{**}$. Note que una cadena $a_1 a_2 \cdots a_n$, puede ser considerada como un árbol *unario* con una sola hoja a_1 y con raíz a_n. Sabemos que los autómatas actúan sobre tales árboles unarios, leyendo los símbolos desde la hoja a la raíz, los estados cambian a medida de que se avanza por el árbol. La extensión a árboles más generales es natural, para esto nos movemos de las hojas a la raíz, cambiando de estados según nos guíen las etiquetas de los nodos. Para el caso de árboles unarios sabemos que significa movernos entre sus nodos, para el caso general necesitamos aclarar que significa movernos en nodos k-arios con $k \leq f$.

Supongamos que ν es un nodo de un árbol con k entradas $i_1, i_2, \ldots, i_k$ y suponga que ν es etiquetado por $a \in \Sigma$.

Inductivamente, podemos suponer que correspondiendo a las k líneas de entrada $i_1, i_2, \ldots, i_k$, el autómata debería estar simultaneamente en los estados $q_{j_1}, q_{j_2}, \ldots, q_{j_k}$. Entonces, en el nodo ν el autómata debería moverse al estado especificado por la $k+1$-tupla

$$< q_{j_1}, q_{j_2}, \ldots, q_{j_k}, a >$$

Se puede suponer que los nodos ν del árbol están ordenados lexicográficamente.

Definición 4.1. *Un **autómata arbóreo determinista** sobre Σ es una quíntupla $< Q, \Sigma, \delta, S, F >$ y una cota f donde Q, S, F son como para un autómata clásico de la Definición 3.1 y δ es una función*

$$\delta : \bigcup_{1 \leq i \leq f} Q^i \times \Sigma \to Q.$$

Para simplificar la notación $\delta(x_1, x_2, ..., x_i, a)$ representa $\delta(< x_1, x_2, ..., x_i >, a)$. Analogamente al caso de un autómata clásico tenemos que la acción "si un autómata M está en el estado $(q_{i_1}, q_{i_2}, \ldots, q_{i_k})$ leyendo $a \in \Sigma$, moverse un símbolo a la derecha y cambiar el estado interno de M a $\delta(q_{i_1}, q_{i_2}, \ldots, q_{i_k}, a)$"

Para cada $T \in \Sigma^{**}$, esta acción se representa por

$$< q_{i_1}, q_{i_2}, \ldots, q_{i_k}, aT > \vdash < \delta(q_{i_1}, q_{i_2}, \ldots, q_{i_k}, a), T > .$$

Se tiene también una relación llamada *eval* análoga a la relación $\vdash^*$, la cual se describe en la próxima definición. Dados $T_1, T_2, \ldots, T_k$ árboles enraizados etiquetados, denotamos por

$$T_1 \odot_a T_2 \odot_a \cdots \odot_a T_k$$

el árbol obtenido de considerar un nuevo nodo r y k nuevas aristas que conecten este nodo con las k raíces $r_{T_1}, r_{T_2}, \ldots, r_{T_k}$ de los árboles T_i y etiquetar r con a. Por ejemplo, si tenemos los árboles T_1 y T_2 de la Figura 4.1.

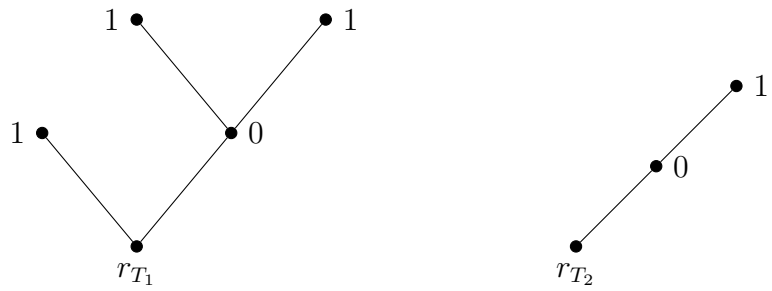


Figura 4.1: Árboles T_1 y T_2 con raices r_{T_1} y r_{T_2}, respectivamente.

Entonces $T_1 \odot_a T_2$ es el árbol de la Figura 4.2.

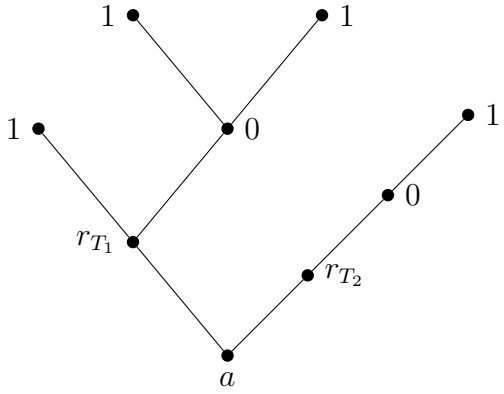


Figura 4.2: Árbol $T = T_1 \odot_a T_2$.

Si adoptamos la convención $\lambda \odot_a = a$ para $a \in \Sigma$, todos los árboles de Σ^{**} pueden ser generados por λ y $\odot_a$, para $a \in \Sigma$. También, si T_1 es como en la Figura 4.1, $T = T_1 \odot_a$ es el árbol de la Figura 4.3.

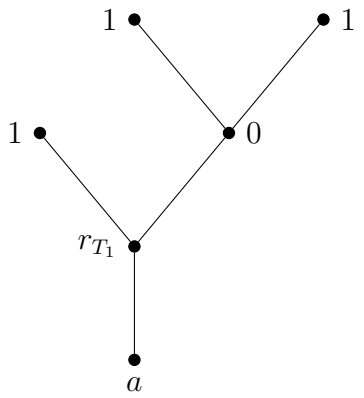


Figura 4.3: Árbol $T = T_1 \odot_a$.

La siguiente definición formaliza lo discutido en el párrafo anterior.

Definición 4.2. ***(Autómata Arbóreo Aceptador)***.
Si $M =< Q, \Sigma, \delta, S, F >$ es un autómata arbóreo determinista, la función $eval_M$ se define por

1. *$eval_M(q_j, a) = \delta(q_j, a)$ para $a \in \Sigma$ y $q_j \in Q$.*

   Denotamos $eval_M(q_0, a)$ por $eval_M(a)$.

2. *$eval_M(T_1 \odot_a T_2 \odot_a \cdots \odot_a T_n) = \delta(eval_M(T_1), \ldots, eval_M(T_n), a)$ para $a \in \Sigma$.*

Decimos que el árbol $T \in \Sigma^{**}$ es ***aceptado*** *si y solo si $eval_M(T) \in F$.*

Cuando sea claro en el contexto quién es el autómata M, denotamos $eval_M$ por $eval$.

Ejemplo 4.3. *Sea $M =< Q, \Sigma, \delta, S, F >$ un autómata arbóreo, donde $Q = \{q_0, q_1\}$, $\Sigma = \{0, 1\}$, $S = \{q_0\}$ y $F = \{q_1\}$. Se define δ en la siguiente tabla:*

Estados	Símbolo	δ(Estados, Símbolo)
q_0	0	q_0
q_0	1	q_1
q_1	0	q_1
q_1	1	q_0
q_0, q_0	0	q_1
q_0, q_0	1	q_1
q_0, q_1	0	q_0
q_0, q_1	1	q_1
q_1, q_0	0	q_0
q_1, q_0	1	q_0
q_1, q_1	0	q_0
q_1, q_1	1	q_1

A continuación veremos si el árbol T de la Figura 4.4 es aceptado por el autómata M.

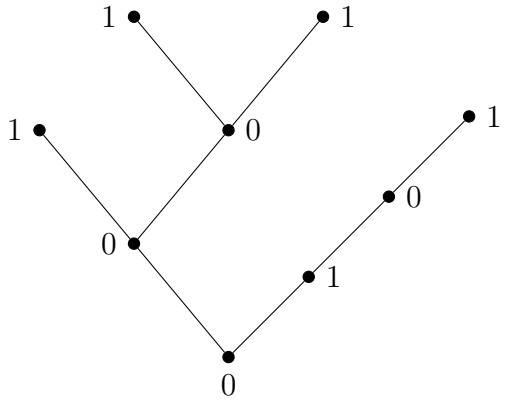
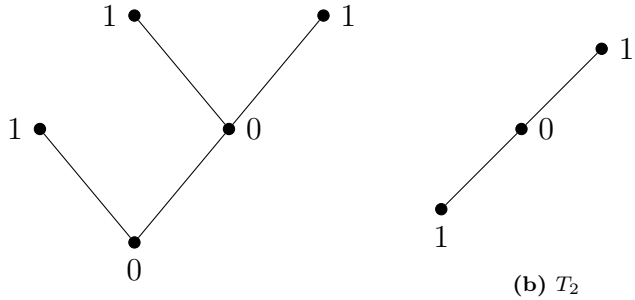


Figura 4.4: Árbol T a verificar si es aceptado por el autómata M.



(a) T_1 **(b)** T_2

Figura 4.5: Subárboles T_1y T_2 de T

En la siguiente figura se muestran dos subárboles de T.

A continuación se muestra la evaluación del árbol T en M, representándolo con la notación $\odot$.

$$T = [T_1 \odot_0 T_2]$$
$$T = [1 \odot_0 (1 \odot_0 1)] \odot_0 [(1\odot_0)\odot_1]$$

$$\begin{aligned}
eval(T) &= eval(T_1 \odot_0 T_2) = \delta(eval(T_1), eval(T_2), 0) \\
&= \delta(eval(1 \odot_0 (1 \odot_0 1)), eval(((1\odot_0)\odot_1), 0)) \\
&= \delta(\delta(eval(1), eval(1 \odot_0 1), 0), \delta(eval(1\odot_0), 1), 0) \\
&= \delta(\delta(\delta(q_0, 1), eval(1 \odot_0 1), 0), \delta(eval(1\odot_0), 1), 0) \\
&= \delta(\delta(\delta(q_0, 1), \delta(eval(1), eval(1), 0), 0), \delta(\delta(eval(1), 0), 1), 0) \\
&= \delta(\delta(\delta(q_0, 1), \delta(\delta(q_0, 1), \delta(q_0, 1), 0), 0), \delta(\delta(\delta(q_0, 1), 0), 1), 0) \\
&= \delta(\delta(q_1, \delta(q_1, q_1, 0), 0), \delta(\delta(q_1, 0), 1), 0) \\
&= \delta(\delta(q_1, q_0, 0), \delta(q_1, 1), 0) \\
&= \delta(q_0, q_1, 0) = q_0 \notin F
\end{aligned}$$

Por lo tanto el árbol T no es aceptado por el autómata M. De manera análoga al caso clásico se puede definir el concepto de autómata arbóreo no determinista introduciendo la multifunción Δ en lugar de la fución δ y también introducir un teorema análogo al teorema 3.6 del capítulo anterior.

4.1. Grámaticas Formales Arbóreas

Sea R cualquier conjunto de símbolos ajenos a Σ, es decir, $R \cap \Sigma = \emptyset$, decimos que T es un árbol *sobre* Σ_R si T es el resultado de tomar un árbol en Σ^{**} y remplazar posiblemente algunas de sus hojas por símbolos de R. Denotamos con Σ_R^{**} la colección de todos los árboles sobre Σ_R.

Definición 4.4. *Una **grámatica arbórea** G sobre Σ es una cuadrupla $\langle \Sigma, S, R, P \rangle$ donde:*

(i) *R es un conjunto de símbolos fuera del alfabeto Σ llamados símbolos **no terminales**,*

(ii) *$S \in R$ es un símbolo distinguido llamado el **símbolo inicial**, y*

(iii) *P es el conjunto de **producciones** de la forma*

$$N \to T$$

*donde $N \in R$ y $T \in \Sigma_R^{**}$. En ocasiones Σ es llamado el conjunto de símbolos **terminales**.*

Antes de dar un ejemplo de una gramática arbórea necesitamos establecer la siguiente notación.

Para $T_1, T_2 \in \Sigma_R^{**}$ denotamos

$$T_1 \to T_2$$

si se cumple lo siguiente

- Existen palabras $X_1, X_2, Y_1, Y_2 \in \Sigma_R^{**}$,
- una hoja $N \in R$ de T_1, y
- una producción $N \to T$ tales que

$$T_1 = X_1 N Y_1 \text{ y } T_2 = X_2 T Y_2$$

Para $T \in \Sigma_R^{**}$ decimos que T' es una **derivación** de T si y solo si existen $T_1, \ldots, T_n$ tales que

$$T \to T_1 \to \cdots \to T_n \to T'$$

en tal caso denotamos $T \to^* T'$.

También decimos que $T \in L(G)$ si

1) $T \in \Sigma^{**}$ (note que esto significa que T no tiene símbolos no terminales), y

2) existe una derivación de T que se obtiene de S usando las producciones de G.

Ejemplo 4.5. *Sea* $\Sigma = \{0, 1\}$ *y consideremos el siguiente conjunto de producciones:*

1. $S \to N_1$
2. $N_1 \to 0, N_2 \to 1$
3. $N_1 \to (N_1 \odot_0 N_2)$
4. $N_2 \to (N_1 \odot_1 N_2 \odot_1 N_1)$

Derivación algebraica:

$$
\begin{array}{ll}
S \to N_1 & T_0 \\
\quad \to (N_1 \odot_0 N_2) & T_1 \\
\quad \to ((N_1 \odot_0 N_2) \odot_0 N_2) & T_2 \\
\quad \to ((0 \odot_0 (N_1 \odot_1 N_2 \odot_1 N_1)) \odot_0 N_2) & T_3 \\
\quad \to ((0 \odot_0 (0 \odot_1 1 \odot_1 (N_1 \odot_0 N_2))) \odot_0 1) & T_4 \\
\quad \to ((0 \odot_0 (0 \odot_1 1 \odot_1 (0 \odot_0 1))) \odot_0 1) & T_5
\end{array}
$$

Las siguientes figuras representan los árboles de derivación de la ejecución de las instrucciones enunciadas anteriormente.

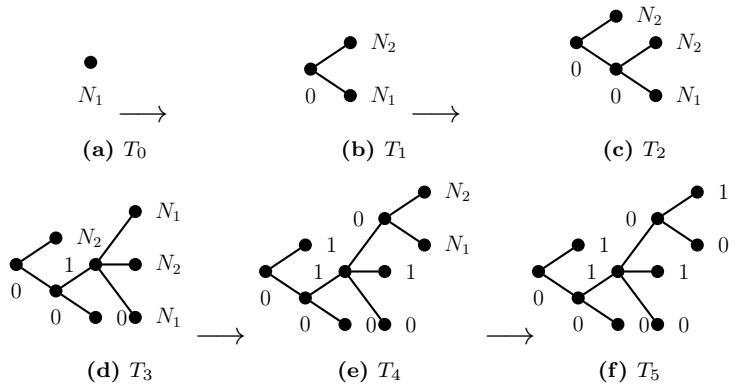


Figura 4.6: Árboles de derivación.

Lema 4.6. ***(Lema de normalización gramatical)****. Si existe una gramática arbórea G que acepta un lenguaje L, entonces existe una gramática G' equivalente a G con la propiedad de que si $N \to T$ es una producción de G', entonces la altura de T es ≤ 1. Además, podemos asegurar que G no tiene producciones de la forma $N_1 \to N_2$ con $N_1, N_2 \in R(G')$.*

Demostración. Para la última parte del lema podemos asegurar que G no tiene producciones de la forma $N_1 \to N_2$ con $N_1, N_2 \in R(G')$, haciendo lo siguiente:

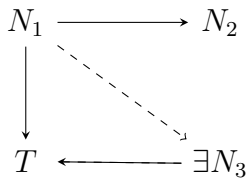
Adicionar en P todas las producciones de la forma

$$N_1 \to T$$

donde $T \in \Sigma_R^{**} - R$ es tal que hay un $N_3 \in R$ que cumple con $N_1 \to^* N_3$ y P contiene una producción de la forma

$$N_3 \to T$$

El siguiente diagrama ilustra la situación de arriba.



Ahora supongamos que existe en G una producción de la forma $N \to T$ con la altura de $T \geq 2$. Entonces la raíz de T es de la forma

$$G_1 \odot_a \cdots \odot_a G_m$$

para algún $a \in \Sigma$ y $G_i \in \Sigma_R^{**}$. Podemos adicionar a R nuevos símbolos no terminales $M_1, M_2, \ldots, M_m$ y nuevas producciones de la forma

$$N \to M_1 \odot_a M_2 \odot_a \ldots \odot_a M_m$$

y

$$M_i \to G_i$$

Claramente la gramática obtenida es equivalente a G y así podemos normalizar a G. □

Definición 4.7. *Un lenguaje arbóreo L es de* ***estado finito*** *si y solo si L es un conjunto de cadenas reconocido por un autómata arbóreo.*

El siguiente teorema atribuido a varios autores es clave para relacionar la eficiencia de algoritmos con la capacidad de aceptar un lenguaje mediante un autómata arbóreo, vea [5].

Teorema 4.8. *Un lenguaje arbóreo L es de estado finito si y solo si existe una gramática arbórea G tal que $L = L(G)$.*

Demostración. Supongamos primero que existe una gramática arbórea normalizada $G = \langle \Sigma, S, R, P \rangle$ tal que $L = L(G)$ (Lema de Normalización Gramatical). Veamos que L es de estado finito, para esto mostramos a continuación un autómata arbóreo aceptador $A =< Q, \Sigma, \delta, S, F >$. Denotamos primero:

- raíz(N) el conjunto de etiquetas de raices de árboles T con $(N \to T) \in P$,
- raíz$(a) = \{a\}$, si $a \in \Sigma$,

- $$entrada(C) = \begin{cases} \{q_0\} & \text{si } C \in \Sigma \\ \{q_C\} & \text{si } C \in R \end{cases}$$
- $Q = \{q_N : N \in R\}$

Por otro lado, dada la producción $N \to T$, si $T = b \in \Sigma$, definimos la transición

$$\delta(q_0, b) = q_N$$

Si T es de altura 1, entonces T es de la forma $b_1 \odot_a b_2 \odot_a \cdots \odot_a b_m$ para algún $a \in \Sigma$ y $b_1, b_2, \ldots, b_m \in \Sigma \cup R$. Para cada $1 \leq i \leq m$, se definen nuevos estados $q'_i = q_{N,b_i,T}$ y transiciones de la forma

$$\delta(entrada(b_i), \text{raíz}(b_i)) = q'_i$$

y

$$\delta(q'_1, \ldots, q'_m, a) = q_N$$

Las hojas de T pueden estar etiquetadas con elementos de R o con símbolos de Σ. Si las etiquetas de las hojas de T están en Σ entonces el único estado entrada posible del autómata es q_0. Ahora, si las hojas de T son etiquetadas con $M \in R$, entonces esta solo puede ser remplazada por otro árbol $T' \in \Sigma_R^{**}$ por una producción de la forma $M \to T'$. Ahora el estado de entrada de T generado por tal producción será el estado final de T', q_M y el símbolo que corresponde a la producción $M \to T'$ será el símbolo raíz de T'.

Ahora para la condición suficiente, supongamos que A es un autómata arbóreo que acepta al lenguaje arbóreo L, entonces el objetivo es construir una grámatica arbórea para L. Para cada estado q_i de A, se tiene un símbolo no terminal Q_i. Dada una transición de la forma

$$\delta(q_{i_1}, \ldots, q_{i_m}, a) = q_k,$$

se construye un producción de la forma

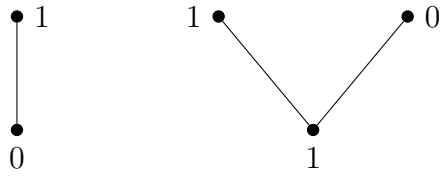
$$Q_k \to Q_{i_1} \odot_a \cdots \odot_a Q_{i_m}.$$

□

También es importante saber que es posible caracterizar mediante expresiones arbóreas regulares los lenguajes arbóreos de estado finito. Para esto es necesario establecer los siguientes conceptos análogos a los presentados en los autómatas clásicos para expresiones regulares.

Definición 4.9. *Dados dos árboles L, W en Σ^{**}, para la etiqueta $a \in \Sigma$, se define el a-**producto** de L con W como el lenguaje $L \cdot_a W$, que se obtiene tomando árboles $T \in W$ y remplazando toda ocurrencia de a como una hoja de T por un árbol T' de L. Si no hay una ocurrencia de a como una hoja de T, entonces esta acción se considera vacía.*

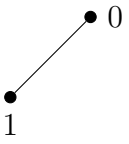
Ejemplo 4.10. *Consideremos los árboles L y W como se muestran en la Figura 4.7.*



(a) L **(b)** W

Figura 4.7: Árboles L y W.

Tomando T como el subárbol en W y



(a) T

Figura 4.8: Subárbol T en W.

$T' = L$ tenemos que el árbol de la Figura 4.9 es una palabra de $L \cdot_0 W$,

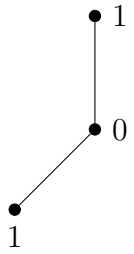


Figura 4.9: Un elemento del lenguaje $L \cdot_0 W$.

Ahora, si elegimos $T = W$ y $T' = L$, otro elemento de $L \cdot_0 W$, es el árbol de la Figura 4.10:

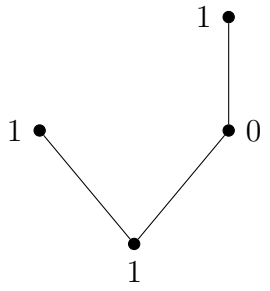
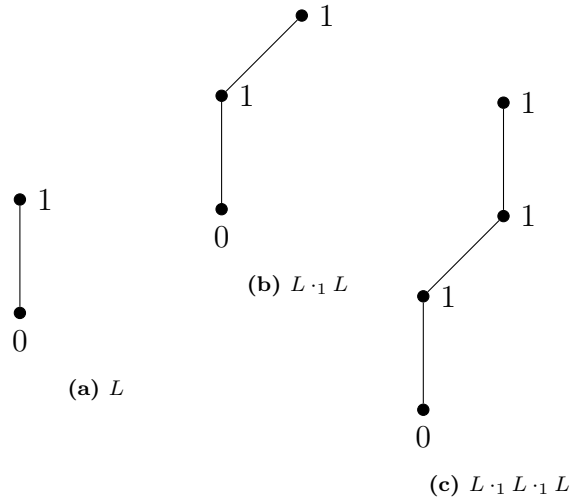


Figura 4.10: Otro elemento de $L \cdot_0 W$.

La a **- estrella** de L es el conjunto

$$L^{*_a} = L \cup (L \cdot_a L) \cup (L \cdot_a L \cdot_a L) \cup \cdots$$

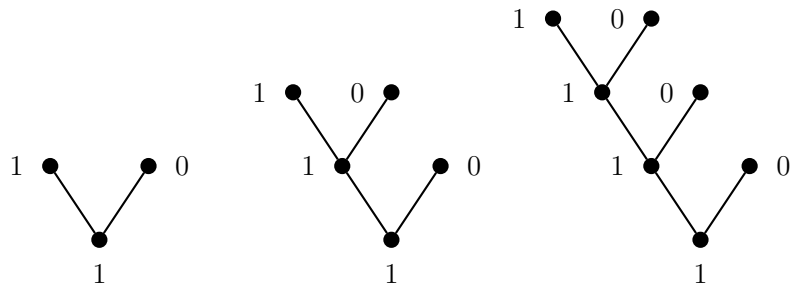
Ejemplo 4.11. *Eligiendo L como en el Ejemplo 4.10, son elementos de la 1-estrella de L los caminos:*



(a) L **(b)** $L \cdot_1 L$ **(c)** $L \cdot_1 L \cdot_1 L$

Figura 4.11: Elementos de L^{*1}.

Ahora si elegimos W como en el Ejemplo 4.10, tenemos los siguientes elementos del lenguaje W^{*_1}:



(a) W **(b)** $W \cdot_1 W$ **(c)** $W \cdot_1 W \cdot_1 W$

Figura 4.12: Elementos de W^{*_1}.

Definición 4.12. *Una **expresión regular arbórea** es definida inductivamente como:*

1.- $\emptyset$ es una expresión regular.

2.- a es una expresión regular para cada $a \in \Sigma$.

3.- $X \cdot_a Y$ *es una expresión regular si* X, Y *son expresiones regulares y* a *es una etiqueta en* Σ.

4.- X^{*_a} *es una expresión regular si* X *y* $a \in \Sigma$ *son expresiones regulares.*

5.- $X_1 \odot_a X_2 \odot_a \cdots \odot_a X_m$ *es una expresión regular, si* $X_1 \ldots X_m$ *y* $a \in \Sigma$ *son expresiones regulares.*

6.- En ningun otro caso.

Dada una expresion regular α *podemos asociar un lenguaje* $L(\alpha)$ *de la siguiente manera:*

1.- $L(\emptyset) = \emptyset$,

2.- $L(a) = \{a\}$,

3.- $L(X \cdot_a Y) = L(X) \cdot_a L(Y)$

4.- $L(X^{*_a}) = L(X)^{*_a}$

5.- $L(X_1 \odot_a \cdots \odot_a X_m) = \{T_1 \odot_a \cdots \odot_a T_m : T_i \in L(X_i)\}$

Naturalmente decimos que L *es regular si y solo si* $L = L(\alpha)$ *para cada expresión regular* α.

El siguiente teorema es análogo al Teorema de Kleene, aquí se establece la equivalencia entre la regularidad y el estado finito, en otros términos la regularidad de un lenguaje implica ser PFT, y recíprocamente.

Teorema 4.13. *Un lenguaje arbóreo es de estado finito si y solo si es regular.*

Demostración. Supongamos que L es de estado finito y que es aceptado por el autómata arbóreo $M =< Q, \Sigma, \delta, S, F >$. Basamos la prueba en el Teorema 3.22. Nuestra inducción involucrará árboles en Σ_Q^{**}; es decir, árboles que son normal excepto que pueden tener hojas etiquetadas por estados de Q. La interpretación de la transición es que δ solo puede actuar

sobre un estado q: $\delta(q, q) = q$.

Sea $V \subseteq Q$. Sea $M(V, j, k)$ que denota el lenguaje que contiene todos los árboles en Σ_V^{**} el cual toma un conjunto de estados como estados de entrada y los transforma via δ al estado j sin usar cualquier estado con indice $\geq k$.

Probamos por inducción sobre k que $M(V, j, k)$ es regular para todo V. Note que esto es suficiente ya que $V = \emptyset$ significa que $\Sigma_Q^{**} = \Sigma^{**}$. Esto es odvio para $k = 1$. Para el paso inductivo, un árbol T está en $M(V, j, k+1)$ si y solo si ya sea $T \in M(V, j, k)$, o:

1. Subárboles de T llevan al estado q_k, entonces
2. quizas para q_k muchas veces, y finalmente
3. a q_j de q_k sin involucrar q_i para $i \geq k$.

Como con el Teorema 3.22, tales consideraciones dan la expresión regular α correspondiente a:

$$M(V,j,k) \cup M(V,k,k+1) \cdot_{q_k} M(V \cup \{q_k\}, k, k+1)^{*k} \cdot_{q_k} M(V,j,k).$$

Es fácil argumentar que $L = L(\alpha)$.
La inversa es sencilla y funciona construyendo un autómata arbóreo finito no determinista correspondiente a una expresión regular α por inducción sobre la complejidad de α. □

4.2. Teorema de Myhill-Nerode para árboles

En esta sección se presenta el Teorema de Myhill-Nerode. Este teorema es el análogo al Teorema de Myhill-Nerode de los autómatas clásicos, del cual puede obtenerse un algoritmo

implícito de minimización de estados. Esto es, un algoritmo basado en la prueba del Teorema de Myhill-Nerode para árboles, que minimiza el número de estados de un autómata que acepta un lenguaje de estado finito L sobre Σ^{**}. Antes de abordar este teorema, será importante considerar los siguientes conceptos.

Definición 4.14. *Sean un nuevo símbolo $x \notin \Sigma$ y árboles etiquetados T_1 y T_2, decimos que $T_1 \sim_L T_2$ si y solo si para todos los árboles $T \in \Sigma^{**}_{\{x\}}$ con exactamente una hoja etiquetada con x, tenemos*

$$T_1 \cdot_x T \in L \text{ si y solo si } T_2 \cdot_x T \in L.$$

La Definición 4.14, expresa que para cualquier árbol T, si concatenamos la raíz de T_1 con la hoja de T etiquetada por x, el resultado está en L si y solo si lo mismo se satisface para T_2.

Definición 4.15. *La relación de equivalencia $\equiv$ es una **congruencia derecha** sobre Σ^{**} si y solo si para árboles $T_1, T_2 \in \Sigma^{**}$,*

$$T_1 \equiv T_2 \text{ si y solo si para todo } T \in \Sigma^{**}_{\{x\}},\ T_1 \cdot_x T \equiv T_2 \cdot_x T.$$

Definición 4.16. *Una relación de equivalencia R sobre Σ^{**} es de **índice finito** si y solo si en Σ^{**} hay un número finito de clases de equivalencia.*

Teorema 4.17. ***(Teorema de Myhill-Nerode)***
Las siguientes proposiciones son equivalentes:

(i) *L es de estado finito sobre Σ^{**}.*

(ii) *L es la unión de clases de equivalencia de alguna congruencia derecha R de índice finito sobre Σ^{**}.*

(iii) *$\sim_L$ tiene índice finito sobre Σ^{**}.*

Demostración.

- $(i) \Rightarrow (ii)$ Sea M un autómata arbóreo determinista que acepta a L, un lenguaje de estado finito. Como con Teorema 3.31, declaramos que dos árboles son $R-$equivalentes si y solo si M los lleva al mismo estado (es decir, de q_0 sobre todas las hojas). Entonces R es una congruencia derecha y tiene índice finito ya que M tiene un número finito de estados.

- $(ii) \Rightarrow (iii)$ Se sigue paso a paso del Teorema 3.31.

- $(iii) \Rightarrow (i)$ Ya que δ es ahora más compleja. Supongamos que L es la unión de clases de equivalencia de $\sim_L$ la cual tiene índice finito. Nuevamente podemos dejar que los estados de la maquina M que construimos sean las clases de congruencia de $\sim_L$. Primero observese que para árboles $T_1 \sim_L T_2$, y $a \in \Sigma$, para $d \leq f$, la cota de Σ^{**} y árboles $Q_1, Q_2, ..., Q_d \in \Sigma^{**}$, tenemos

$$T_1 \odot_a Q_1 \odot_a \cdots \odot_a Q_d \sim_L T_2 \odot_a Q_1 \odot_a \cdots \odot_a Q_d.$$

Esto se sigue del hecho de que

$$T_1 \odot_a B = T_1 \cdot_x x \odot_a B \text{ para todo } B \in \Sigma^{**}.$$

Reemplazando el Q_1 una a la vez, la observación se generaliza para establecer el siguiente hecho.
Para árboles $T_1, ..., T_m$ y $T'_1, ..., T'_m$, con $m \leq f$, si para toda i, $T_i \sim_L T'_i$, entonces para toda $a \in \Sigma$

$$T_1 \odot_a \cdots \odot_a T_m \sim_L T'_1 \odot_a \cdots \odot_a T'_m.$$

Ahora definimos

$$\delta([T_1], ..., [T_m]) = [T_1 \odot_a \cdots \odot_a T'_m].$$

Del hecho anterior, se deduce que si especificamos δ de este modo, entonces δ está bien definida. Los estados aceptados son otra vez $\{[T] \colon T \in L\}$ y otra vez, como

en Teorema 3.31, el estado inicial es $[\lambda]$.

Observe que nuevamente obtendremos un algoritmo de minimización implícito del Teorema de Myhill-Nerode para árboles y un método de conjuntos de prueba.

□

Conclusiones y trabajo a futuro

El Teorema de Myhill-Nerode para autómatas clásicos establece equivalencias con lenguajes de estado finito, esto es, lenguajes reconocidos por autómatas finitos, lo que significa en terminos computacionales que tenemos algoritmos tratables de reconocimiento de cadenas. Los árboles etiquetados pueden ser vistos como la generalización de cadenas de símbolos, por lo que resulta natural extender la teoría clásica de lenguajes formales y autómatas a una teoría de lenguajes y autómatas arbóreos. Los árboles que son grafos acíclicos conectados, tienen la limitación en el número de nodos que pueden contener sus aristas, este hecho limita la posibilidad de tratar (con esta teoría) más problemas que pueden ser resueltos por algoritmos tratables o parámetro fijo tratables, por lo que también resulta natural desarrollar una teoría de lenguajes y autómatas hiperarbóreos. Las aristas en un hipergrafo como se ha visto en el capítulo 1, pueden contener un número arbitrario de nodos. En una teoría de lenguajes y autómatas hiperarbóreos es de esperarse un análogo al Teorema de Myhill-Nerode. También será de gran interés el desarrollo e implantación de los algoritmos implícitos en este teorema.

Bibliografía

[1] Balcázar, José L., Josep Díaz, and Joaquim Gabarró. *Structural Complexity I.* Springer-Verlag Berlin Heidelberg, ISBN-13: 978-3-642-97064-1, first edition 1988.

[2] Bretto, Alain, *Hypergraph theory an introduction.* Mathematical Engineering. Cham: Springer, ISBN 978-3-319-00079-4 (2013).

[3] Courcelle, Bruno, and Joost Engelfriet, *Graph structure and monadic second-order logic.* A language-theoretic approach. Vol. 138. Cambridge University Press, ISBN 978-0-521-8983 (2012).

[4] Diestel, Reinhard, *Graph theory.* Graduate texts in mathematics, Springer, fifth edition, ISBN 978-3-662-53621-6 (2017).

[5] Downey, Rodney G., and Michael R. Fellows, *Fundamentals of parameterized complexity.* Vol. 4. London: springer, 2013.

[6] Flum, Jörg, and Martin Grohe, *Parameterized Complexity Theory.* Texts Theoretical Computer Science. An EATCS Series. Springer-Verlag, Berlin Heidelberg, ISBN 103-540-29952-1 (2006).

[7] Gries, David, and Fred B. Schneider. *Monographs in Computer Sciences* Springers (2007).

[8] Kleene, Stephen C. *Representation of events in nerve nets and finite automata.* Automata studies 34 (1956).

[9] McNaughton, Robert, and Hisao Yamada. *Regular expressions and state graphs for automata.* IRE transactions on Electronic Computers 1 (1960).

[10] Mendelson, Elliott, *Introduction to mathematical logic, Wadsworth Publ.* Co., Belmont, CA (1987).

[11] Myhill, John. *Finite automata and the representation of events.* WADD Technical Report 57 (1957): 112-137.

[12] Nerode, Anil. *Linear automaton transformations.* Proceedings of the American Mathematical Society 9.4 (1958): 541-544.

[13] Rabin, Michael O. and Dana Scott, *Finite automata and their decision problems.* In IBM journal of research and development, vol.3, no.2, pp. 114-125, April 1959, doi: 10.1147/rd.32.0114.

[14] Van Bevern, R., Downey, R. G., Fellows, M. R., Gaspers, S., & Rosamond, F. A. (2015). *Myhill–Nerode methods for hypergraphs.* Algorithmica, 73(4), 696-729.