

UNIVERSIDAD POLITÉCNICA DE VALENCIA
DEPARTAMENTO DE SISTEMAS INFORMÁTICOS Y COMPUTACIÓN

Autómatas finitos: Irreducibilidad e Inferencia

Tesis Doctoral presentada por Manuel Vázquez de Parga Andrade
Dirigida por Pedro García Gómez
Valencia, Abril 2008

A Mendri

Índice

1. Introducción.	6
2. Lenguajes y Autómatas finitos.	8
2.1. Alfabetos, palabras y lenguajes.	8
2.2. Autómatas finitos.	9
2.3. Autómatas finitos deterministas.	11
2.4. Otros tipos de autómatas finitos.	13
2.4.1. Autómatas finitos con transiciones vacías.	13
2.4.2. RFSA (Residual Finite State Automaton).	13
2.4.3. Autómatas maximales asociados a lenguajes finitos.	14
2.5. Operaciones sobre autómatas finitos.	16
2.5.1. Cocientes.	16
2.5.2. Reverso.	17
2.5.3. Subautómatas.	17
2.5.4. Homomorfismos.	18
2.5.5. Unión.	18
2.5.6. Determinización.	19
2.5.7. Minimización.	19
2.6. Autómata Universal.	19
3. Inferencia gramatical de lenguajes regulares.	22
3.1. Inferencia gramatical.	22
3.2. Inferencia gramatical de lenguajes regulares.	24
3.2.1. Conceptos básicos.	24
3.2.2. El algoritmo Trakhtenbrot-Barzdin-Gold.	25
3.2.3. El algoritmo RPNI.	26
3.2.4. Inferencia no determinista.	28
4. Minimalización de Autómatas Finitos.	32
4.1. Introducción.	32
4.2. Minimalidad.	32
4.2.1. Autómatas finitos minimales.	32
4.2.2. Minimalidad relativa.	34
4.3. Irreducibilidad.	35
4.3.1. Autómatas finitos irreducibles.	35
4.3.2. Estados fusionables.	36
4.3.3. Estados fusionables y el autómata universal.	38
4.3.4. Irreducibilidad relativa.	41
4.4. Concisión.	44
4.4.1. Autómatas finitos concisos.	44
4.4.2. Estados superfluos.	45
4.4.3. Autómatas finitos concisos irreducibles.	46
4.4.4. Concisión relativa.	47

4.4.5. Irreducibilidad y concisión relativas.	48
5. Redes de autómatas cocientes.	52
5.1. Red de cocientes de un autómata finito.	52
5.2. Completitud y universalidad estructural.	53
5.2.1. Aplicaciones a la inferencia de L_3	58
5.3. Semired de cocientes asociada a un par de autómatas finitos.	59
5.3.1. Aplicaciones a la inferencia de L_3	63
6. Inferencia mediante subautómatas asociados a palabras.	66
6.1. Subautómatas asociados a una palabra en un lenguaje.	66
6.2. Una familia de algoritmos para la inferencia de \mathcal{L}_3 mediante autómatas finitos.	68
6.2.1. La familia WASRI.	68
6.2.2. WASRI1.	69
6.3. Dos ejemplos.	70
6.3.1. Ejemplo de ejecución.	70
6.3.2. Un ejemplo bonito.	71
7. Experimentación.	73
7.1. Las muestras.	73
7.2. <i>WASRI2</i>	74
7.3. <i>LPEF</i>	74
7.4. Resultados	74
8. Conclusión.	76

1. Introducción.

Los lenguajes regulares se encuentran entre los objetos más básicos de la teoría de lenguajes y entre los más estudiados. Una de las razones para que esto sea así es la gran variedad y riqueza formal de sus representaciones: gramáticas de tipo tres, expresiones regulares, autómatas finitos deterministas y no deterministas, etc. Estas representaciones juegan además un importante papel en múltiples aplicaciones informáticas en los ámbitos del reconocimiento de formas, aprendizaje artificial, verificación de programas, especificación sintáctica, sistemas de manipulación de textos, lenguajes de programación (awk, perl, php), editores, etc. Evidentemente, el tamaño de la representación interviene crucialmente en el tiempo de ejecución de los algoritmos en que se aplica.

El sistema más extendido de representación de lenguajes regulares es el dispositivo abstracto denominado autómata finito determinista. Estos autómatas, además de poseer gran riqueza formal, ostentan una serie de propiedades que facilitan en diferentes maneras su manejo. Una de estas propiedades, quizás la más importante, es que para cada lenguaje regular existe un autómata finito determinista canónico, mínimo en cuanto a tamaño y el único con esta propiedad y que además puede ser calculado en tiempo polinómico a partir de cualquier otro autómata finito determinista equivalente.

Sin embargo, cuando el tamaño de la representación es muy relevante puede ocurrir que los autómatas finitos deterministas no sean la representación más adecuada, pues el tamaño del autómata finito determinista mínimo puede ser exponencialmente grande respecto al de otras representaciones equivalentes, en particular al de algún otro automata finito (no determinista) para el mismo lenguaje.

Los autómatas finitos (no deterministas) presentan sin embargo el inconveniente de que muchos de los problemas relacionados con ellos no admiten una solución eficiente. En particular, para cada lenguaje regular existen en general varios autómatas finitos de tamaño minimal y el problema de encontrar alguno de estos equivalente a otro autómata finito dado es *PSPACE_completo*. Así, siendo inabordable el problema de la minimalización, diferentes autores han acometido un problema similar aunque más modesto en sus objetivos, el problema de la reducción, consistente en encontrar un autómata más pequeño que el de entrada y equivalente a él.

Otro problema relevante de características similares al anterior es el de la inferencia de lenguajes regulares, que persigue encontrar un autómata finito lo más pequeño posible consistente con una serie de datos de entrada. Tampoco este problema admite solución general polinómica, ni aún en el caso de los autómatas finitos deterministas, aunque existen diferentes algoritmos polinómicos que lo resuelven cuando se dan determinadas circunstancias y que en muchos casos son capaces de encontrar soluciones razonables.

En este trabajo abordamos estos dos problemas, el de la reducción de autómatas finitos y el de la inferencia de los lenguajes regulares, y estudiamos las relaciones entre ellos.

Tras esta introducción, en el segundo capítulo se repasan los conceptos y

proposiciones básicas de teoría de lenguajes y se establece la notación utilizada a lo largo del trabajo. Lo mismo, pero en lo que concierne a la inferencia gramatical, se realiza en el tercer capítulo, donde se describe además el actual estado del arte. En el cuarto capítulo se discuten los problemas de minimalización y reducción de autómatas finitos y se demuestra el primero de los resultados centrales de esta tesis, la existencia de una cota en el tamaño de los autómatas finitos irreducibles que aceptan un determinado lenguaje regular. Se introducen además los conceptos de irreducibilidad y concisión relativas. En el capítulo quinto se discuten y amplían los conceptos de red de autómatas cocientes y completitud estructural y se introduce el concepto de universalidad estructural. Como resultado de este estudio se obtiene el segundo de los resultados centrales de este trabajo, el cual establece que cualquier algoritmo de fusión de estados que obtenga como resultado un autómata finito irreducible compatible con los datos de entrada infiere en el límite la clase de los lenguajes regulares siempre y cuando se aplique en cierta manera, por lo demás poco restrictiva. En el capítulo sexto se introduce el concepto de subautómata asociado a una palabra en un lenguaje y a partir de él se describe una nueva familia de algoritmos de inferencia de la clase de los lenguajes regulares representados mediante autómatas finitos, algoritmos que tienen la particularidad de que su complejidad temporal es función básicamente de la longitud de las palabras de la muestra de entrada, mientras que son prácticamente lineales respecto al número de las mismas. En el capítulo séptimo se describen algunos algoritmos de las familias mencionadas y mediante la experimentación se compara con éxito su rendimiento con el de los algoritmos de inferencia hoy por hoy más relevantes. Por último, en el capítulo octavo se enumeran las conclusiones de este trabajo y se exponen diferentes vías para la futura ampliación del mismo.

2. Lenguajes y Autómatas finitos.

2.1. Alfabetos, palabras y lenguajes.

En este apartado repasaremos brevemente los conceptos más básicos de la teoría de lenguajes con el objetivo principal de establecer la notación básica que utilizaremos a lo largo de este trabajo. Los lectores interesados podrán encontrar estos conceptos desarrollados con mas amplitud en cualquiera de los manuales clásicos sobre introducción a la teoría de lenguajes como [17], [19] o [35].

Un alfabeto Σ es cualquier conjunto finito y no vacío de símbolos.

Una palabra (sobre Σ) x es cualquier secuencia finita de símbolos de Σ . Denominaremos $|x|$ a la longitud de la palabra x , esto es, al número de símbolos que la componen. Así mismo denotaremos por $|x|_a$ al número de veces que el símbolo a aparece en la palabra x . Representaremos mediante λ a la palabra vacía, aquella formada por cero símbolos.

Llamaremos Σ^* al conjunto formado por todas las palabras sobre Σ . Σ^* constituye un monoide libre junto con la operación de concatenación de palabras, asociativa, con elemento neutro λ . Supuesto un orden alfabético sobre los símbolos de Σ , consideraremos un orden canónico sobre las palabras de Σ^* en el que las palabras se ordenan de menor a mayor longitud y alfabéticamente dentro de una misma longitud. Representaremos por Σ^n al conjunto formado por todas las palabras de longitud n .

Utilizaremos la notación exponencial para representar concatenaciones de una palabra consigo misma, de manera que para cualquier palabra x , $x^0 = \lambda$, y para $n > 0$, $x^n = xx^{n-1}$.

El reverso de x , x^R , es la imagen especular de x . Formalmente $\lambda^R = \lambda$, $(xa)^R = ax^R$ para cualesquiera $a \in \Sigma, x \in \Sigma^*$.

Denotaremos por $Pref(x)$, $Suf(x)$, $Seg(x)$, $SubW(x)$ respectivamente a los conjuntos formados por los prefijos, sufijos, segmentos y subpalabras de x . Tanto λ como x pertenecen a cualquiera de estos conjuntos.

Un lenguaje (sobre Σ) L es cualquier subconjunto de Σ^* , esto es, cualquier conjunto de palabras sobre Σ . Representaremos por \emptyset al lenguaje vacío. Un lenguaje es finito si contiene un número finito de palabras e infinito en caso contrario. Representaremos mediante $|L|$ el número de palabras del lenguaje L y mediante $\|L\|$ a la suma de las longitudes de las palabras de L .

Algunas de las operaciones mas usuales sobre lenguajes son:

Las operaciones booleanas de unión, \cup , intersección, \cap , complementación, y diferencia, $-$, definidas igual que para los conjuntos en general.

La concatenación de dos lenguajes L_1 y L_2 es $L_1L_2 = \{xy : x \in L_1, y \in L_2\}$. Esta operación es asociativa y tiene como elemento neutro el lenguaje $\{\lambda\}$. Utilizaremos también para ella la notación exponencial, de forma que $L^0 = \{\lambda\}$ y para $n > 0$, $L^n = LL^{n-1}$.

La clausura de un lenguaje es $L^* = \bigcup_{n \geq 0} L^n$, esto es, el submonoide de Σ^* generado por L . La clausura positiva es $L^+ = \bigcup_{n \geq 1} L^n$.

El reverso de L es $L^R = \{x^R : x \in L\}$.

$Pref(L) = \bigcup_{x \in L} Pref(x)$. En forma similar se definen $Suf(L)$, $Seg(L)$ y $SubW(L)$.

Dados un lenguaje L y una palabra x denominaremos derivada de L respecto a x al conjunto $x^{-1}L = \{y \in \Sigma^* : xy \in L\}$. Además denotaremos por Lx^{-1} al lenguaje $\{y \in \Sigma^* : yx \in L\}$.

Siendo Σ_1 y Σ_2 alfabetos un homomorfismo de Σ_1 sobre Σ_2 es cualquier función $h : \Sigma_1 \rightarrow \Sigma_2^*$. Esta operación se extiende a palabras de forma que para cualquier homomorfismo h , $h(\lambda) = \lambda$ y $h(ax) = h(a)h(x)$ para cada símbolo a y cada palabra x . Se extiende asimismo a lenguajes siendo $h(L) = \{h(x) : x \in L\}$.

Si h es un homomorfismo de Σ_1 sobre Σ_2 y L es un lenguaje sobre Σ_2 entonces $h^{-1}(L) = \{x \in \Sigma_1^* : h(x) \in L\}$. El operador h^{-1} se denomina homomorfismo inverso.

De los diferentes sistemas de representación de lenguajes utilizaremos con cierta frecuencia las expresiones regulares, definidas en la siguiente manera: \emptyset , λ y a , para cada $a \in \Sigma$, son expresiones regulares que denotan respectivamente a los lenguajes \emptyset , $\{\lambda\}$ y $\{a\}$. Si r_1 y r_2 son expresiones regulares que representan respectivamente a los lenguajes L_1 y L_2 entonces $r_1 + r_2$, r_1r_2 y r_1^* son expresiones regulares que representan respectivamente a los lenguajes $L_1 \cup L_2$, L_1L_2 y L_1^* . La prioridad de estos operadores es, de menor a mayor, $+$, \cdot , $*$, donde \cdot denota la concatenación. Esta prioridad puede alterarse en la forma habitual mediante el uso de paréntesis. Sólo las expresiones así construidas son expresiones regulares, si bien, por concisión, en algunos casos las utilizaremos combinadas con otros operadores como la notación exponencial y la clausura positiva.

Dado un alfabeto Σ llamaremos clase de lenguajes a cualquier conjunto de lenguajes sobre Σ . La clase de los lenguajes que pueden ser representados mediante expresiones regulares se llama clase de los lenguajes regulares o \mathcal{L}_3 y es la misma que la formada por los lenguajes que pueden ser representados mediante autómatas finitos. \mathcal{L}_3 es cerrada bajo cada una de las operaciones sobre lenguajes anteriormente descritas.

2.2. Autómatas finitos.

Rabin y Scott [31] introdujeron el concepto de autómata finito no determinista. Nosotros utilizaremos una ligera generalización del mismo a la que denominaremos autómata finito.

Intuitivamente, un autómata finito es un dispositivo abstracto que, partiendo de una configuración inicial, procesa secuencias de símbolos mediante una unidad de control que admite un número finito de estados diferentes. Cada vez que se procesa un símbolo la unidad de control cambia de estado.

Los autómatas finitos configuran una de las clases de sistemas mas fundamentales en la teoría de lenguajes, tanto por la riqueza de su estructura como por la diversidad e importancia de sus aplicaciones prácticas.

Las definiciones y proposiciones de este apartado pueden encontrarse en cualquier manual de teoría de lenguajes, tales como el de Hopcroft y Ullman

[19], el de Harrison [17], etc.

Definición 1 *Un autómata finito es cualquier quintupla $A = \langle Q, \Sigma, \delta, I, F \rangle$, donde Q es un conjunto finito de estados, Σ es un alfabeto, I es un subconjunto no vacío de Q a cuyos elementos llamamos estados iniciales, F es un subconjunto de Q a cuyos elementos llamamos estados finales y, por último, δ , a la que se llama función de transición, es una función $\delta : (Q \times \Sigma) \rightarrow 2^Q$ o, equivalentemente, un subconjunto de $Q \times \Sigma \times Q$.*

Cada elemento $\langle q_1, a, q_2 \rangle$ de $Q \times \Sigma \times Q$ tal que $q_2 \in \delta(q_1, a)$ es una transición de δ .

La función δ se extiende en forma natural a conjuntos de estados, de manera que para cualquier subconjunto P de Q y cualquier símbolo a de Σ , $\delta(P, a) = \bigcup_{q \in P} \delta(q, a)$.

Asimismo δ se extiende a palabras, siendo para todo $P \subseteq Q$, $\delta(P, \lambda) = P$ y para todo $a \in \Sigma, x \in \Sigma^*$, $\delta(P, ax) = \delta(\delta(P, a), x)$.

Una palabra x sobre Σ es aceptada o reconocida por el autómata finito $A = \langle Q, \Sigma, \delta, I, F \rangle$ si $\delta(I, x) \cap F \neq \emptyset$.

Llamaremos camino en A para la palabra $x = a_1 a_2 \dots a_{n-1}$ a cualquier secuencia finita $\langle q_1, a_1, q_2 \rangle \langle q_2, a_2, q_3 \rangle \langle q_3, a_3, q_4 \rangle \dots \langle q_{n-1}, a_{n-1}, q_n \rangle$ tal que $q_{i+1} \in \delta(q_i, a_i)$ para cada $i, 1 \leq i < n$. Si además $q_1 \in I$ y $q_n \in F$ diremos que ese camino es una aceptación de x .

Definición 2 *El lenguaje aceptado o reconocido por A , $L(A)$, es el formado por todas aquellas palabras que son aceptadas por A . Un lenguaje es regular si es aceptado por algún autómata finito. La clase de los lenguajes regulares se denota \mathcal{L}_3 .*

Dos autómatas finitos son equivalentes si ambos aceptan el mismo lenguaje.

Definición 3 *En un autómata finito $A = \langle Q, \Sigma, \delta, I, F \rangle$ el lenguaje por la izquierda de un estado q es $L_q = \{x \in \Sigma^* : q \in \delta(I, x)\}$, su lenguaje por la derecha es $R_q = \{x \in \Sigma^* : \delta(q, x) \cap F \neq \emptyset\}$ y su lenguaje interior es $I_q = \{x \in \Sigma^* : q \in \delta(q, x)\}$. Denotaremos I_{pq} al lenguaje $\{x \in \Sigma^* : q \in \delta(p, x)\}$.*

Proposición 4 *Para cualquier autómata finito $A = \langle Q, \Sigma, \delta, I, F \rangle$, $L(A) = \bigcup_{q \in I} I_q R_q = \bigcup_{q \in I} R_q = \bigcup_{q \in Q} L_q I_q R_q = \bigcup_{q \in Q} L_q R_q = \bigcup_{q \in F} L_q I_q = \bigcup_{q \in F} L_q$.*

Demostración. Trivial. ■

Un estado q de un autómata finito $A = \langle Q, \Sigma, \delta, I, F \rangle$ es inalcanzable si no existe $x \in \Sigma^*$ tal que $q \in \delta(I, x)$ o, equivalentemente si $L_q = \emptyset$. Es un sumidero si $\forall x \delta(q, x) \cap F = \emptyset$ o, lo que es lo mismo, si $R_q = \emptyset$. Un estado es inútil si es inalcanzable o es un sumidero y, por tanto, si $L_q R_q = \emptyset$. Un autómata es aseado si no contiene estados inútiles. El autómata $A' = \langle Q', \Sigma, \delta', I', F' \rangle$ donde Q' se obtiene de eliminar los estados inútiles de Q , $I' = I \cap Q'$, $F' = F \cap Q'$ y $\delta' = \delta \cap (Q' \times \Sigma \times Q')$ es aseado y equivalente a A . Generalmente, si no se indica

lo contrario, supondremos que los autómatas finitos con los que trabajamos son aseados.

Un estado q es superfluo si el autómata resultante de eliminarlo junto con todas las transiciones en las que interviene es equivalente al autómata original. Todos los estados inútiles son superfluos pero lo contrario no es cierto en general. Diremos que un autómata es conciso si no contiene estados superfluos.

Un autómata finito es minimal si no existe otro equivalente a él con menos estados. En general cada lenguaje regular es aceptado por más de un autómata finito minimal.

Usualmente describiremos un autómata finito $A = \langle Q, \Sigma, \delta, I, F \rangle$ mediante su diagrama de transiciones, que consiste en un grafo dirigido con tantos nodos como estados tiene Q , un nodo asociado a cada estado. Los estados iniciales se distinguen mediante una flecha, los finales se distinguen en negrita y por último, el grafo contendrá un arco del nodo asociado a q_1 al asociado a q_2 etiquetado a si y sólo si $\langle q_1, a, q_2 \rangle \in \delta$. Véase un ejemplo en la figura 1.

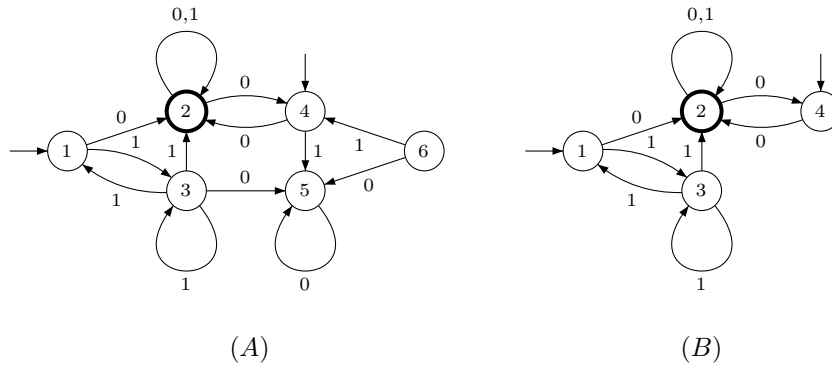


Figura 1: (A) Diagrama de transiciones del autómata finito $A = \langle Q, \Sigma, \delta, I, F \rangle$, donde $Q = \{1, 2, 3, 4, 5, 6\}$, $\Sigma = \{0, 1\}$, $\delta = \{(1, 0, 2), (1, 1, 3), (2, 0, 2), (2, 1, 2), (2, 0, 4), (3, 1, 1), (3, 1, 2), (3, 1, 3), (3, 0, 5), (4, 0, 2), (4, 1, 5), (5, 0, 5), (6, 1, 4), (6, 0, 5)\}$, $I = \{1, 4\}$ y $F = \{2\}$. Acepta el lenguaje $(0 + 11)(0 + 1)^*$. (B) Dado que el estado 5 es un sumidero y el 6 es inalcanzable, este autómata es equivalente al anterior.

Desde el punto de vista de los diagramas de transición una palabra x es aceptada por el autómata si existe un camino etiquetado con los símbolos de x que conduce desde algún estado inicial hasta alguno final.

2.3. Autómatas finitos deterministas.

Definición 5 Un autómata finito $A = \langle Q, \Sigma, \delta, I, F \rangle$ es determinista si $|I| = 1$ y para todo estado q y todo símbolo a , $|\delta(q, a)| \leq 1$.

Usualmente, si $I = \{q_0\}$, A se representa en la forma $\langle Q, \Sigma, \delta, q_0, F \rangle$, donde δ es una función parcial $\delta : (Q \times \Sigma) \rightarrow Q$.

Si δ es una función total diremos que A es un autómata finito determinista completamente especificado o completo. A partir de cada autómata finito determinista no completo $A = \langle Q, \Sigma, \delta, q_0, F \rangle$ puede obtenerse uno completo equivalente $A' = \langle Q \cup \{q_s\}, \Sigma, \delta', q_0, F \rangle$ sin más que añadir un nuevo estado q_s de forma que para todo $q \in Q \cup \{q_s\}$ y para todo $a \in \Sigma$ tales que $\delta(q, a)$ no está definida, $\delta'(q, a) = q_s$, siendo en los demás casos $\delta'(q, a) = \delta(q, a)$. En general consideraremos que A y A' son el mismo autómata.

Proposición 6 *Todo lenguaje regular es aceptado por algún autómata finito determinista.*

Demostración. Sea $A = \langle Q, \Sigma, \delta, I, F \rangle$ un autómata finito. El autómata finito determinista $A' = \langle 2^Q, \Sigma, \delta, I, 2^Q - 2^{Q-F} \rangle$, donde ahora δ se interpreta como una función de $2^Q \times \Sigma \rightarrow 2^Q$, es equivalente a A . ■

Los autómatas de la figura 1 no son deterministas. En la figura 2(A) se muestra otro equivalente a ellos que sí lo es.

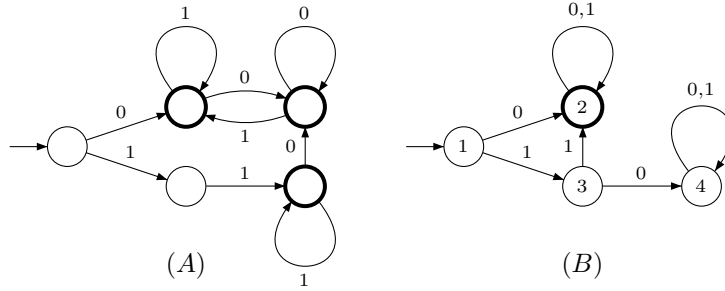


Figura 2: (A) Autómata finito determinista para el lenguaje $(0 + 11)(0 + 1)^*$. No es completo. (B) Autómata finito determinista mínimo completo equivalente al anterior. Los estados se corresponden con los siguientes valores de derivadas: $1 := \lambda^{-1}L = L$, $2 := 0^{-1}L = (0 + 1)^*$, $3 := 1^{-1}L = 1(0 + 1)^*$, $4 := 10^{-1}L = \emptyset$. Este último estado es inútil, por lo que puede ser suprimido.

Proposición 7 *Para cada lenguaje $L \subseteq \Sigma^*$ sea $D = \{x^{-1}L : x \in \Sigma^*\}$. $L \in \mathcal{L}_3 \Leftrightarrow |D| < \infty$.*

Demostración. Por una parte, el autómata $D = \langle D, \Sigma, \delta, L, D - 2^{\Sigma^+} \rangle$ donde $\delta(x^{-1}L, a) = (xa)^{-1}L$ acepta el lenguaje L , y si $|D| < \infty$, D es un autómata finito determinista.

Por otra parte, sea $A = \langle Q, \Sigma, \delta, q_0, F \rangle$ un autómata finito determinista completo tal que $L(A) = L$. Puesto que $\delta(q_0, x) = \delta(q_0, y) = q \Rightarrow x^{-1}L = y^{-1}L = R_q$ tenemos que $|Q| \geq |D|$ y por tanto $|D| < \infty$. ■

Corolario 8 *Para cada lenguaje regular el autómata finito determinista D es mínimo respecto al número de estados, el único excepto isomorfismos.*

En la figura 2(B) se muestra el mínimo equivalente al de la figura 2(A).

Sin embargo D , considerado como autómata finito, no será en general un minimal.

La proposición anterior fue establecida por Myhill y Nerode y \equiv_L , la congruencia por la derecha sobre Σ^* tal que $x \equiv_L y \Leftrightarrow x^{-1}L = y^{-1}L$, es conocida como la congruencia de Nerode. Evidentemente el índice de esta congruencia es igual al número de estados del autómata finito determinista mínimo para L .

2.4. Otros tipos de autómatas finitos.

2.4.1. Autómatas finitos con transiciones vacías.

El concepto de autómata finito con transiciones vacías es una generalización del de autómata finito. Dado que para cada autómata de este tipo existe un autómata finito equivalente con el mismo número de estados, los autómatas finitos con transiciones vacías apenas incrementan la capacidad representativa de los autómatas finitos. Su principal utilidad reside en contribuir a simplificar algunas demostraciones, pero incluso esta utilidad ha disminuido debido a la utilización de autómatas finitos con varios estados iniciales.

A lo largo de este trabajo apenas serán utilizados, por lo que nos limitaremos a dar una breve descripción de los mismos.

Definición 9 *Se denomina autómata finito con transiciones vacías a cualquier quintupla $A = \langle Q, \Sigma, \delta, I, F \rangle$, donde Q , Σ , I y F significan lo mismo que en el caso de los autómatas finitos, mientras que δ es en este caso una función $\delta : (Q \times (\Sigma \cup \{\lambda\})) \rightarrow 2^Q$.*

Los autómatas finitos con transiciones vacías se representan mediante diagramas de transiciones contruidos de la misma manera que los asociados a autómatas finitos. A los arcos etiquetados λ se les denomina transiciones vacías.

Desde el punto de vista de estos diagramas una palabra x es aceptada por un autómata A si existe algún camino en A etiquetado con los símbolos de x y con transiciones vacías intercaladas en cualesquiera posición y número que conduzca desde algún estado inicial hasta algún estado final.

Véase un ejemplo en la figura 3(A).

2.4.2. RFSA (Residual Finite State Automaton).

La derivada de un lenguaje L respecto a una palabra x , $x^{-1}L$, recibe también el nombre de lenguaje residual de L asociado a x . Un lenguaje residual $x^{-1}L$ es compuesto si $x^{-1}L = \cup \{y^{-1}L : y^{-1}L \subsetneq x^{-1}L\}$. En caso contrario es primo.

Denis, Lemay y Terlutte introdujeron en [9],[10] y [11] el concepto de *RFSA* como autómata finito cuyos estados están asociados a (o dicho de otra manera, tienen lenguajes por la derecha iguales a) lenguajes residuales del lenguaje que

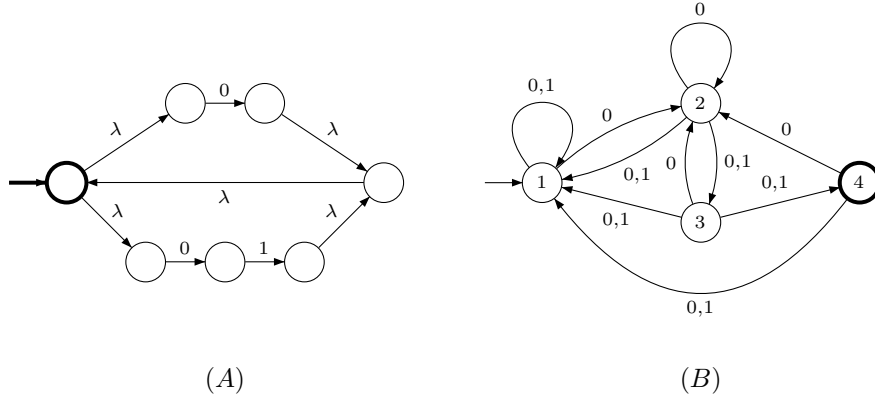


Figura 3: (A) Autómata finito con transiciones vacías para el lenguaje $(0 + 01)^*$. (B) *RFSM* canónico para el lenguaje $(0 + 1)^*0(0 + 1)^2$. Los estados se corresponden con los siguientes valores de residuos primos: $1 := \lambda^{-1}L = L$, $2 := 0^{-1}L = L + (0 + 1)^2$, $3 := 01^{-1}L = L + 0 + 1$, $4 := 011^{-1}L = L + \lambda$. Los demás residuos hasta un total de 8, tamaño del autómata finito determinista mínimo, son compuestos.

reconocen. Desarrollaron además el algoritmo para la inferencia de lenguajes regulares *DeLeTe2*, que produce como salida un *RFSM*. Este algoritmo, uno de los mas relevantes propuestos últimamente, será uno de los que comparemos con aquellos que proponemos en este trabajo.

La utilización de *RFSMs* para representar lenguajes resulta interesante por el hecho de la existencia para cada lenguaje regular de un *RFSM* canónico, minimal entre los *RFSMs*, que consta siempre de un número de estados menor o igual al del autómata finito determinista mínimo para ese lenguaje. Además, para algunas subfamilias parametrizadas de lenguajes el tamaño del autómata finito determinista mínimo crece respecto al valor del parámetro exponencialmente más deprisa que el del *RFSM* canónico.

Definición 10 Dado un lenguaje regular $L \subseteq \Sigma^*$, el *RFSM* canónico de L es el autómata finito $A = \langle Q, \Sigma, \delta, I, F \rangle$ donde $Q = \{x^{-1}L : x^{-1}L \text{ es primo}\}$, $I = \{L\}$, $F = \{x^{-1}L : \lambda \in x^{-1}L\}$ y $\delta(x^{-1}L, a) = \{y^{-1}L : y^{-1}L \subseteq xa^{-1}L\}$.

Véase un ejemplo en la figura 3(B).

2.4.3. Autómatas maximales asociados a lenguajes finitos.

Sea L un lenguaje finito sobre un cierto alfabeto Σ . Asociamos entonces a L diferentes autómatas finitos que lo reconocen, que pasamos a definir a continuación:

Definición 11 El autómata canónico maximal de L es el autómata $MCA(L) = \langle Q, \Sigma, \delta, I, F \rangle$, donde

$$\begin{aligned}
Q &= \{\langle x, y \rangle : xy \in L\}, \\
I &= \{\langle \lambda, x \rangle : x \in L\}, \\
F &= \{\langle x, \lambda \rangle : x \in L\} \text{ y} \\
\delta(\langle x, ay \rangle, b) &= \{\langle xa, y \rangle\} \text{ si } a = b \text{ o } \emptyset \text{ si } a \neq b, \text{ para } a, b \in \Sigma.
\end{aligned}$$

$MCA(L)$ es el mayor autómata sin estados superfluos que acepta el lenguaje L . Véase un ejemplo de esta definición y de las siguientes en la figura 4.

Esta definición difiere de la establecida en [12] puesto que los autores consideran solamente autómatas finitos con un único estado inicial. Damos ahora esa definición bajo un nombre diferente:

Definición 12 *El autómata canónico maximal con un estado inicial de L es el autómata $MOICA(L) = \langle Q, \Sigma, \delta, I, F \rangle$, donde*

$$\begin{aligned}
Q &= \{\langle x, y \rangle : x \in \Sigma^+, xy \in L\} \cup \{\lambda\}, \\
I &= \{\lambda\}, \\
F &= \{\langle x, \lambda \rangle : x \in L\} \text{ si } \lambda \notin L \text{ o } \{\langle x, \lambda \rangle : x \in L\} \cup \{\lambda\} \text{ si } \lambda \in L, \\
\delta(\langle x, ay \rangle, b) &= \{\langle xa, y \rangle\} \text{ si } a = b \text{ o } \emptyset \text{ si } a \neq b, \text{ para } a, b \in \Sigma \text{ y} \\
\delta(\lambda, a) &= \{\langle a, x \rangle : ax \in L\}, \text{ para } a \in \Sigma.
\end{aligned}$$

Evidentemente $MOICA(L) = MCA(L)/\pi_I$.

Definición 13 *El árbol aceptor de prefijos de L es el autómata $PTA(L) = \langle Q, \Sigma, \delta, I, F \rangle$, donde*

$$\begin{aligned}
Q &= Pref(L), \\
I &= \{\lambda\}, \\
F &= L \text{ y} \\
\delta(x, a) &= \{xa\} \text{ si } xa \in Pref(L) \text{ o } \emptyset \text{ si } xa \notin Pref(L).
\end{aligned}$$

$PTA(L)$ es por tanto el mayor autómata finito determinista sin estados inútiles que acepta a L . Además, $PTA(L) = MOICA(L)/\pi$ siendo π tal que $\langle x_1, y_1 \rangle$ y $\langle x_2, y_2 \rangle$ pertenecen al mismo bloque si y sólo si $x_1 = x_2$ y $\{\lambda\}$ es el único elemento de su bloque. Puesto que $PTA(L)$ es un cociente de $MOICA(L)$ y éste lo es a su vez de $MCA(L)$, el primero es a su vez un cociente de éste último.

Asociaremos ahora un autómata finito a cada secuencia finita de palabras, teniendo en cuenta que, como tal secuencia, puede contener palabras repetidas, lo cual no tiene sentido cuando hablamos de lenguajes por ser estos conjuntos.

Definición 14 *Sea $S = \langle x_i, 1 \leq i \leq n \rangle$ una secuencia finita de palabras. Llamaremos autómata canónico secuencial de S al autómata finito $SCA(S) = \langle Q, \Sigma, \delta, I, F \rangle$ donde*

$$\begin{aligned}
Q &= \{\langle i, y, z \rangle : yz = x_i\}, \\
I &= \{\langle i, \lambda, x_i \rangle\}, \\
F &= \{\langle i, x_i, \lambda \rangle\} \text{ y} \\
\delta(\langle i, y, az \rangle, b) &= \{\langle i, ya, z \rangle\} \text{ si } yaz = x_i \text{ y } a = b \text{ o } \emptyset \text{ en caso contrario.}
\end{aligned}$$

Está claro que $SCA(S)$ puede contener estados superfluos. Por otra parte $MCA(S) = SCA(S)/\pi$ siendo π tal que $\langle i_1, x_1, y_1 \rangle$ e $\langle i_2, x_2, y_2 \rangle$ pertenecen al mismo bloque si y sólo si $x_1 = x_2$ e $y_1 = y_2$.

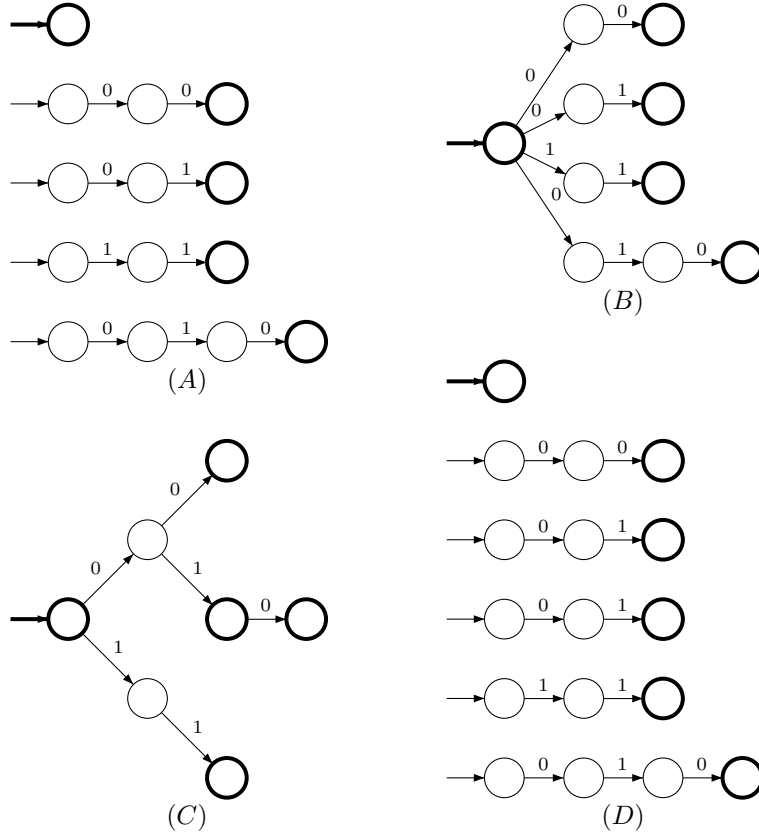


Figura 4: Siendo $L = \{\lambda, 00, 01, 11, 010\}$ se representan los siguientes autómatas: (A) $MCA(L)$, (B) $MOICA(L)$, (C) $PTA(L)$, (D) $SCA((\lambda, 00, 01, 01, 11, 010))$.

2.5. Operaciones sobre autómatas finitos.

A lo largo de este apartado describiremos algunos de los mas conocidos operadores sobre autómatas finitos.

2.5.1. Cocientes.

Dado un conjunto cualquiera C , una partición de C es cualquier colección π de subconjuntos disjuntos y no vacíos de C cuya unión es el propio C . A cada el-

emento de π se le denomina bloque. Para cada $x \in C$ llamaremos $\pi(x)$ al bloque que contiene a x . La partición trivial es aquella en la que cada bloque contiene un solo elemento. Para simplificar la descripción de una partición, en algunos casos enumeraremos solo aquellos bloques que contienen más de un elemento.

Definición 15 Sea $A = \langle Q, \Sigma, \delta, I, F \rangle$ un autómata finito y sea π una partición de Q . El autómata finito $A/\pi = \langle Q/\pi, \Sigma, \delta^\pi, Q/\pi - 2^{Q-I}, Q/\pi - 2^{Q-F} \rangle$, siendo $\delta^\pi(\pi(p), a) = \{\pi(q) : \delta(\pi(p), a) \cap \pi(q) \neq \emptyset\}$, se denomina autómata cociente de A por π .

Evidentemente si $p \in \delta(q, x)$ entonces $\pi(p) \in \delta^\pi(\pi(q), x)$ y además $p \in I \Rightarrow \pi(p) \in Q/\pi - 2^{Q-I}$ y $p \in F \Rightarrow \pi(p) \in Q/\pi - 2^{Q-F}$ por lo que en todo caso $L(A) \subseteq L(A/\pi)$.

Definición 16 Diremos que un autómata finito A es irreducible si $L(A/\pi) = L(A)$ implica que π es la partición trivial.

2.5.2. Reverso.

Definición 17 Se denomina reverso de un autómata finito $A = \langle Q, \Sigma, \delta, I, F \rangle$ al autómata finito $A^R = \langle Q, \Sigma, \delta^R, F, I \rangle$, donde $\langle p, a, q \rangle \in \delta^R \Leftrightarrow \langle q, a, p \rangle \in \delta$.

Por inducción sobre $|x|, x \in \Sigma^*$, se demuestra que $\delta^R(P_1, x) = P_2 \Leftrightarrow \delta(P_2, x^R) = P_1$ y de esto se deduce que $L(A^R) = (L(A))^R$.

Evidentemente $(A^R)^R = A$. Además, si A es aseado también lo es A^R , si A es irreducible también lo es A^R y si A es minimal también lo es A^R .

2.5.3. Subautómatas.

Un subautómata de un autómata finito $A = \langle Q, \Sigma, \delta, I, F \rangle$ es cualquier autómata $A' = \langle Q', \Sigma, \delta', I', F' \rangle$ tal que $Q' \subseteq Q, \delta' \subseteq \delta \cap (Q' \times \Sigma \times Q'), I' \subseteq I \cap Q', F' \subseteq F \cap Q'$. Obviamente, si A' es un subautómata de A entonces $L(A') \subseteq L(A)$.

Definición 18 Sea $A = \langle Q, \Sigma, \delta, I, F \rangle$ un autómata finito y sea P un subconjunto de Q . Denominaremos subautómata de A inducido por P al autómata $A_P = \langle P, \Sigma, \delta \cap (P \times \Sigma \times P), I \cap P, F \cap P \rangle$. Igualmente, dado $I' \subseteq I, F' \subseteq F$ o $\delta' \subseteq \delta$ llamaremos $A_{I \rightarrow I'}, A_{F \rightarrow F'}, A_{\delta \rightarrow \delta'}$ respectivamente a los subautómatas $\langle Q, \Sigma, \delta, I', F \rangle, \langle Q, \Sigma, \delta, I, F' \rangle, \langle Q, \Sigma, \delta', I, F \rangle$.

Así, A' es un subautómata de $A = \langle Q, \Sigma, \delta, I, F \rangle$ si y sólo si existen $I' \subseteq I, F' \subseteq F, \delta' \subseteq \delta$ y $P \subseteq Q$ tales que A' es isomorfo a $((A_{I \rightarrow I'})_{F \rightarrow F'})_{\delta \rightarrow \delta'}_P$.

En algunas construcciones será conveniente considerar para un autómata $A = \langle Q, \Sigma, \delta, I, F \rangle$ conjuntos, por ejemplo, de "estados" que no están incluidos en Q . Supondremos en esos casos que

$A_P = \langle Q \cap P, \Sigma, \delta \cap (P \times \Sigma \times P), I \cap P, F \cap P \rangle, A_{I \rightarrow I'} = \langle Q, \Sigma, \delta, I \cap I', F \rangle, A_{F \rightarrow F'} = \langle Q, \Sigma, \delta, I, F \cap F' \rangle$ y $A_{\delta \rightarrow \delta'} = \langle Q, \Sigma, \delta', I, F \rangle$. Esta generalización presenta algunas ventajas, entre otras que los operadores $I \rightarrow I', F \rightarrow F', \delta \rightarrow \delta'$ y P así definidos conmutan.

Definición 19 *Un subconjunto no vacío de estados P de un autómata finito A es superfluo si $L(A_{Q-P}) = L(A)$. Diremos que un autómata finito es conciso si no contiene estados superfluos.*

Puede ocurrir que P_1 y P_2 sean subconjuntos superfluos y sin embargo $P_1 \cup P_2$ no lo sea. Todo estado inútil es superfluo pero lo inverso no es necesariamente cierto, aunque sí lo es en el caso de los autómatas finitos deterministas. Un autómata irreducible puede tener estados superfluos.

Así mismo un conjunto de transiciones δ' , de estados iniciales I' o de estados finales F' es superfluo si, respectivamente, $L(A_{\delta \rightarrow \delta - \delta'}) = L(A)$, $L(A_{I \rightarrow I - I'}) = L(A)$, $L(A_{F \rightarrow F - F'}) = L(A)$. Diremos que un autómata finito es estrictamente conciso si no contiene estados, transiciones, estados iniciales ni estados finales superfluos.

2.5.4. Homomorfismos.

Definición 20 *Un homomorfismo de un autómata $A = \langle Q, \Sigma, \delta, I, F \rangle$ sobre un autómata $A' = \langle Q', \Sigma, \delta', I', F' \rangle$ es cualquier función $h : Q \rightarrow Q'$ tal que $h(I) \subseteq I'$, $h(F) \subseteq F'$ y $\langle p, a, q \rangle \in \delta \Rightarrow \langle h(p), a, h(q) \rangle \in \delta'$.*

Claramente, para todo $q \in Q$, $L_q \subseteq L_{h(q)}$ y $R_q \subseteq R_{h(q)}$ por lo que si existe algún homomorfismo de A sobre A' entonces necesariamente $L(A) \subseteq L(A')$.

Así, dados dos autómatas finitos A y A' , existe un homomorfismo de A sobre A' si y sólo si existe un subautómata de A' isomorfo a algún cociente de A .

Definición 21 *Sean $A = \langle Q, \Sigma, \delta, I, F \rangle$ y $A' = \langle Q', \Sigma, \delta', I', F' \rangle$ autómatas finitos y sea h un homomorfismo de A sobre A' . Sea π_h la partición de Q tal que para cada estado q , $\pi_h(q) = \{p \in Q : h(p) = h(q)\}$. Llamaremos $h(A)$ al subautómata de A' isomorfo a A/π_h .*

La definición anterior no es completamente estandar, pues también se encuentra en la literatura otra según la cual $h(A) = A'_{h(Q)}$, esto es, la imagen de A sería el subautómata de A' inducido por el subconjunto $h(Q)$ de Q' . Hemos preferido utilizar la primera por considerar que nos será de mayor utilidad.

Definición 22 *Diremos que el homomorfismo $h : A \rightarrow A'$ es inyectivo o subyectivo si lo es la función asociada $h : Q \rightarrow Q'$.*

No es suficiente con que $h : Q \rightarrow Q'$ sea inyectiva y subyectiva para que $h : A \rightarrow A'$ sea un isomorfismo; ha de cumplirse además que $h(A) = A'$.

De esa definición se desprende que un autómata finito A es irreducible si y sólo si $L(h(A)) = L(A)$ implica que h es un isomorfismo.

2.5.5. Unión.

Definición 23 *Dados $A = \langle Q, \Sigma, \delta, I, F \rangle$ y $A' = \langle Q', \Sigma, \delta', I', F' \rangle$ autómatas finitos, llamaremos al autómata $A \cup A' = \langle Q \cup Q', \Sigma, \delta \cup \delta', I \cup I', F \cup F' \rangle$, donde todas las uniones son disjuntas, unión de A y A' .*

Evidentemente $L(A \cup A') = L(A) \cup L(A')$.

2.5.6. Determinización.

Denominaremos determinización de un autómata finito al método clásico de obtención de un autómata finito determinista aseado equivalente a un autómata finito dado.

Sea $A = \langle Q, \Sigma, \delta, I, F \rangle$ un autómata finito. Diremos que un subconjunto P de Q es alcanzable si existe $x \in \Sigma^*$ tal que $\delta(I, x) = P$. Llamaremos $Alc(2^Q)$ al conjunto de partes alcanzables de Q .

Definición 24 Dado un autómata finito $A = \langle Q, \Sigma, \delta, I, F \rangle$ llamaremos al autómata finito determinista $Det(A) = \langle Alc(2^Q), \Sigma, \delta, I, Alc(2^Q) - 2^{Q-F} \rangle$ determinización de A .

Como es bien sabido, el autómata $Det(A)$ es equivalente a A , pero en el peor de los casos su número de estados es $2^{|Q|}$.

2.5.7. Minimización.

La minimización es el proceso de obtención del autómata finito determinista mínimo equivalente a un autómata finito dado.

Definición 25 Dado un autómata finito $A = \langle Q, \Sigma, \delta, I, F \rangle$ llamaremos $Min(A)$ al autómata finito determinista mínimo equivalente a A .

Sea $A = \langle Q, \Sigma, \delta, q_0, F \rangle$ un autómata finito determinista y sea π la partición de Q tal que $\pi(q) = \{p \in Q : R_p = R_q\}$. Entonces $Min(A) = A/\pi$.

Este cálculo se puede realizar en tiempo $|Q| \ln |Q|$. Sin embargo cuando tenemos de partida un autómata finito es preciso realizar primero la determinización del mismo por lo que el proceso en su conjunto es de complejidad exponencial.

Un método alternativo [3], no más eficiente pero sí elegante, de minimalizar un autómata finito se desprende de la identidad $Min(A) = Det((Det(A^R))^R)$.

2.6. Autómata Universal.

El concepto de autómata universal de un lenguaje fue establecido por Carrez en [4]. También, independientemente, aparece implícito en los trabajos de Conway [7] y de Kameda y Weiner [23]. Arnold, Dicky y Nivat en [1] formalizan el concepto y establecen su relevancia en el problema de encontrar autómatas finitos minimales equivalentes a uno dado. Lombardy y Sakarovitch en [26] lo aplicaron al estudio de la "star height" (el anidamiento mínimo del operador clausura en las expresiones regulares que representan a un lenguaje) de los lenguajes regulares. Las definiciones y proposiciones de este apartado pueden encontrarse, por ejemplo en [29] o [30], donde Polák utiliza el autómata universal como base de partida en la búsqueda de autómatas finitos minimales.

Definición 26 Sea $L \subseteq \Sigma^*$ un lenguaje cualquiera. Se define el autómata universal de L como $U = \langle U, \Sigma, \delta, I, F \rangle$, donde $U = \{x_1^{-1}L \cap \dots \cap x_n^{-1}L : n \geq 1\}$.

$0, x_1, \dots, x_n \in \Sigma^*$, $I = \{q \in U : q \subseteq L\}$, $F = \{q \in U : \lambda \in q\}$ y para dos estados cualesquiera de U y cualquier símbolo a de Σ , $q \in \delta(p, a) \Leftrightarrow q \subseteq a^{-1}p$.

Véase como ejemplo la figura 5.

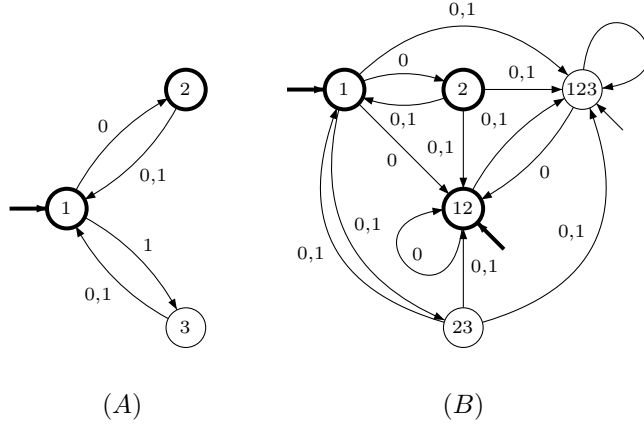


Figura 5: (A) Autómata finito determinista mínimo para $L = ((0 + 1)^2)^*(0 + \lambda)$. Los estados se corresponden con los siguientes valores de derivadas de L : $1 := L$, $2 := (0 + 1)L + \lambda$, $3 := (0 + 1)L$. (B) Autómata universal para el mismo lenguaje. Los estados se corresponden con los siguientes valores de intersecciones entre derivadas: $1 := L$, $2 := (0 + 1)L + \lambda$, $23 := (0 + 1)L$, $12 := (0 + 1)^*0 + \lambda$, $123 := (0 + 1)^*0$.

Proposición 27 Sean $\langle p_1, q_1 \rangle, \dots, \langle p_n, q_n \rangle$ las soluciones maximales de la inequación $L_1 L_2 \subseteq L$ y sea $U = \langle U, \Sigma, \delta, I, F \rangle$ el autómata universal de L . Entonces $\{q_1, \dots, q_n\} = U$.

Demostración. Por el requisito de maximalidad $q_i = \bigcap_{x \in p_i} x^{-1}L$. Por el mismo motivo $p_i = \bigcup_{x \in q_i} [x]_{R_L} : yx \in L$. ■

Proposición 28 Sea $U = \langle U, \Sigma, \delta, I, F \rangle$ el autómata universal de L . Entonces U acepta a L .

Demostración. De la proposición anterior se desprende que para cada $q_i \in U$, $L_{q_i} = p_i$ y $R_{q_i} = q_i$, por lo que $L(U) = \bigcup_{i: q_i \in F} p_i = [y]_{R_L} : y \in L = L$. ■

Proposición 29 $L \in \mathcal{L}_3 \iff$ su autómata universal U es un autómata finito.

Demostración. La implicación \Leftarrow es obvia por la proposición anterior. Para demostrar la implicación \Rightarrow es suficiente con observar que si $L \in \mathcal{L}_3$ el número de conjuntos diferentes de la forma $x^{-1}L$ es finito por lo que también es finito el número de intersecciones diferentes de los mismos, esto es, de estados de U . ■

Proposición 30 Sea $L \in \mathcal{L}_3$, sean $D = \langle Q_D, \Sigma, \delta_D, I_D, F_D \rangle$ su autómata finito determinista mínimo y $U = \langle U, \Sigma, \delta, I, F \rangle$ su autómata universal. Entonces $|U| \leq 2^{|Q_D|}$.

Demostración. Dado que $Q_D = \{x^{-1}L : x \in \Sigma^*\}$ tenemos que $U = \{ \bigcap_{q \in P} q : P \subseteq Q_D \}$, de donde se deduce trivialmente el enunciado. ■

Proposición 31 Un autómata finito $A = \langle Q_A, \Sigma_A, \delta_A, I_A, F_A \rangle$ reconoce un subconjunto de L si y sólo si existe un homomorfismo de A en U .

Demostración. La implicación \Leftarrow es trivial por la propia definición de homomorfismo entre autómatas finitos. Para demostrar la implicación \Rightarrow se define la función φ que asigna a cada estado $q \in Q_A$ el estado de U $\varphi(q) = \bigcap_{x \in L_q} x^{-1}L$. La función φ es un homomorfismo por lo que la implicación queda demostrada. ■

3. Inferencia gramatical de lenguajes regulares.

3.1. Inferencia gramatical.

Se llama inducción al proceso mediante el cual podemos pasar de lo particular a lo general, de observaciones concretas a una proposición que da cuenta de las mismas. El proceso de aprendizaje mediante inducción y en particular su aplicación a la demostración de proposiciones, ha sido objeto de estudio y controversia desde su primera formulación, realizada por Aristóteles. En líneas generales el paradigma de la inferencia inductiva puede ser descrito como sigue: Un agente, enfrentado con algún fenómeno, recoge datos experimentales, bien de forma activa mediante la realización de experimentos, o bien de forma pasiva mediante la mera observación. A continuación el agente formula una hipótesis consistente con los datos que pretende explicar o describir el fenómeno estudiado. Si la hipótesis satisface determinados criterios, por ejemplo permite vaticinar con éxito los resultados de nuevas observaciones del fenómeno, es aceptada como teoría.

Cuando los fenómenos objeto de estudio son o se representan mediante lenguajes la inferencia inductiva recibe el nombre de inferencia gramatical.

Para definir un problema de inferencia gramatical es necesario especificar los siguientes conceptos: Dominio de objetos, espacio de hipótesis, método de presentación, método de inferencia y criterio de éxito. Hablaremos brevemente sobre ellos.

El dominio de objetos es el conjunto de lenguajes que se pretende inferir, entre los cuales pensamos que está aquel que buscamos como resultado final del proceso. En principio podría considerarse como dominio de objetos cualquier familia de lenguajes, pero consideraciones tales como la necesidad de poder averiguar si una palabra pertenece o no al lenguaje en cuestión reducen en la práctica los posibles dominios a las subclases enumerables de la clase de los lenguajes recursivos.

El espacio de hipótesis es un conjunto de descriptores de los elementos del dominio de objetos tal que contiene al menos un descriptor para cada elemento del dominio. Los elementos de este espacio pueden ser gramáticas, autómatas, máquinas de Turing, etc.

El método de presentación es el tipo de información que se utiliza para aprender. En el caso del aprendizaje activo este puede ser muy diverso e incluir cosas como consultas a oráculos, solicitudes de determinada información al "maestro", etc. En el caso del aprendizaje pasivo se distinguen dos tipos básicos de presentación de la información, presentación positiva y presentación completa. Una presentación positiva es una secuencia, posiblemente infinita, de todas las palabras del lenguaje a aprender. Una presentación completa es una sucesión de todas las palabras sobre el alfabeto en que está descrito el lenguaje etiquetadas según su pertenencia o no al mismo. En la práctica diremos que se utiliza presentación positiva si se aprende exclusivamente a partir de ejemplos, esto es, de palabras del lenguaje y diremos que se utiliza presentación completa si se aprende usando también contraejemplos, esto es palabras que no pertenecen al

lenguaje.

Un método de inferencia es una función que hace corresponder una hipótesis a cada lenguaje finito, en caso de trabajar con presentación positiva, o a cada par de lenguajes finitos disjuntos en caso de hacerlo con presentación completa. Tradicionalmente los métodos de inferencia se dividen en dos familias, enumerativos y constructivos. Los métodos enumerativos se basan en una enumeración del espacio de hipótesis y proporcionan como resultado la primera hipótesis compatible con los datos. Los métodos constructivos elaboran su salida a partir de los datos de entrada. Entre los primeros resultados obtenidos en el estudio de la inferencia gramatical se encuentra la demostración de que todo lo que puede ser aprendido mediante métodos constructivos puede ser también aprendido mediante métodos enumerativos. Algunas propiedades consideradas en general deseables en un método de inferencia son la consistencia (en todo caso la hipótesis de salida clasifica correctamente todos los datos de entrada) y la conservación (si la hipótesis obtenida como salida para un conjunto de datos D_1 es compatible con un superconjunto D_2 de D_1 entonces el algoritmo produce la misma salida cuando tiene como entrada D_2).

El concepto de criterio de éxito pretende establecer un sistema de evaluación de la eficacia del método de inferencia. Las primeras ideas sobre el mismo fueron extraídas del concepto de límite de una sucesión por lo que se contempla la labor de inferencia como un proceso infinito cuyo éxito está relacionado con su comportamiento en el límite. Posteriormente se han definido otros criterios tanto de tipo probabilístico como basados en la cantidad de datos necesarios para aprender con determinada aproximación. El criterio de éxito más utilizado es el de identificación en el límite, introducido por Gold [14][15], según el cual un algoritmo de inferencia identifica en el límite una cierta clase de lenguajes si para cada elemento L de la misma existe un conjunto de datos tal que para cualquier superconjunto del mismo compatible con L el algoritmo obtiene como hipótesis la misma representación de L . Otros criterios de éxito son los de concordancia en el límite [13], ε -identificación en el límite [34], identificación PAC (probablemente aproximadamente correcta) [33], etc.

El marco actual de la inferencia gramatical fue establecido por Gold, Feldman y otros a finales de la década de los sesenta y principios de la de los setenta del siglo XX. Entre sus principales trabajos destacan aquellos que marcan el límite de lo que puede ser aprendido mediante los diferentes tipos básicos de presentación. Se demuestra entonces que mediante presentación completa pueden ser aprendidas las subclases enumerables de la clase de los lenguajes recursivos, mientras que mediante presentación positiva sólo pueden ser aprendidas aquellas clases de lenguajes cuyos elementos pueden ser enumerados de forma que para cada lenguaje exista un subconjunto finito no incluido en ninguno de los lenguajes anteriores en la enumeración. Esto supone, entre otras cosas, que ninguna clase de lenguajes superfinita (que contiene todos los lenguajes finitos y al menos uno infinito) puede ser aprendida mediante presentación positiva. Entre las clases superfinitas se encuentran todas las de la jerarquía de Chomsky, en particular la de los lenguajes regulares así como algunas de sus subclases más relevantes como, por ejemplo, la de los lenguajes sin estrella (star-free en

la terminología inglesa).

3.2. Inferencia gramatical de lenguajes regulares.

3.2.1. Conceptos básicos.

Dado un lenguaje L llamaremos muestra positiva de L a cualquiera de sus subconjuntos finitos y llamaremos muestra negativa de L a cualquier subconjunto finito de su complementario. Una muestra de L es un par formado por una muestra positiva y una muestra negativa. El tamaño de una muestra es la suma de las longitudes de las palabras que contiene.

Un algoritmo de inferencia es cualquier algoritmo tal que para cualquier muestra de entrada obtiene como salida una representación de un lenguaje compatible con la muestra, esto es, un lenguaje que incluye a la muestra positiva y es disjunto con la negativa.

Un algoritmo infiere una familia de lenguajes si para cada muestra de entrada obtiene como salida un lenguaje de esa familia y además cada lenguaje de la familia se obtiene como salida para al menos una muestra.

En el ámbito de un determinado algoritmo de inferencia una muestra S es característica de un lenguaje L si toda muestra que incluya a S produce como salida una representación de L . Un algoritmo infiere en el límite una familia de lenguajes si para cada lenguaje de la familia existe alguna muestra característica.

Una familia de lenguajes \mathcal{L} es identificable en el límite en tiempo y datos polinómicos si para todo sistema finito de representación R de los lenguajes de \mathcal{L} existe algún algoritmo \mathcal{A} tal que \mathcal{A} infiere en el límite \mathcal{L} , \mathcal{A} es polinómico respecto al tamaño de la muestra de entrada y para cada lenguaje existe una muestra característica de tamaño polinómico respecto a su representación en R .

De forma menos restrictiva, una familia de lenguajes \mathcal{L} es identificable en el límite en tiempo y datos polinómicos respecto a la representación R [18] si existen algoritmos $\mathcal{A}_1, \mathcal{A}_2$ tales que \mathcal{A}_1 infiere en el límite \mathcal{L} , \mathcal{A}_1 es polinómico respecto al tamaño de la muestra de entrada y \mathcal{A}_2 es un algoritmo que para cada lenguaje calcula una muestra característica en tiempo polinómico respecto al tamaño de cualquier representación del lenguaje en R . El hecho de que una familia de lenguajes sea identificable en el límite en tiempo y datos polinómicos respecto a una representación R no significa que esa familia sea en sí identificable en el límite en tiempo y datos polinómicos, por lo que este concepto solo tiene sentido si la representación en cuestión puede ser considerada razonable”.

La familia de los lenguajes regulares no es identificable en el límite en tiempo y datos polinómicos, pues no lo es respecto a su representación mediante autómatas finitos [18]. Sin embargo sí lo es respecto a su representación mediante autómatas finitos deterministas que evidentemente puede ser considerada razonable que es una de las estructuras más estudiadas y utilizadas en teoría de lenguajes, además de ser una de las más simples y de las que presentan más propiedades interesantes.

3.2.2. El algoritmo Trakhtenbrot-Barzdin-Gold.

En 1978 Gold demostró [15] que el problema de dada una muestra encontrar el autómata finito determinista mas pequeño compatible con ella es $NP - duro$. Este resultado supone evidentemente un freno al aprendizaje de lenguajes regulares. De hecho, este resultado, junto con otros resultados negativos ya mencionados condujo a que durante las dos siguientes décadas los métodos de inferencia propuestos son en su inmensa mayoría heurísticos que utilizan presentación positiva e infieren subclases no caracterizadas de la familia de los lenguajes regulares. Esta situación no deja de ser paradójica: Por una parte, el mismo Gold había demostrado que la clase de los lenguajes regulares no es inferible mediante presentación positiva y por otra parte del resultado de Gold se desprende únicamente que un algoritmo consistente en obtener cada vez que se recibe un nuevo dato un autómata finito determinista, compatible con el conjunto actual de datos procesados y minimal en cuanto al número de estados, no sería eficiente. Esta reacción resulta tanto mas sorprendente si se tiene en cuenta que en el mismo artículo antes referenciado Gold propone un algoritmo polinómico que permite identificar en el límite cualquier lenguaje regular.

El algoritmo expuesto por Gold había sido en realidad propuesto ya por Trakhtenbrot y Barzdin [32] en 1973, pero había pasado inadvertido hasta que fué redescubierto por Gold. Este algoritmo se realiza en dos etapas. En la primera se calcula a partir de los datos una matriz de caracterización de estados y en la segunda se obtiene a partir de esta un autómata finito determinista presentado inicialmente en forma de máquina de Mealey. Veamos a continuación una somera descripción del mismo que comenzaremos describiendo los conceptos básicos utilizados.

Una matriz de caracterización de estados sobre un alfabeto Σ es un triplete $\langle S, E, T \rangle$ donde S y E son lenguajes finitos sobre Σ a cuyos elementos se denomina respectivamente estados y experimentos y $T : (S \cup S\Sigma)E \rightarrow \{0, 1, 2\}$ es una función que asigna una etiqueta a cada palabra de $(S \cup S\Sigma)E$.

Los datos D_+ y D_- contenidos en la matriz son:

$$\begin{aligned} x \in D_+ &\Leftrightarrow \exists u \in (S \cup S\Sigma), \exists v \in E : x = uv \wedge T(uv) = 1 \\ x \in D_- &\Leftrightarrow \exists u \in (S \cup S\Sigma), \exists v \in E : x = uv \wedge T(uv) = 0 \end{aligned}$$

Para el resto de palabras $T(uv) = 2$. Cada elemento u de $S \cup S\Sigma$ define una hilera llamada $row(u)$. Se dice que $row(u)$ es obviamente diferente de $row(v)$, escrito $row(u) \not\cong row(v)$, si existe un experimento $e \in E$ tal que $T(ue) + T(ve) = 1$.

Una matriz de caracterización de estados está cerrada si ninguna hilera de $S\Sigma - S$ es obviamente diferente de cada hilera de S .

Dada una muestra $\langle D_+, D_- \rangle$ el algoritmo comienza asignando a S el valor $\{\lambda\}$ y a E el conjunto de sufijos propios de palabras de la muestra. Se construye entonces la matriz de caracterización de estados correspondiente. En cada uno de los siguientes pasos se selecciona una palabra u de $S\Sigma - S$ tal que su hilera es obviamente diferente de cada una de las correspondientes a palabras de S y se añade al conjunto S . Se actualiza entonces la matriz. El proceso continúa hasta que la matriz esté cerrada.

Finalmente se construye el autómata finito determinista cuyos estados son las palabras de S y cuyas transiciones son de forma que $\delta(u, a) = v$ siendo $v = ua$ si $ua \in S$ o si no siendo v una palabra de S tal que $row(v)$ no es obviamente diferente de $row(ua)$. Por último, el estado x es final si $T(x) = 1$.

Véase un ejemplo en la figura 6.

	0	λ	00	001	01	1	1001	110	10
λ	0	0	1	2	0	2	2	2	2
0	1	0	2	1	2	0	1	0	2
1	2	2	1	2	0	2	2	2	2
00	2	1	2	2	1	2	2	2	2
01	2	0	2	1	2	2	2	2	0
000	2	2	2	2	2	1	2	2	2
001	2	2	2	2	2	2	2	2	2

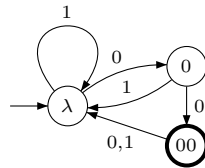


Figura 6: Proceso mediante el algoritmo de Gold de la muestra $S = \langle S_+, S_- \rangle$, siendo $S_+ = \{00, 100, 0001, 01001\}$ y $S_- = \{\lambda, 0, 01, 101, 0110\}$. Para el cálculo de la matriz se incluye primero la fila para λ . A continuación las filas para 0 y 1, siendo ésta última equivalente a la de λ . Se incluyen entonces las filas para 00 y 01, siendo ésta última equivalente a la de λ . Por último se incluyen las filas para 000 y 001, ambas equivalentes a λ , con lo que la matriz queda cerrada. A partir de la matriz se construye el autómata

Puede ocurrir que el autómata calculado no sea compatible con la muestra de entrada, en cuyo caso el algoritmo produce como salida el árbol aceptor de prefijos de la muestra positiva.

En cualquier caso, Gold demostró que este algoritmo infiere polinómicamente en el límite la clase de los lenguajes regulares, esto es, que construye un autómata finito determinista en tiempo polinómico respecto al tamaño de la muestra, que para cada lenguaje regular L existe una muestra característica S_L de tamaño polinómico respecto al número de estados de su autómata finito determinista mínimo y que si una muestra para L contiene a S_L entonces el algoritmo produce como salida el autómata finito determinista mínimo para L .

3.2.3. El algoritmo RPNI.

Como ya hemos comentado, tras la publicación de los resultados de Gold el problema de la inferencia en el límite de la clase de los lenguajes regulares mediante presentación completa quedó prácticamente abandonado. Esta situación

permanece inalterada hasta que en 1992 Oncina y García [28] presentan el algoritmo RPNI (regular positive and negative inference) y, simultáneamente, Lang [24] modifica el algoritmo Trakhtenbrot-Barzdin-Gold para garantizar la consistencia con la muestra de entrada del autómata obtenido aun en el caso de que esta no sea completa, obteniendo así un algoritmo que se comporta exactamente igual que el RPNI.

Desde entonces el RPNI ha sido sin duda el algoritmo más relevante y el más utilizado en la inferencia de lenguajes regulares y ha dado lugar al desarrollo de numerosas variantes algunas de las cuales mencionaremos más adelante.

Una de las características más destacables del RPNI es la introducción del sistema de inferencia mediante fusión de estados, método del que posteriormente han hecho uso la mayoría de algoritmos y heurísticos de inferencia gramatical de lenguajes regulares.

Dada una muestra, RPNI comienza construyendo una máquina de Moore cuyos estados son los prefijos de la muestra y cuyas transiciones son de la forma $\delta(x, a) = xa$. A cada estado se le asigna la salida 0, 1 o ? según esté asociado a una palabra de la muestra positiva, a una de la muestra negativa, o a un prefijo de alguna palabra de la muestra que no es a su vez una palabra de la misma. A continuación intenta fusionar cada estado con alguno de los anteriores en orden lexicográfico. Si esta fusión da lugar a la aparición de no determinismo han de realizarse además todas las fusiones necesarias para restablecer el determinismo. Si como resultado de este proceso se produce la fusión de un estado con salida 0 con otro con salida 1 se descarta la fusión inicial y se prueba con la siguiente. La fusión de un estado con salida s con otro con salida ? da como resultado un estado con salida s .

Véase un ejemplo en la figura 7.

La complejidad de este algoritmo está acotada por mn^2 siendo m el tamaño de la máquina de Moore inicial y n el tamaño del autómata final. Además, para cada lenguaje regular L existe una muestra característica de tamaño acotado por $n^3 |\Sigma|$ siendo n el número de estados del autómata finito determinista mínimo para L . Siempre que la muestra de entrada contenga una muestra característica de un lenguaje regular L RPNI produce como salida el autómata finito determinista mínimo para L . En cualquier otro caso RPNI produce como salida el autómata finito determinista mínimo de un lenguaje compatible con la muestra.

Las prestaciones del RPNI han hecho de él un algoritmo de referencia en inferencia gramatical en él que se han inspirado otros algoritmos y heurísticos basados igualmente en el método de inferencia por fusión de estados. De hecho, desde su aparición, la mayor parte de trabajos presentados en el campo de la inferencia gramatical de lenguajes regulares son heurísticos que pretenden mejorar algunos aspectos del RPNI. Destacan entre estos aquellos basados en el método EDSM (evidence driven state merging) [25]. La idea general de este método consiste en utilizar la información proporcionada por la muestra de entrada para guiar el orden en que se realizan las fusiones. Así, por ejemplo, si la fusión de dos estados da lugar a sucesivas fusiones por determinización de numerosos pares de estados etiquetados con la misma salida es razonable suponer que ambos estados son efectivamente el mismo.

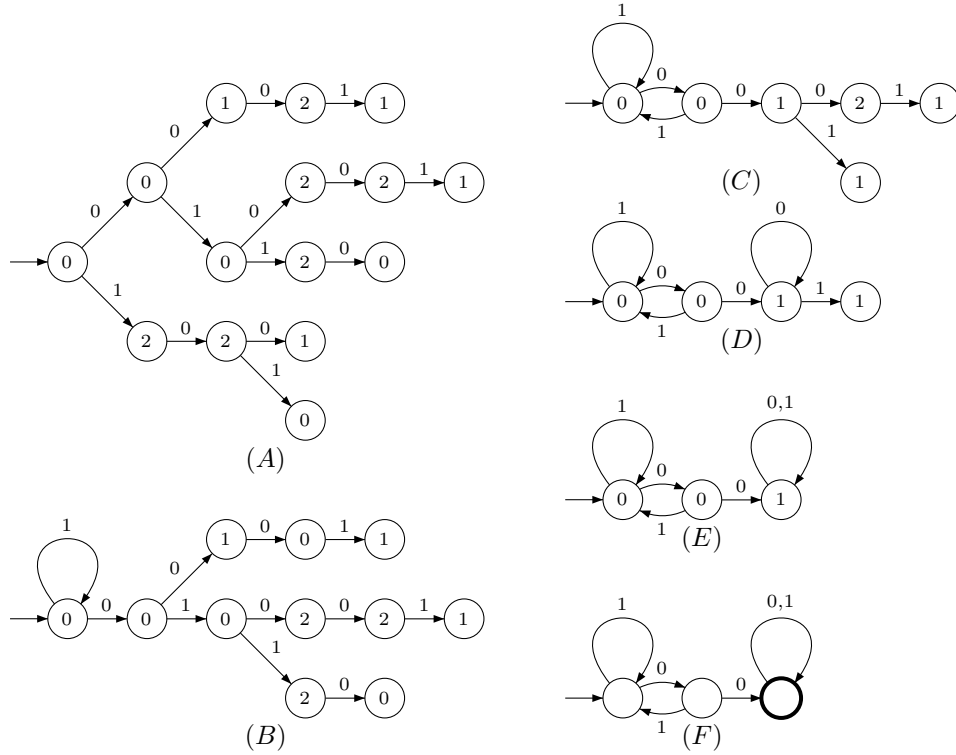


Figura 7: Proceso mediante el RPNi de la muestra $S = \langle S_+, S_- \rangle$, siendo $S_+ = \{00, 100, 0001, 01001\}$ y $S_- = \{\lambda, 0, 01, 101, 0110\}$. (A) Árbol aceptor de prefijos representado mediante una máquina de Moore. (B) Tras descartar la unión de los estados asociados a los prefijos λ y 0 , ya que por determinización implicaría la fusión prohibida de los estados λ y 00 , se fusionan los estados λ y 1 dando lugar por determinización a la máquina de la figura. (C) El estado 00 no puede fusionarse con ninguno de los anteriores. Se fusionan entonces los estados λ y 01 . (D) El estado 000 no puede fusionarse ni con λ ni con 0 . Se fusiona con 00 . (E) El estado 0001 no puede fusionarse ni con λ ni con 0 . Se fusiona con 00 . (F) La máquina de Moore obtenida es transformada en este autómata.

3.2.4. Inferencia no determinista.

El problema de aprender lenguajes regulares utilizando como representación autómatas finitos es *NP - duro*. Esto es debido a que el tamaño de la menor muestra característica crece de forma no acotada polinómicamente respecto al número de estados de un autómata finito minimal en algunas subfamilias de lenguajes regulares. Este resultado ha desalentado la investigación hasta el punto de que solo recientemente se han empezado a publicar algunos resultados. Sin embargo, no conviene olvidar que, de hecho, este resultado implica que la clase de los lenguajes regulares no es identificable en tiempo y datos polinómicos, de forma que sólo asumiendo que los autómatas finitos deterministas son una re-

presentación razonable” de los mismos puede hablarse de identificación mediante estos de \mathcal{L}_3 en tiempo y datos polinómicos.

La utilización de autómatas finitos deterministas tiene en cualquier caso el inconveniente de que para muchas familias sencillas de \mathcal{L}_3 el número de estados del autómata finito determinista mínimo de cada lenguaje es exponencial respecto al de los autómatas finitos deterministas mínimos para ese lenguaje. Así pues ¿Por qué no utilizar algoritmos polinómicos que infieran en el límite la clase de los lenguajes regulares utilizando como representación autómatas finitos?. Al fin y al cabo estos algoritmos siempre pueden ser modificados para ofrecer como resultado el autómata finito obtenido por el RPNI u otro algoritmo similar en aquellos casos en que no han podido obtener uno de menor tamaño.

El mas conocido de los algoritmos que infieren autómatas finitos no deterministas es el DeLeTe2 [9][10][11]. Este algoritmo utiliza como sistema de representación RFSAs. \mathcal{L}_3 no es identificable en el límite en tiempo y datos polinómicos respecto a su representación mediante RFSAs, pero sin embargo DeLeTe2 construye un RFSA consistente con la muestra de entrada en tiempo polinómico y para cada lenguaje regular es posible construir una muestra característica de tamaño polinómico respecto al tamaño del autómata finito determinista mínimo para el lenguaje.

Hay que señalar no obstante, que el uso de RFSAs no elimina en muchos casos el inconveniente señalado para los autómatas finitos deterministas, pues también para muchas familias de \mathcal{L}_3 el número de estados de los RFSAs minimales de cada lenguaje es exponencial respecto al de los autómatas finitos mas pequeños para ese mismo lenguaje. Este es el caso, por ejemplo, de la familia indexada por $n \{0^i : i \text{ admite algún divisor mayor que } 1 \text{ y menor que } n\}$ en la que el número de estados crece polinómicamente respecto a n en el caso de los autómatas finitos y exponencialmente en caso de los RFSAs. Véase un ejemplo de autómatas para un lenguaje de esta familia en la figura 8.

El algoritmo DeLeTe2 es una adaptación del algoritmo de Gold al trabajo con RFSAs. Se basa en el cálculo de una relación, que denominaremos \prec^* y que definimos a continuación:

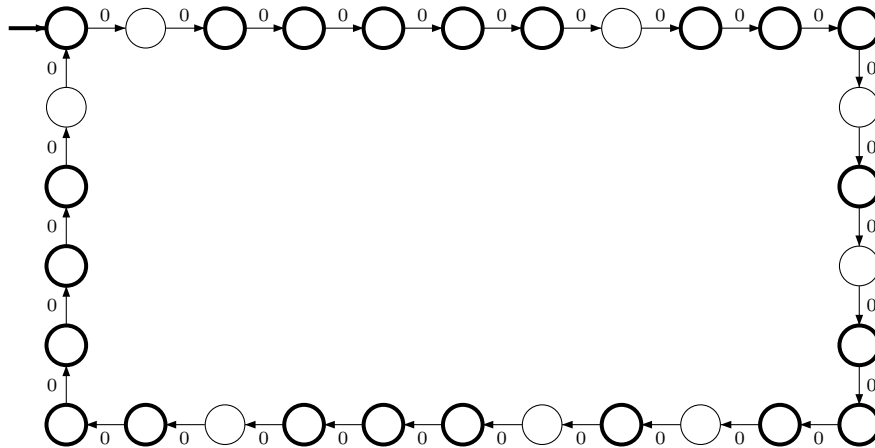
Sea $S = \langle S_+, S_- \rangle$ una muestra. Se define la relación \prec sobre $Pref(S_+)$ de forma que $u \prec v$ si no existe ninguna palabra w tal que $uw \in S_+$ y $vw \in S_-$. La relación \prec^* es la clausura transitiva de \prec . Se define además la equivalencia \simeq^* de forma que $u \simeq^* v$ si $u \prec^* v$ y $v \prec^* u$.

Cuando DeLeTe2 recibe como entrada la muestra S actúa de la siguiente manera:

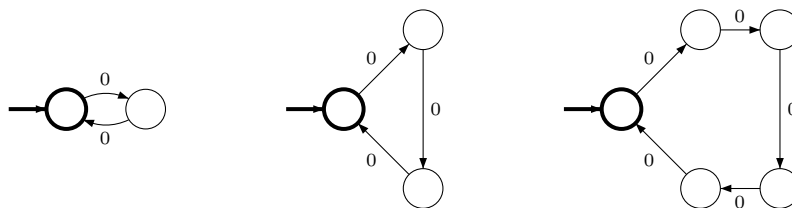
Calcula en primer lugar la secuencia $Pref$ de prefijos de S_+ ordenados en orden canónico e inicializa el autómata $A = \langle Q, \Sigma, \delta, I, F \rangle$ al valor $\langle \emptyset, \Sigma, \emptyset, \emptyset, \emptyset \rangle$.

A continuación, para cada elemento u de $Pref$ mira si existe algún estado v en Q tal que $u \simeq^* v$. Si lo hay, elimina de $Pref$ todas las palabras de la forma $u\Sigma^*$. En caso contrario añade u a Q , si $u \prec^* \lambda$ añade u a I , si $u \in S_+$ añade u a F y por último añade a δ todas las transiciones de la forma $\langle v, a, u \rangle$ tales que $v \in Q$, $va \in Pref$ y $u \prec^* va$ y de la forma $\langle u, a, v \rangle$ tales que $v \in Q$, $ua \in Pref$ y $v \prec^* ua$.

El proceso finaliza en el momento en que el autómata A es consistente con



(A)



(B)

Figura 8: (A) Autómata finito determinista mínimo, que es además un RFSA minimal, para el lenguaje formado por aquellas palabras cuya longitud es múltiplo de algún número i siendo $1 < i \leq n$ y siendo $n = 5$. (B) Autómata finito minimal para el mismo lenguaje. Para $n = 11$ el RFSA minimal tiene 2310 estados mientras que existen autómatas finitos para ese mismo lenguaje con 28 estados.

la muestra S . Puede verse un ejemplo en la figura 9.

Otros algoritmos que infieren autómatas finitos son NRPNI [2], variante del RPNI que trabaja con RFSAs y U_{hc} y U_{hc} [6], variantes del RPNI que trabajan con autómatas finitos inambiguos.

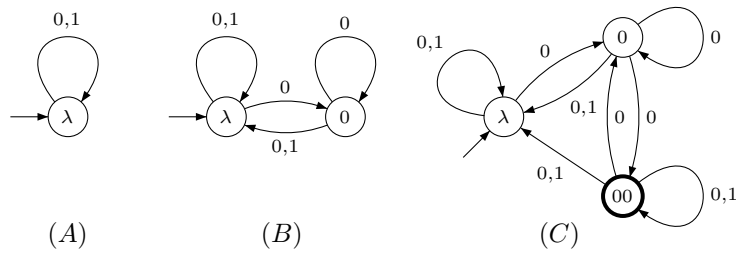


Figura 9: Proceso mediante el DeLeTe2 de la muestra $S = \langle S_+, S_- \rangle$, siendo $S_+ = \{00, 100, 0001, 01001\}$ y $S_- = \{\lambda, 0, 01, 101, 0110\}$. El conjunto de prefijos es $Pref = \{\lambda, 0, 1, 00, 01, 10, 000, 010, 100, 0001, 0100, 01001\}$. (A) Añadimos a Q y a I el estado λ . Puesto que $\lambda \prec^* \lambda, 0$ ($\lambda \prec 0$ y para todo x $\lambda.x \prec 0.x$), añadimos la transición $\langle \lambda, 0, \lambda \rangle$, y puesto que $\lambda \prec^* \lambda, 1$, añadimos la transición $\langle \lambda, 1, \lambda \rangle$. (B) Tomamos el prefijo 0. No es equivalente a λ , pues $0 \not\prec^* \lambda$, así que añadimos a Q el estado 0. Puesto que $0 \prec^* \lambda, 0$ añadimos la transición $\langle \lambda, 0, 0 \rangle$. Puesto que $\lambda \prec^* 0, 0$ y $\lambda \prec^* 0, 1$, añadimos las transiciones $\langle 0, 0, \lambda \rangle$ y $\langle 0, 1, \lambda \rangle$. Como además $0 \prec^* 0, 0$ añadimos la transición $\langle 0, 0, 0 \rangle$. (C) $1 \simeq^* \lambda$ por lo que suprimimos de $Pref$ las palabras que comienzan por 1. Consideramos entonces 00 obteniendo el autómata de la figura que, puesto que es compatible con la muestra, será la salida del algoritmo.

4. Minimalización de Autómatas Finitos.

4.1. Introducción.

Si estamos interesados en representar una familia de objetos mediante autómatas finitos es evidentemente conveniente que el tamaño de la representación sea lo menor posible, esto es, que los autómatas finitos en cuestión tengan el menor número posible de estados. Esto resulta particularmente evidente cuando queremos aplicar a esos autómatas algoritmos con una cierta complejidad.

Lo ideal sería representar cada lenguaje mediante un autómata finito minimal, esto es, tal que no existe otro con menos estados que reconozca el mismo lenguaje. Sin embargo, los autómatas obtenidos mediante algoritmos o sistemas heurísticos de inferencia no tienen en general esa propiedad, por lo que cabe plantear el problema de dado un autómata finito, encontrar uno minimal equivalente.

Para los autómatas finitos deterministas existen algoritmos eficientes (de complejidad $O(n \log n)$) y sencillos que producen como resultado el autómata finito determinista mínimo para el lenguaje en cuestión. Por desgracia éste no es el caso cuando se trata de autómatas finitos en general, pues tanto el problema de encontrar un autómata finito minimal equivalente a uno dado como el de averiguar si un autómata finito es minimal son *PSPACE* – completos.

No queda entonces mas remedio que enfocar el problema de la obtención de autómatas finitos pequeños desde otros puntos de vista como, por ejemplo, intentar que nuestros autómatas finitos posean determinadas propiedades que garanticen que su tamaño no es innecesariamente grande. Algunas de estas propiedades deseables son la irreducibilidad, la concisión y la primalidad. A su estudio están dedicadas las siguientes secciones.

4.2. Minimalidad.

4.2.1. Autómatas finitos minimales.

Como es bien sabido, el autómata finito determinista mínimo de un lenguaje regular no es en general minimal respecto al número de estados entre los autómatas finitos que reconocen ese lenguaje. De hecho su tamaño puede llegar a ser de hasta 2^n , siendo n el número de estados de un autómata minimal para el lenguaje correspondiente. Por ejemplo, para los lenguajes de la forma $\Sigma^*0\Sigma^m$ sus autómatas finitos deterministas mínimos tienen tamaño 2^{m+1} , mientras que son también aceptados por autómatas con $m + 2$ estados. En la figura 10 se muestra el caso en que $m = 2$.

Además, mientras que el autómata finito determinista es único, para cada lenguaje regular puede haber varios autómatas minimales diferentes. En la figura 11 podemos ver dos autómatas minimales para el lenguaje $\{01, 02, 12\}$.

Aunque el problema de encontrar un autómata minimal equivalente a un autómata finito dado es *PSPACE* – completo, se han propuesto algoritmos no

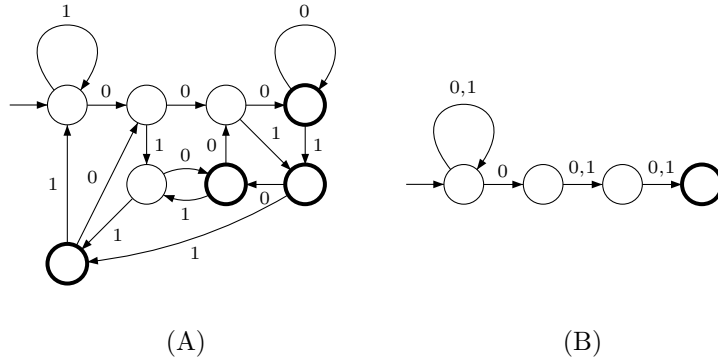


Figura 10: (A) Autómata finito determinista mínimo para $\{0, 1\}^* 0 \{0, 1\}^2$. (B) Un autómata minimal para el mismo lenguaje.

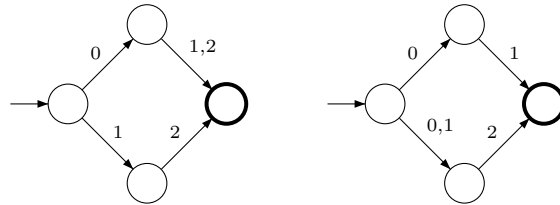


Figura 11: Dos autómatas minimales para el lenguaje $\{01, 02, 12\}$.

excesivamente complejos que en ciertos casos consiguen aproximaciones relativamente buenas.

El primer avance significativo lo establecieron Arnold, Dicky y Nivat en [1] donde demuestran que cualquier autómata minimal para un lenguaje regular L es isomorfo a un subautómata del autómata universal para L . Este resultado fija un espacio de búsqueda finito pero muy grande, pues el número de estados del autómata universal puede llegar a ser de hasta 2^n , siendo n el número de estados del autómata finito determinista mínimo.

Polák en [30] expone una serie de condiciones, necesarias unas y suficientes otras, para que el subautómata del autómata universal inducido por un subconjunto de estados acepte el lenguaje completo. Estas condiciones pueden ser testeadas en tiempo polinómico respecto al tamaño del autómata universal y pueden ser utilizadas como guías en la búsqueda sistemática de minimales. Evidentemente, esta aproximación no es en absoluto eficiente en la práctica.

Kameda y Weiner, en [23], han desarrollado una teoría para abordar el problema de la minimalización. Para cada autómata finito definen una matriz que denominan mapa reducido de estados del autómata, de la que demuestran que es isomorfa a la de cualquier otro autómata finito para el mismo lenguaje, ad-

mitiendo tan solo permutaciones de las filas y columnas. A partir de este mapa reducido pueden obtener una secuencia de autómatas ordenada en cuanto a número de estados que son autómatas minimales. Sin embargo la construcción no garantiza que en todos los casos el autómata obtenido sea equivalente al original, por lo que esto debe comprobarse, comprobación que como sabemos es de complejidad $PSPACE - completo$.

Otros autores presentan aproximaciones cuyo objetivo no es ya la búsqueda de autómatas minimales sino simplemente de autómatas mas pequeños equivalentes al original. Algunos de estos algoritmos han probado bastante eficacia en ciertos casos, pero están lejos de garantizar siquiera la obtención de autómatas polinómicos en tamaño respecto al autómata finito determinista mínimo para el lenguaje.

4.2.2. Minimalidad relativa.

En ciertos campos de la teoría de lenguajes, tales como la inferencia inductiva o la representación de lenguajes finitos, podemos encontrar el problema de calcular un autómata finito lo mas pequeño posible que acepte todas las palabras de un cierto lenguaje finito M_+ y rechace todas las de otro lenguaje finito M_- . En la figura 12 podemos ver algunas soluciones al caso en que $M_+ = \{010, 101\}$, $M_- = \{00, 11\}$.

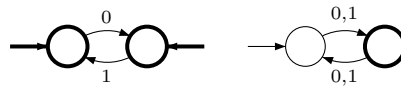


Figura 12: Dos autómatas minimales compatibles con la muestra $M_+ = \{010, 101\}$, $M_- = \{00, 11\}$.

Una solución será minimal si no existe otro autómata finito con menos estados, para el mismo lenguaje o para cualquier otro, compatible con la muestra. La obtención de una solución minimal [15] es un problema $NP - duro$ aún en el caso de restringir el espacio de búsqueda al de las autómatas finitos deterministas. Si prescindimos de la condición de minimalidad el problema deviene trivial. Una vez más surge la necesidad de buscar una situación de compromiso entre ambos extremos. Para el caso de los autómatas finitos deterministas se han diseñado algoritmos polinómicos [28][24] que proporcionan una solución y además garantizan que ésta es mínima cuando los conjuntos M_+ y M_- satisfacen determinadas condiciones, pero en el de los autómatas finitos no se han encontrado algoritmos de características similares.

En resumidas cuentas, parece que estamos todavía lejos de encontrar algoritmos no ya que permitan resolver el problema de la minimalización, si no que tan siquiera consigan aproximaciones razonables con una cierta garantía. Dejamos aquí esta cuestión y pasamos a estudiar otras formas de abordar el problema de la obtención de autómatas finitos "pequeños".

4.3. Irreducibilidad.

4.3.1. Autómatas finitos irreducibles.

Una de las propiedades encaminadas a reducir el tamaño de los autómatas finitos más estudiadas es la irreducibilidad respecto a la fusión de estados. Como ya hemos visto, un autómata finito es irreducible si ninguno de sus autómatas cocientes es equivalente a él o, lo que es lo mismo, si no contiene dos estados cuya fusión de lugar a un autómata finito equivalente al original.

En el caso de los autómatas finitos deterministas la irreducibilidad es equivalente a la minimalidad, por lo que todo autómata finito determinista con un número de estados mayor que el índice de la congruencia por la derecha de Nerode asociada al lenguaje aceptado contiene estados equivalentes que pueden ser mezclados preservando el lenguaje reconocido. La aplicación de esta misma idea a los autómatas finitos ha dado lugar a diferentes sistemas para la reducción del número de estados ([27],[21],[22],[5],[20]), pero dado que la fusión puede realizarse en diferentes órdenes conduciendo a resultados diferentes (ver figura 13) y que la congruencia de Nerode no es suficiente para detectar todos los pares de estados fusionables (ver figura 14), ninguno de los algoritmos eficientes existentes garantiza la obtención de un autómata finito irreducible. De hecho no se conocen algoritmos eficientes que permitan decidir si dos estados dados de un autómata finito son o no fusionables.

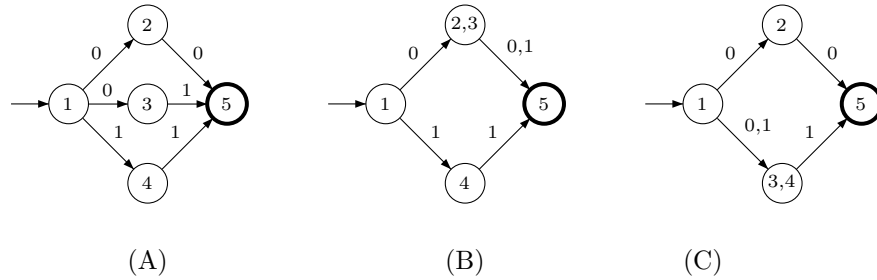


Figura 13: (A) Autómata finito reducible. (B) y (C) Diferentes autómatas cocientes de (A) equivalentes a él e irreducibles.

Una de las primeras preguntas que uno puede plantearse a la hora de abordar este problema es si está o no acotado de alguna manera, esto es, si dado un lenguaje regular existen autómatas finitos irreducibles que lo aceptan con cualquier número de estados. Câmpeanu, Sântean y Yu han planteado recientemente este problema en los siguientes términos:

”Sea L un lenguaje regular arbitrario y $k \geq 2$ un entero arbitrario. ¿Existe (y si la respuesta es ”sí”, construir) una constante $E_{L,k}$ tal que cualquier autómata finito para L de tamaño mayor o igual a $E_{L,k}$ contiene al menos k estados fusionables?”

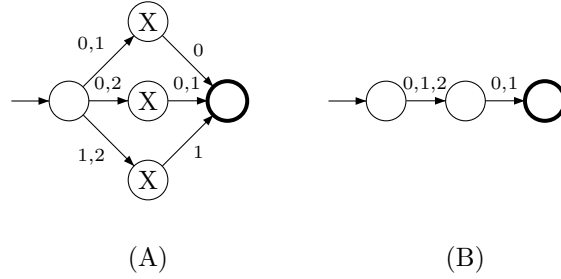


Figura 14: Los nodos marcados del autómata (A) pueden mezclarse para obtener el autómata equivalente (B), pero dado que cada estado tiene lenguajes por la derecha y por la izquierda diferentes a los de los demás, la fusión no es detectada por los métodos basados en la congruencia de Nerode.

Los mismos autores proponen en [8] una solución, basada en la definición, para cada estado del autómata, de dos relaciones de equivalencia, relacionada una de ellas con la congruencia por la derecha de Nerode y la otra con la congruencia sintáctica, como consecuencia de la cual obtienen la cota $E_{L,k} = (k-1)P(H_L)N_L P(N_L) + 1$ donde N_L es el índice de la congruencia por la derecha de Nerode (tamaño del autómata finito determinista mínimo que reconoce L), H_L es el índice de la congruencia sintáctica y $P(n)$, el llamado número de Bell, es el número de todas las posibles particiones del conjunto $\{1, \dots, n\}$. Dado que H_L está acotado por $N_L^{N_L}$ y que $P(n) = \sum_{j=1}^n S_n^j$, siendo S_n^j el número de Stirling de segunda clase, que tiene como valor el número de particiones diferentes de un conjunto de n elementos en j subconjuntos disjuntos no vacíos, tenemos que la cota encontrada crece en forma similar a $N_L^{N_L^{N_L}}$.

Esta cota parece demasiado grande, y los autores del trabajo citado proponen como problema abierto la obtención de una más ajustada. En este capítulo, demostramos la existencia de una cota sustancialmente menor, que además es minimal, utilizando el concepto de autómata universal de un lenguaje L y las propiedades que convierten a éste en una representación canónica de L . Lo mas sustancial de esta demostración puede encontrarse en [16].

4.3.2. Estados fusionables.

Definición 32 Sea Q un conjunto y sea P un subconjunto del mismo. Llamaremos π_P a la partición de Q en la que P es un bloque y cada uno de los demás bloques está formado por un único elemento.

Definición 33 En un autómata finito $A = \langle Q, \Sigma, \delta, I, F \rangle$, un conjunto P de estados es fusionable si $L(A/\pi_P) = L(A)$.

Equivalentemente, los estados de P son fusionables si el autómata finito con transiciones vacías resultante de añadir a δ una de tales transiciones desde cada

estado de P a cada uno de los demás estados de P es equivalente al autómata original.

Teorema 34 Sea $A = \langle Q, \Sigma, \delta, I, F \rangle$ un autómata finito y sea P un subconjunto de Q . Entonces $L(A/\pi_P) = L(A) \cup \left(\bigcup_{q \in P} L_q \right) \left(\bigcup_{p, q \in P} I_{pq} \right)^* \left(\bigcup_{q \in P} R_q \right)$.

Demostración. Demostraremos primero que

$$L(A/\pi_P) \subseteq L(A) \cup \left(\bigcup_{q \in P} L_q \right) \left(\bigcup_{p, q \in P} I_{pq} \right)^* \left(\bigcup_{q \in P} R_q \right) :$$

Si $x \in L(A/\pi_P)$ existe un camino en A/π_P etiquetado con los símbolos de x . Si ese camino no pasa por el estado P , ese mismo camino existe para x en A y por tanto $x \in L(A)$. En caso contrario existe una factorización $x = x_1 \dots x_n$ tal que el camino en cuestión pasa por el estado P exclusivamente después de procesar cada segmento x_i con $1 \leq i < n$. Entonces $x_1 \in \bigcup_{q \in P} L_q, x_n \in \bigcup_{q \in P} R_q$, y para $1 < i < n, x_i \in \bigcup_{p, q \in P} I_{pq}$ por lo que $x \in \left(\bigcup_{q \in P} L_q \right) \left(\bigcup_{p, q \in P} I_{pq} \right)^* \left(\bigcup_{q \in P} R_q \right)$.

Haremos ahora la demostración de la inclusión inversa:

Primero, si $x \in L(A)$ entonces $x \in L(A/\pi_P)$. Ahora, si $x \in \left(\bigcup_{q \in P} L_q \right) \left(\bigcup_{p, q \in P} I_{pq} \right)^* \left(\bigcup_{q \in P} R_q \right)$ existe una factorización $x = x_1 \dots x_n$ tal que $x_1 \in L_{q_1}, x_n \in R_{q_n}$ y para $1 < i < n, x_i \in I_{p_i q_i}$, donde los p_i y los q_i pertenecen a P . De ahí se deduce trivialmente la existencia de un camino de aceptación para x en A/π_P que pasa por el estado P exclusivamente tras procesar cada segmento x_i para $1 \leq i < n$. ■

Corolario 35 Los estados del conjunto P son fusionables si y solo si $\left(\bigcup_{q \in P} L_q \right) \left(\bigcup_{p, q \in P} I_{pq} \right)^* \left(\bigcup_{q \in P} R_q \right) \subseteq L(A)$.

Demostración. Se desprende trivialmente del teorema anterior. ■

Dado que la verificación de la anterior desigualdad no es realizable mediante algoritmos eficientes, la mayoría de heurísticos para la reducción mediante fusión del número de estados se basan en condiciones suficientes aunque no necesarias similares a la enunciada en el siguiente lema:

Lema 36 Sea $A = \langle Q, \Sigma, \delta, I, F \rangle$ un autómata finito con transiciones vacías y sean q_1 y q_2 dos estados de Q . Si q_1 y q_2 tienen el mismo lenguaje por la derecha entonces son fusionables.

Demostración. Demostraremos en primer lugar que si $R_{q_1} = R_{q_2}$ entonces

$$\begin{aligned} R_{q_1} &= (I_{q_1} \cup I_{q_2} \cup I_{q_1 q_2} \cup I_{q_2 q_1})^* R_{q_1} : \\ R_{q_1} &= \\ R_{q_1} \cup R_{q_2} &= \\ I_{q_1} R_{q_1} \cup I_{q_1 q_2} R_{q_2} \cup I_{q_2} R_{q_2} \cup I_{q_2 q_1} R_{q_1} &= \\ (I_{q_1} \cup I_{q_2} \cup I_{q_1 q_2} \cup I_{q_2 q_1}) R_{q_1} &= \end{aligned}$$

$$\begin{aligned}
& (I_{q_1} \cup I_{q_2} \cup I_{q_1 q_2} \cup I_{q_2 q_1}) R_{q_1} \cup R_{q_1} = \\
& (I_{q_1} \cup I_{q_2} \cup I_{q_1 q_2} \cup I_{q_2 q_1})^2 R_{q_1} \cup (I_{q_1} \cup I_{q_2} \cup I_{q_1 q_2} \cup I_{q_2 q_1}) R_{q_1} \cup R_{q_1} = \\
& (I_{q_1} \cup I_{q_2} \cup I_{q_1 q_2} \cup I_{q_2 q_1})^n R_{q_1} \cup \dots \cup (I_{q_1} \cup I_{q_2} \cup I_{q_1 q_2} \cup I_{q_2 q_1}) R_{q_1} \cup R_{q_1} = \\
\text{(por inducción)} & \\
& (I_{q_1} \cup I_{q_2} \cup I_{q_1 q_2} \cup I_{q_2 q_1})^* R_{q_1}. \\
\text{Entonces} & \\
& (L_{q_1} \cup L_{q_2}) (I_{q_1} \cup I_{q_2} \cup I_{q_1 q_2} \cup I_{q_2 q_1})^* (R_{q_1} \cup R_{q_2}) = \\
& (L_{q_1} \cup L_{q_2}) (I_{q_1} \cup I_{q_2} \cup I_{q_1 q_2} \cup I_{q_2 q_1})^* R_{q_1} = \\
& (L_{q_1} \cup L_{q_2}) R_{q_1} = \\
& L_{q_1} R_{q_1} \cup L_{q_2} R_{q_1} = \\
& L_{q_1} R_{q_1} \cup L_{q_2} R_{q_2} \subseteq \\
& L(A). \blacksquare
\end{aligned}$$

Este lema nos proporciona una idea intuitiva del camino a seguir para acotar el tamaño de los autómatas finitos sin estados fusionables:

Sea $A = \langle Q, \Sigma, \delta, I, F \rangle$ un autómata finito. Sean q_1 y q_2 dos estados tales que las clases en la congruencia por la derecha de Nerode de las palabras que permiten acceder a ellos desde los estados iniciales son exactamente las mismas. Sea R el lenguaje regular formado por todas aquellas palabras que son un buen final en $L(A)$ para todas esas clases. Sea A_R un autómata finito que acepta R . Si ampliamos A añadiéndole A_R y enviamos transiciones vacías desde q_1 y q_2 hasta el estado inicial de R , que ahora no será ya inicial, el autómata finito con transiciones vacías así obtenido será equivalente a A . Además el lenguaje por la derecha de q_1 y q_2 en el nuevo autómata es en ambos casos R , por lo que si fusionamos ambos estados el autómata obtenido será también equivalente a A . Si, por último eliminamos de ese autómata el subautómata A_R obtenemos como resultado $A/\pi_{\{q_1, q_2\}}$ que evidentemente será equivalente a A . Véase un ejemplo en la figura 15.

En consecuencia, si dos estados de un autómata finito tienen lenguajes por la izquierda formados por palabras (no necesariamente todas, ni las mismas) de las mismas clases en la congruencia por la derecha de Nerode, esos estados son fusionables. Siendo N_L el índice de esa congruencia, si un autómata finito para L tiene $2^{N_L} + 1$ o más estados contendrá necesariamente dos estados fusionables, por lo que el tamaño de un autómata finito para L sin estados fusionables queda acotado por 2^{N_L} .

En el siguiente apartado damos una demostración más formal de este resultado.

4.3.3. Estados fusionables y el autómata universal.

Proposición 37 [1]. *Sea U el autómata universal de un lenguaje regular L y sea $A = \langle Q, \Sigma, \delta, I, F \rangle$ un autómata finito que acepta un subconjunto cualquiera de L . La función φ que asigna a cada estado q de A el estado de U $\varphi(q) = \bigcap_{x \in L_q^A} x^{-1}L$ es un homomorfismo de A en U .*

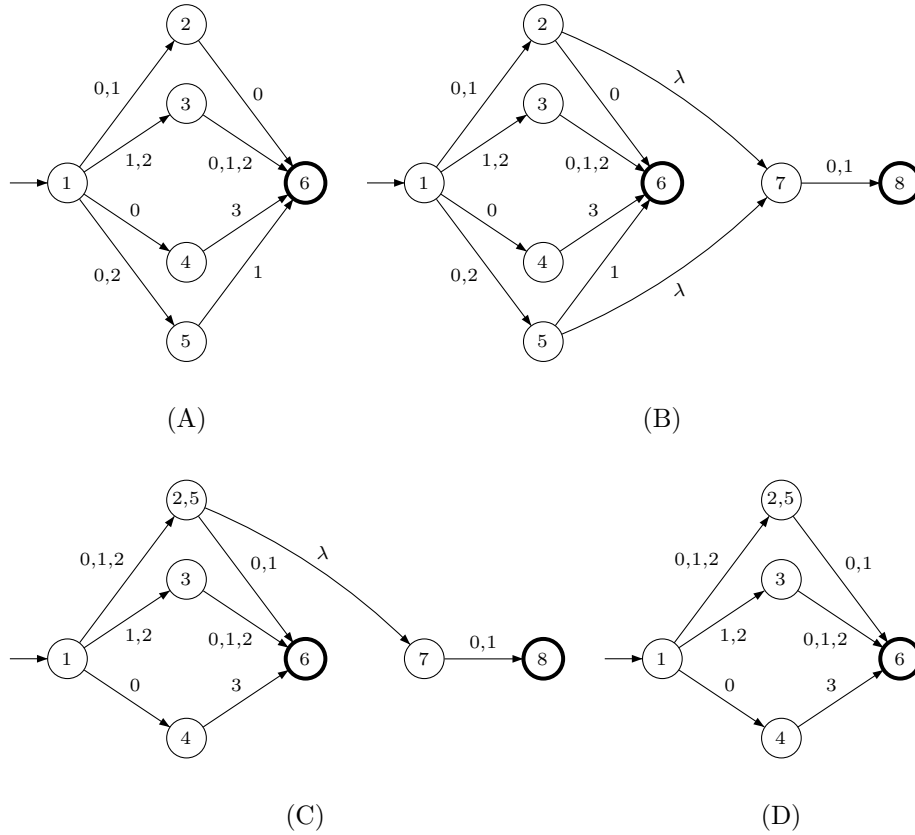


Figura 15: (A) Autómata finito cuyo lenguaje genera las clases $\{\lambda\}, \{0\}, \{1, 2\}, \{00, 01, 03, 10, 11, 12, 20, 21, 22\}$, y otra formada por las demás palabras. A los estados 2 y 5 llegan palabras de las mismas clases, la $\{0\}$, y la $\{1, 2\}$ cuyos buenos finales comunes forman el lenguaje $\{0, 1\}$. (B) Agregamos el autómata para el lenguaje $\{0, 1\}$ y dirigimos a su estado inicial transiciones vacías desde los estados 2 y 5. Esto no varía el lenguaje reconocido. (C) Ahora los estados 2 y 5 tienen el mismo lenguaje por la derecha, por lo que podemos fusionarlos sin alterar el lenguaje reconocido. (D) Si eliminamos el subautómata agregado obtenemos un autómata cociente del original y equivalente al mismo.

Llamaremos homomorfismo canónico a ese homomorfismo.

Comentario 38 Sea $U = (U, \Sigma, \delta, I, F)$ el autómata universal de un lenguaje regular L . Para cada estado q de U sea $\{x_1, \dots, x_n\}$ un conjunto maximal de palabras tal que para $i, j \in \{1, \dots, n\}$, si $i \neq j$, entonces $x_i^{-1}L \neq x_j^{-1}L$ y $q = \bigcap_{i=1}^n x_i^{-1}L$. Se puede apreciar fácilmente que cada palabra de $[x_i]_{\equiv_L}$ alcanza el estado $x_i^{-1}L$ y, por la definición de la función de transición de U , alcanza cualquier estado de U que sea un subconjunto de $x_i^{-1}L$. Por tanto

$$L_q^U = \bigcup_{I=1}^n [x_i]_{\equiv_L}.$$

Lema 39 Sea A un autómata finito para el lenguaje L y sea U el autómata universal de L . Sean p y q estados de A y sea φ el homomorfismo canónico de A en U . Entonces:

1. $L_q^A \subseteq L_{\varphi(q)}^U$
2. $R_q^A \subseteq \varphi(q) \subseteq R_{\varphi(q)}^U$
3. $I_{pq}^A \subseteq I_{\varphi(q)\varphi(q)}^U$

Demostración. La primera afirmación se deduce directamente del comentario anterior. La segunda proviene del hecho de que si un estado q es alcanzado por palabras pertenecientes a diferentes clases en \equiv_L , su lenguaje por la derecha debe estar incluido en la intersección de los lenguajes residuales de esas clases. Por último, la tercera se sigue del hecho de que φ es un homomorfismo entre autómatas. ■

Lema 40 Sea A un autómata finito para el lenguaje L y sea U el autómata universal de L . Sea φ el homomorfismo canónico de A en U . Si existen k estados de A , q_1, \dots, q_k tales que $\varphi(q_1) = \dots = \varphi(q_k)$ entonces los estados q_1, \dots, q_k son fusionables.

Demostración. Por el lema anterior

$$\left(\bigcup_{i=1}^k L_{q_i}^A \right) \left(\bigcup_{i,j=1}^k I_{q_i, q_j}^A \right)^* \left(\bigcup_{i=1}^k R_{q_i}^A \right) \subseteq L_{\varphi(q_1)}^U \left(I_{\varphi(q_1)}^U \right)^* R_{\varphi(q_1)}^U \subseteq L_{\varphi(q_1)}^U I_{\varphi(q_1)}^U R_{\varphi(q_1)}^U \subseteq L$$

■

Teorema 41 Sea $A = \langle Q, \Sigma, \delta, I, F \rangle$ un autómata finito para el lenguaje L . Si $|Q| > (k-1)2^{N_L}$ el autómata A contiene al menos k estados fusionables.

Demostración. El número de estados del autómata universal está acotado por 2^{N_L} . Por el principio de la caja de Dirichlet, si $|Q| > (k-1)2^{N_L}$, al menos k tendrán la misma imagen bajo φ . ■

La cota calculada no puede ser reducida. Para demostrarlo, dado que el autómata universal de un lenguaje regular no contiene estados fusionables, es suficiente con definir para cada alfabeto Σ una familia de autómatas finitos deterministas mínimos tales que las intersecciones de sus lenguajes residuales son todas diferentes entre sí y cuyos autómatas universales tienen, por tanto, 2^{N_L} estados. Para $n \geq 2$ considérese el autómata $A(n) = (\mathbb{Z}_n, \Sigma, \delta, 0, \mathbb{Z}_n - \{0\})$ con función de transición definida como sigue:

Para cada estado i en \mathbb{Z}_n y cada símbolo a , $\delta(i, a) = (i+1) \bmod n$. Véase como ejemplo el autómata $A(3)$ y su autómata universal en la figura 16.

Demostraremos ahora un resultado adicional:

Teorema 42 Sea $A = \langle Q, \Sigma, \delta, I, F \rangle$ un autómata finito para el lenguaje L y sea U el autómata universal de L . Los estados q_1 y q_2 de A son fusionables si y solo si existe algún homomorfismo ψ de A en U tal que $\psi(q_1) = \psi(q_2)$.

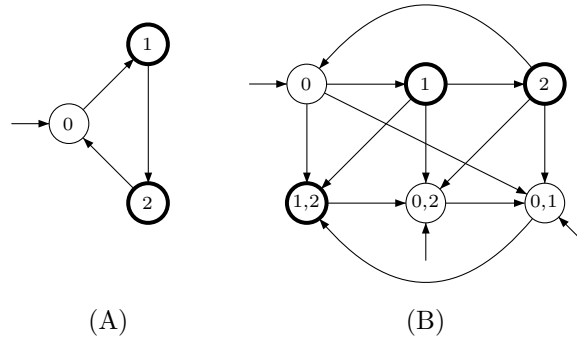


Figura 16: (A) Autómata finito determinista mínimo para el lenguaje formado por aquellas palabras cuya longitud no es múltiplo de tres. (B) Autómata universal para el mismo lenguaje en el que se han suprimido los estados inútiles correspondientes a la intersección de todos los estados del mínimo y a la de ninguno.

Demostración. En primer lugar, si existe el homomorfismo ψ entonces $L(A/\pi_{\{q_1, q_2\}}) \subseteq L(\psi(A)) \subseteq L(U) \subseteq L(A)$ y por tanto q_1 y q_2 son fusionables. Y, en segundo lugar, si q_1 y q_2 son fusionables sea φ el homomorfismo canónico de $A/\pi_{\{q_1, q_2\}}$ en U . Entonces la función ψ de Q en U tal que a cada estado p de $Q - \{q_1, q_2\}$ le asigna $\varphi(\pi(p))$ y $\psi(q_1) = \psi(q_2) = \varphi(\pi(q_1))$ es un homomorfismo de A en U tal que $\psi(q_1) = \psi(q_2)$. ■

En términos de la congruencia por la derecha de Nerode el teorema anterior significa que los estados q_1 y q_2 de A son fusionables si y solo si existe un conjunto finito de palabras $\{x_i\}$ tal que $L_{q_1} \cup L_{q_2} \subseteq \bigcup_{x_i \in \{x_i\}} [x_i]_{\equiv_{L(A)}}$ y $R_{q_1} \cup R_{q_2} \subseteq$

$$\bigcap_{x_i \in \{x_i\}} x_i^{-1} L(A).$$

Corolario 43 Sea $A = \langle Q, \Sigma, \delta, I, F \rangle$ un autómata finito para el lenguaje L y sea U el autómata universal de L . A es irreducible si y solo si todo homomorfismo de A en U es inyectivo.

Demostración. Se desprende trivialmente del teorema anterior. ■

4.3.4. Irreducibilidad relativa.

En ciertos campos de la teoría de lenguajes, tales como la inferencia inductiva o la representación de lenguajes finitos, podemos encontrar el problema de calcular un autómata finito lo mas pequeño posible que acepte todas las palabras de un cierto lenguaje finito M_+ y rechace todas las de otro lenguaje finito M_- . La obtención de una solución minimal [15] es un problema $NP - duro$. Si prescindimos de la condición de minimalidad el problema deviene trivial. Una vez mas surge la necesidad de buscar una situación de compromiso entre am-

bos extremos, consistente en exigir al autómata obtenido ciertas propiedades, irreducibilidad, concisión etc. que garanticen que no es innecesariamente grande.

Como veremos más adelante, sí es fácil garantizar que el resultado obtenido mediante un método cualquiera sea cuanto menos un autómata finito irreducible.

Definición 44 *Un autómata finito $A = \langle Q, \Sigma, \delta, I, F \rangle$ es irreducible en el lenguaje L si y sólo si $L(A) \subseteq L$ y para toda partición no trivial π de Q se cumple que $L(A/\pi) - L \neq \emptyset$.*

Teorema 45 *Sean L un lenguaje regular, U el autómata universal de L y $A = \langle Q, \Sigma, \delta, I, F \rangle$ un autómata finito tal que $L(A) \subseteq L$. A es irreducible en L si y sólo si todo homomorfismo de A en U es inyectivo.*

Demostración. Demostraremos primero la condición "sólo si": Si ψ es un homomorfismo no inyectivo de A en U existirán estados q_1 y q_2 de Q tales que $\psi(q_1) = \psi(q_2)$. Entonces $L(A/\pi_{\{q_1, q_2\}}) \subseteq L(\psi(A)) \subseteq L(U) \subseteq L$ y por tanto A no es irreducible en L .

Veamos ahora la condición si: Si A no es irreducible en L existirán estados q_1 y q_2 de Q tales que $L(A/\pi_{\{q_1, q_2\}}) \subseteq L$. Entonces, siendo φ el homomorfismo canónico de $A/\pi_{\{q_1, q_2\}}$ en U , la función ψ de A en U tal que a cada estado p de $Q - \{q_1, q_2\}$ le asigna $\varphi(\pi(p))$ y $\psi(q_1) = \psi(q_2) = \varphi(\pi(q_1))$ es un homomorfismo no inyectivo de A en U . Véase un ejemplo en la figura 17. ■

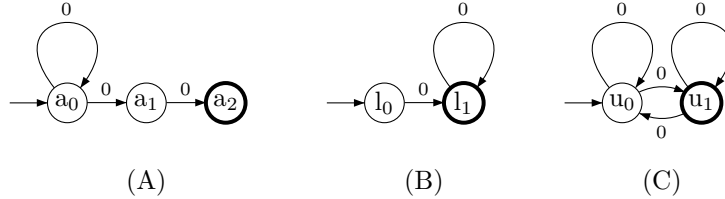


Figura 17: (A) Autómata finito A para el lenguaje 0^*00 . (B) y (C) Autómata finito determinista mínimo y Autómata universal para el lenguaje $L = 0^*0$. A no es irreducible en L pues el homomorfismo $h(a_0) = h(a_1) = u_0, h(a_2) = u_1$ no es inyectivo.

Lema 46 *Un autómata finito A es irreducible si y sólo si existe algún lenguaje L tal que A es irreducible en L .*

Demostración. La implicación \Rightarrow se deduce de que si A es irreducible entonces es irreducible en $L(A)$. Para demostrar la implicación \Leftarrow es suficiente con observar que, puesto que $L(A) \subseteq L$, si $L(A/\pi)$ acepta alguna palabra que no pertenece a L entonces acepta también una palabra (la misma) que no pertenece a $L(A)$. ■

Lema 47 *Un automata finito $A = \langle Q, \Sigma, \delta, I, F \rangle$ es irreducible si y sólo si existe algún lenguaje cofinito L tal que A es irreducible en L .*

Demostración. Por el lema anterior es suficiente con demostrar la implicación \Rightarrow . Dado que el conjunto Q es finito también lo es el conjunto $\{\pi_i\}$ de sus particiones no triviales. Puesto que A es irreducible, para cada partición no trivial π_i existe al menos una palabra que no pertenece a $L(A)$ y sí a $L(A/\pi_i)$. Sea, para cada i , x_i la menor en orden canónico de tales palabras, sea $\{x_i\}$ el lenguaje finito formado por ellas y sea $L = \overline{\{x_i\}}$. Entonces A es irreducible en L . ■

Lema 48 Sea $A = \langle Q, \Sigma, \delta, I, F \rangle$ un autómata finito, sean P_1 y P_2 subconjuntos de Q y sean $q_1 \in P_1$ y $q_2 \in P_2$. Entonces $L(A/\pi_{\{q_i, q_j\}}) \not\subseteq L \Rightarrow L(A/\pi_{P_1 \cup P_2}) \not\subseteq L$.

Demostración. Se desprende trivialmente del hecho de que para toda partición π de Q , $L(A) \subseteq L(A/\pi)$. ■

Teorema 49 El problema de dados un autómata finito $A = \langle Q, \Sigma, \delta, I, F \rangle$ y un lenguaje finito $M_- \subseteq \overline{L(A)}$ encontrar un autómata finito A' de tamaño menor o igual que A e irreducible en $\overline{M_-}$ tal que $L(A) \subseteq L(A')$ es polinómico respecto a $|Q|$ y $\|M_-\|$.

Demostración. Sea $Q = \{q_1, \dots, q_n\}$ y sea $M_- = \{x_1, \dots, x_m\}$. Supongamos que en el autómata $A/\pi_{\{q_j, q_i\}}$ el estado $\{q_j, q_i\}$ recibe como nombre q_j . El algoritmo

```

A' = A
DESDE i = 2 HASTA n
  DESDE j = 1 HASTA i
    SI q_j es un estado de A'
      compatible = CIERTO
    DESDE k = 1 HASTA m
      SI x_k ∈ L(A'/π_{q_j, q_i})
        compatible = FALSO
    SI compatible
      A' = A'/π_{q_j, q_i}

```

computa un autómata A' con las características requeridas. (Véase un ejemplo en la figura 18). ■

Ya hemos visto que un autómata finito determinista $A = \langle Q, \Sigma, \delta, I, F \rangle$ es irreducible si y sólo si es mínimo. Pero puede ocurrir que A no sea irreducible en un lenguaje L y sin embargo no exista ninguna partición π de Q tal que $L(A/\pi) \subseteq L$ y A/π sea un autómata finito determinista. Cabe entonces introducir el concepto de irreducibilidad determinista relativa.

Definición 50 Un autómata finito determinista $A = \langle Q, \Sigma, \delta, I, F \rangle$ es determinísticamente irreducible en un lenguaje L si y sólo si $L(A) \subseteq L$ y no existe ninguna partición π de Q tal que $L(A/\pi) \subseteq L$ y A/π sea un autómata finito determinista.

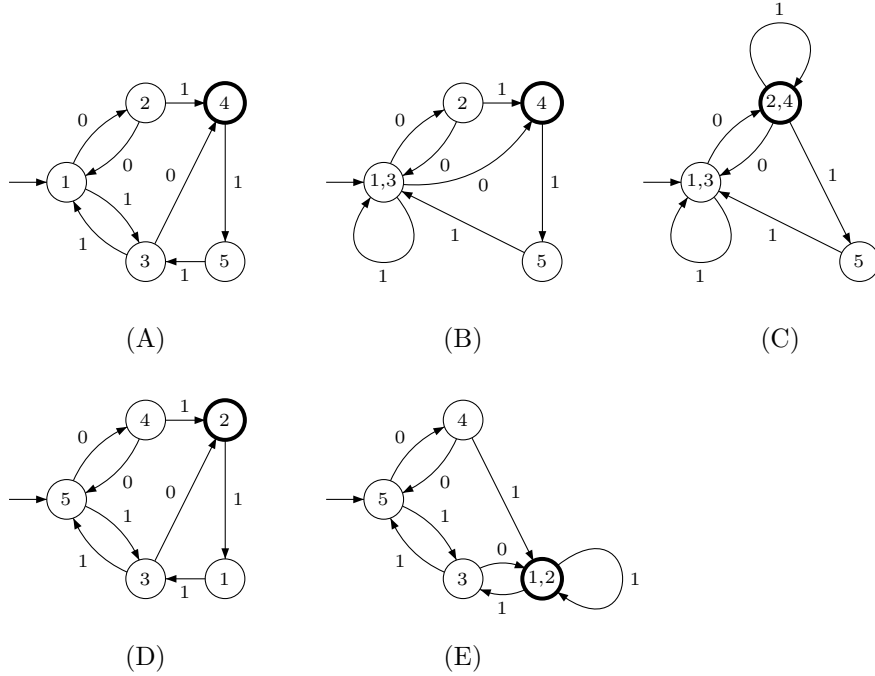


Figura 18: Se pretende obtener un autómata finito a partir del mostrado en (A) irreducible respecto a $M_- = \{001, 010\}$. Se prueba a fusionar primero los estados 1 y 2, pero el autómata obtenido acepta la palabra 001. A continuación se prueba a fusionar 1 y 3 obteniéndose el de la figura (B), compatible con la muestra. Se desecha la mezcla de $\{1, 3\}$ y 4 que acepta 010 y se prueba con 2 y 4 resultando el autómata (C). Dado que 5 no puede mezclarse ni con $\{1, 3\}$ ni con $\{2, 4\}$ (C) es el autómata buscado. (D) y (E) muestran que una ordenación diferente de los nodos produce un resultado diferente. Los intentos de mezcla pueden en realidad hacerse en cualquier orden aleatorio.

Lema 51 *La irreducibilidad determinista relativa en un lenguaje no implica la irreducibilidad en ese mismo lenguaje.*

Demostración. Basta con considerar el autómata $A = \langle \{q_0, q_1, q_2\}, \{a\}, \delta, q_0, \{q_2\} \rangle$ con $\delta(q_i, a) = q_{(i+1) \bmod 3}$, el cual no es irreducible en a^+ pero si es determinísticamente irreducible en el mismo. ■

4.4. Concisión.

4.4.1. Autómatas finitos concisos.

Un autómata finito es conciso si no contiene estados superfluos, esto es, si la eliminación de cualquier estado del autómata junto con las transiciones que acceden a él o parten de él da lugar a un autómata que no es equivalente al original.

Un autómata finito determinista es conciso si y sólo si es aseado. En otras palabras un estado de un autómata finito determinista es superfluo si y sólo si es inútil. Debido a ello la eliminación de los estados superfluos puede realizarse mediante algoritmos lineales. También en el caso de los autómatas finitos en general la eliminación de los estados inútiles puede hacerse linealmente, pero en este caso nos encontramos con que no todos los estados superfluos son inútiles. De hecho, los problemas de dado un autómata finito averiguar si es o no conciso y de dado un estado de un autómata finito averiguar si es o no superfluo son *PSPACE* – completos.

En esta situación nos planteamos la misma cuestión ya resuelta en el caso de los autómatas irreducibles: ¿Dado un lenguaje regular existen autómatas finitos concisos arbitrariamente grandes que lo acepten o, por el contrario, existe una constante asociada al lenguaje tal que todo autómata finito que lo acepte de tamaño mayor a ese valor contiene necesariamente estados superfluos?. Como veremos, en este caso la respuesta es muy distinta a la obtenida en aquel.

4.4.2. Estados superfluos.

Definición 52 Sea $A = \langle Q, \Sigma, \delta, I, F \rangle$ un autómata finito y sea P un subconjunto de Q . Denominaremos A_{-P} al subautómata de A inducido por $Q - P$.

Por lo tanto,

$$A_{-P} = \langle Q - P, \Sigma, \delta - (P \times \Sigma \times Q) - (Q \times \Sigma \times P), I - P, F - P \rangle.$$

Definición 53 Sea $A = \langle Q, \Sigma, \delta, I, F \rangle$ un autómata finito y sea P un subconjunto de Q . Diremos que el subconjunto P es superfluo si $L(A_{-P}) = L(A)$.

Puede ocurrir que los subconjuntos P_1 y P_2 sean superfluos y que sin embargo $P_1 \cup P_2$ no lo sea, pues aunque P_2 sea superfluo en A no tiene por que serlo en A_{-P_1} .

Lema 54 Sea L un lenguaje finito y sea $\|L\|$ la suma de las longitudes de las palabras de L . El autómata finito para L de mayor tamaño sin estados superfluos contiene $\|L\| + |L|$ estados.

Demostración. Sea $MCA(L)$ el autómata finito maximal para L . Este autómata contiene $\|L\| + |L|$ estados, ninguno de los cuales es superfluo. Cada estado corresponde a un par prefijo-sufijo de una palabra de L . Este autómata utiliza un subconjunto de estados en el proceso de cada palabra x de tamaño $|x| + 1$, el máximo posible, y además disjunto al utilizado en el proceso de cualquier otra palabra. Cualquier otro autómata que reconozca L podrá hacerlo utilizando como máximo $\|L\| + |L|$ estados por lo que si tiene más, algunos serán superfluos.

■

Teorema 55 Todo lenguaje regular infinito es aceptado por autómatas finitos arbitrariamente grandes sin estados superfluos.

Demostración. Sea L un lenguaje regular infinito. Entonces existe una palabra x de L que admite una factorización $x = uvw$ tal que $v \neq \lambda$ y $uv^*w \subseteq L$. Sea A_1 un autómata finito sin estados superfluos para $L - uv^*w$. Sea, para cada $n \geq 0$, $A_{2,n}$ el autómata finito determinista mínimo para uv^nw y $A_{3,n}$ el autómata finito determinista mínimo para $uv^*w - \bigcup_{i=0}^n uv^i w$. Cada autómata $A_n = (\bigcup_{i=0}^n A_{2,i}) \cup A_1 \cup A_{3,n}$ es un autómata finito sin estados superfluos para L . Además, siendo Q_n el conjunto de estados de A_n tenemos que $|Q_n| > |Q_{n-1}|$.

■

En la figura 19 puede verse un ejemplo de esta construcción.

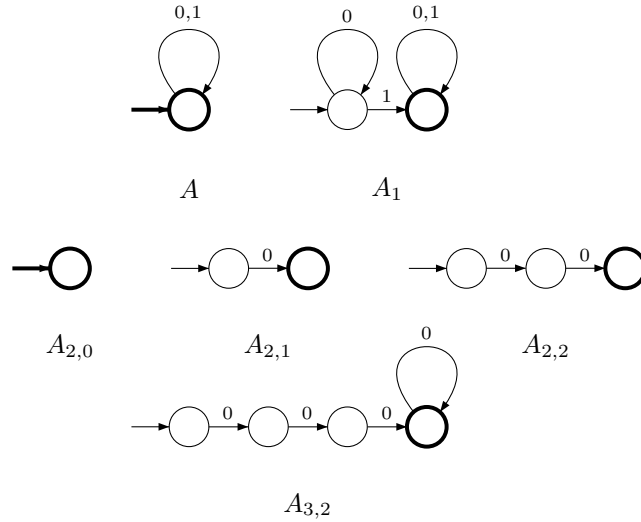


Figura 19: Para obtener autómatas arbitrariamente grandes equivalentes a A seleccionamos una palabra de $L(A)$, 0 en este caso, y la dividimos en tres factores, en el ejemplo $u = \lambda, v = 0, w = \lambda$, de forma que $uv^*w \subseteq L(A)$. Ahora, para cualquier valor de n , 2 en el ejemplo, obtenemos los autómatas $A_1 : L(A_1) = L(A) - uv^*w$, $A_{2,i}, 0 \leq i \leq n$ tales que $L(A_{2,i}) = uv^i w$ y por último $A_{3,n} : L(A_{3,n}) = \{uv^j w, j > n\}$. El autómata resultante de la unión de todos ellos es equivalente a A , no tiene estados superfluos y el número de estados crece estrictamente con el valor de n .

4.4.3. Autómatas finitos concisos irreducibles.

Como hemos visto, el tamaño de un autómata finito irreducible para un lenguaje L está acotado superiormente por el tamaño del autómata universal para ese lenguaje, el cual a su vez está acotado por 2^{N_L} siendo N_L el tamaño de su autómata finito determinista mínimo. Pero en el autómata universal el subconjunto formado por todos los estados excepto los N_L que corresponden a valores del tipo $x^{-1}L$ es un conjunto superfluo. Cabría entonces plantearse la

existencia de una cota polinómica en N_L al tamaño del mayor autómata finito para L conciso e irreducible. Veremos a continuación que no existe esa cota.

Lema 56 *Sea Σ un alfabeto no trivial, esto es, formado por dos o más símbolos. No existe una cota superior polinómica respecto a N_L al tamaño de los autómatas finitos concisos irreducibles que aceptan un lenguaje $L \subseteq \Sigma^*$.*

Demostración. Basta para demostrar el enunciado con encontrar una familia $\{L_n\}_{n \geq 1}$ de lenguajes tal que N_{L_n} crezca polinómicamente con n y una familia $\{A_n\}_{n \geq 1}$ de autómatas finitos concisos e irreducibles tales que $L(A_n) = L_n$ y el número de estados de cada A_n sea función exponencial de n .

Sea $L_n = \Sigma^{n-1}0\Sigma^*$ el lenguaje formado por todas aquellas palabras sobre Σ tales que su enésimo símbolo es 0. Evidentemente $N_{L_n} = n + 2$ aunque si eliminamos el estado superfluo su autómata finito determinista mínimo tendrá $n + 1$ estados. Sea $L_n^R = \Sigma^*0\Sigma^{n-1}$ el reverso de L_n . El autómata finito determinista mínimo para L_n^R tiene 2^n estados, por lo que su reverso es un autómata finito conciso e irreducible para L_n con $2^{N_{L_n}-2}$ estados. ■

Vease en la figura 20 el caso para $\Sigma = \{0, 1\}$ y $n = 2$.

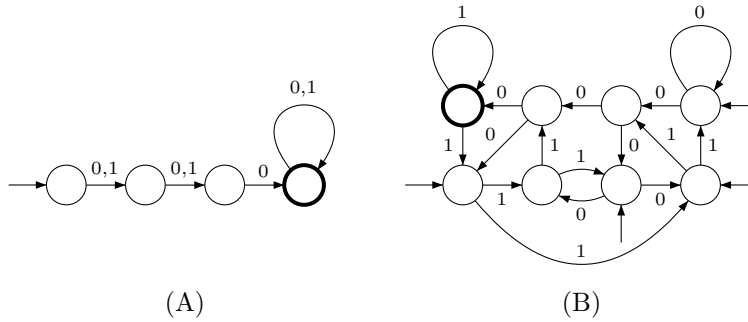


Figura 20: (A) Autómata finito determinista mínimo para $\{0, 1\}^2 0 \{0, 1\}^*$. (B) Un autómata irreducible y conciso para el mismo lenguaje.

4.4.4. Concisión relativa.

Por motivos similares a los expuestos para introducir el concepto de irreducibilidad relativa introducimos ahora el de concisión relativa:

Definición 57 *Un autómata finito $A = \langle Q, \Sigma, \delta, I, F \rangle$ es conciso en el lenguaje L si y solo si $L(A) \supseteq L$ y para todo subconjunto no vacío P de Q se cumple que $L - L(A_{-P}) \neq \emptyset$.*

Lema 58 *Un automata finito A es conciso si y solo si existe algún lenguaje L tal que A es conciso en L .*

Demostración. La implicación \Rightarrow se deduce de que si A es conciso entonces es conciso en $L(A)$. Para demostrar la implicación \Leftarrow es suficiente con observar que, puesto que $L(A) \supseteq L$, si $L(A_{-P})$ no acepta alguna palabra que pertenece a L entonces tampoco acepta una palabra (la misma) que pertenece a $L(A)$. ■

Lema 59 *Un automata finito $A = \langle Q, \Sigma, \delta, I, F \rangle$ es conciso si y solo si existe algún lenguaje finito L tal que A es conciso en L .*

Demostración. Por el lema anterior es suficiente con demostrar la implicación \Rightarrow : Dado que el conjunto Q es finito también lo es el conjunto $\{P_i\}$ de sus subconjuntos no vacíos. Puesto que A es conciso, para cada subconjunto no vacío P_i existe al menos una palabra que pertenece a $L(A)$ y no a $L(A_{-P_i})$. Sea, para cada i , x_i la menor en orden canónico de tales palabras, sea $L = \{x_i\}$ el lenguaje finito formado por ellas. Entonces A es conciso en L . ■

Lema 60 *Sea $A = \langle Q, \Sigma, \delta, I, F \rangle$ un autómata finito, sea P un subconjunto de Q y sea $q \in P$. Entonces $L(A_{-q}) \not\supseteq L \Rightarrow L(A_{-P}) \not\supseteq L$.*

Demostración. Se desprende trivialmente del hecho de que para todo subconjunto P de Q , $L(A) \supseteq L(A_{-P})$. ■

Teorema 61 *El problema de dados un autómata finito $A = \langle Q, \Sigma, \delta, I, F \rangle$ y un lenguaje finito $M_+ \subseteq L(A)$ encontrar un autómata finito A' con un número de estados menor o igual que el de A conciso en M_+ tal que $L(A) \supseteq L(A')$ es polinómico respecto a $|Q|$ y $\|M_+\|$.*

Demostración. Sea $Q = \{q_1, \dots, q_n\}$ y sea $M_+ = \{x_1, \dots, x_m\}$. El algoritmo

```

A' = A
DESDE i = 1 HASTA n
  compatible = CIERTO
  DESDE j = 1 HASTA m
    SI  $x_j \notin L(A'_{-q_i})$ 
      compatible = FALSO
  SI compatible
    A' = A'_{-q_i}

```

computa un autómata A' con las características requeridas. ■

El autómata obtenido dependerá del orden establecido entre los estados de Q . Véase por ejemplo la figura 21.

4.4.5. Irreducibilidad y concisión relativas.

Nos fijaremos ahora en el problema de dados dos lenguajes finitos M_+ y M_- tales que $M_+ \cap M_- = \emptyset$ encontrar un autómata finito irreducible en $\overline{M_-}$ y conciso en M_+ .

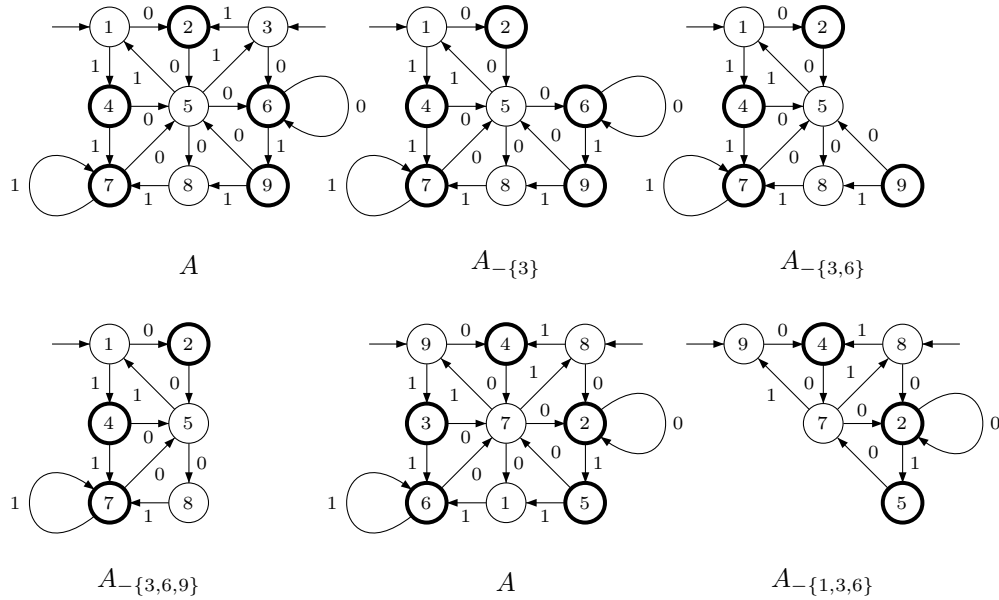


Figura 21: Se pretende obtener un autómata finito a partir de A conciso respecto a $M_+ = \{0, 0010, 1001, 1010\}$. Se prueba a eliminar primero los estados 1 y 2, pero los autómatas obtenidos no aceptan la palabra 0010. A continuación se prueba a eliminar 3 obteniéndose $A_{-\{3\}}$, compatible con la muestra. Se deshecha la eliminación de 4 y la de 5 ya que en ambos casos no se aceptaría 1001 y se prueba con 6 resultando el autómata $A_{-\{3,6\}}$, compatible. Se deshecha la eliminación de 7 y la de 8 por no aceptar 1001 y por último se elimina el 9 obteniéndose $A_{-\{3,6,9\}}$, autómata conciso en M_+ . Una ordenación diferente de los nodos produce un resultado diferente.

Notación 62 A lo largo de este apartado, denominaremos $Irreducible(A, L)$ y $Conciso(A, L)$ a cualquier autómata obtenido a partir de A utilizando los algoritmos descritos anteriormente para la obtención de autómatas irreducibles en L y concisos respecto a L respectivamente.

Lema 63 Sean M_+ y M_- dos lenguajes finitos tales que $M_+ \cap M_- = \emptyset$. Sea $A = \langle Q, \Sigma, \delta, I, F \rangle$ un autómata finito tal que $M_+ \subseteq L(A) \subseteq \overline{M_-}$. Entonces $M_+ \subseteq L(Irreducible(A, \overline{M_-}))$ y $L(Conciso(A, M_+)) \subseteq \overline{M_-}$.

Demostración. Se deduce trivialmente del hecho de que $L(A) \subseteq L(Irreducible(A, \overline{M_-}))$ y $L(Conciso(A, M_+)) \subseteq L(A)$. ■

Lema 64 Sean M_+ y M_- dos lenguajes finitos tales que $M_+ \cap M_- = \emptyset$. Sea $A = \langle Q, \Sigma, \delta, I, F \rangle$ un autómata finito irreducible en $\overline{M_-}$ y tal que $M_+ \subseteq L(A)$. El autómata $Conciso(A, M_+)$ no es necesariamente irreducible en $\overline{M_-}$. En otras

palabras $Irreducible(A, \overline{M_-}) = A \not\Rightarrow Irreducible(Conciso(A, M_+), \overline{M_-}) = Conciso(A, M_+)$.

Demostración. Sean por ejemplo $M_+ = \{0010\}$, $M_- = \{\lambda, 1, 10, 110\}$ y $A = \langle \{1, 2, 3, 4\}, \{0, 1\}, \delta, \{1\}, \{2\} \rangle$ con $\delta = \{\langle 1, 0, 2 \rangle, \langle 1, 1, 4 \rangle, \langle 2, 0, 3 \rangle, \langle 3, 1, 1 \rangle, \langle 4, 0, 3 \rangle\}$. Se puede comprobar sencillamente que se cumplen las condiciones expuestas en el enunciado del lema. Sin embargo $A' = Conciso(A, M_+) = \langle \{1, 2, 3\}, \{0, 1\}, \delta', \{1\}, \{2\} \rangle$ con $\delta' = \{\langle 1, 0, 2 \rangle, \langle 2, 0, 3 \rangle, \langle 3, 1, 1 \rangle\}$ no es irreducible en $\overline{M_-}$ pues los estados 2 y 3 pueden fusionarse dando lugar al autómata $A'' = A'/\pi_{\{2,3\}} = Irreducible(Conciso(A, M_+), \overline{M_-}) = \langle \{1, 2\}, \{0, 1\}, \delta'', \{1\}, \{2\} \rangle$ con $\delta'' = \{\langle 1, 0, 2 \rangle, \langle 2, 0, 2 \rangle, \langle 2, 1, 1 \rangle\}$. Para mas claridad véase la figura 22. ■

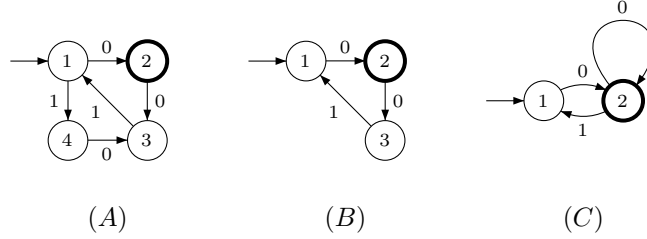


Figura 22: (A) Este autómata es irreducible en $\overline{M_-}$ siendo $M_- = \{\lambda, 1, 10, 110\}$, pero no es conciso respecto a $M_+ = \{0010\}$. Aplicando el algoritmo obtenemos (B) que es conciso respecto a M_+ pero no irreducible en $\overline{M_-}$ ya que podemos unir los estados 2 y 3 obteniendo (C) que es irreducible en $\overline{M_-}$ y conciso respecto a M_+ .

Lema 65 Sean M_+ y M_- dos lenguajes finitos tales que $M_+ \cap M_- = \emptyset$. Sea $A = \langle Q, \Sigma, \delta, I, F \rangle$ un autómata finito conciso en M_+ y tal que $L(A) \subseteq \overline{M_-}$. El autómata $Irreducible(A, \overline{M_-})$ no es necesariamente conciso en M_+ . En otras palabras $Conciso(A, M_+) = A \not\Rightarrow Conciso(Irreducible(A, \overline{M_-}), M_+) = Irreducible(A, \overline{M_-})$.

Demostración. Sean $M_+ = \{2, 01, 10, 20, 21\}$, $M_- = \{\lambda, 0, 00, 11, 22, 200\}$, $A = \langle \{1, 2, 3, 4, 5, 6\}, \{0, 1, 2\}, \delta, \{1\}, \{2, 6\} \rangle$ con $\delta = \{\langle 1, 2, 2 \rangle, \langle 1, 1, 3 \rangle, \langle 1, 0, 4 \rangle, \langle 1, 2, 4 \rangle, \langle 1, 2, 5 \rangle, \langle 3, 0, 6 \rangle, \langle 4, 1, 6 \rangle, \langle 5, 0, 6 \rangle, \langle 5, 1, 6 \rangle\}$. Entonces $L(A)$ es conciso en M_+ y está incluido en $\overline{M_-}$. Sea $\langle Q \rangle = \langle 1, 2, 3, 4, 5, 6 \rangle$. Entonces $Irreducible(A, \langle Q \rangle, \overline{M_-}) = A' = \langle \{1, 2, 4, 5, 6\}, \{0, 1, 2\}, \delta', \{1\}, \{2, 6\} \rangle$ con $\delta' = \{\langle 1, 1, 2 \rangle, \langle 1, 2, 2 \rangle, \langle 1, 0, 4 \rangle, \langle 1, 2, 4 \rangle, \langle 1, 2, 5 \rangle, \langle 2, 0, 6 \rangle, \langle 4, 1, 6 \rangle, \langle 5, 0, 6 \rangle, \langle 5, 1, 6 \rangle\}$ que no es conciso en M_+ ya que el estado 5 es superfluo. Véase la figura 23. ■

Teorema 66 El problema de dados dos lenguajes finitos M_+ y M_- tales que $M_+ \cap M_- = \emptyset$ encontrar un autómata finito A conciso en M_+ e irreducible en $\overline{M_-}$ es polinómico respecto a $\|M_+\|$ y $\|M_-\|$.

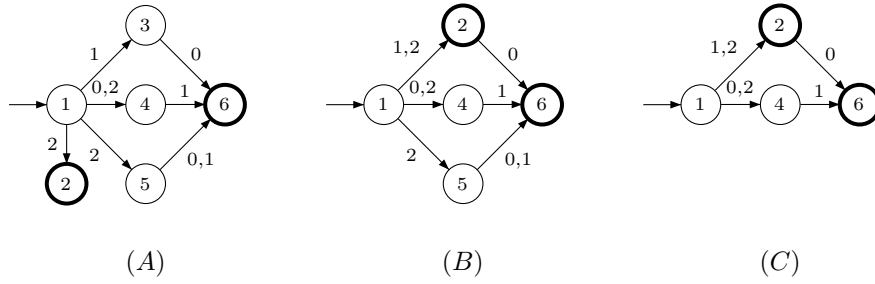


Figura 23: (A) Este autómata es conciso respecto a $M_+ = \{2, 01, 10, 20, 21\}$ pero no es irreducible en $\overline{M_-}$ siendo $M_- = \{\lambda, 0, 00, 11, 22, 200\}$, ya que podemos unir los estados 2 y 3 obteniendo (B) que es irreducible en $\overline{M_-}$ pero no conciso respecto a M_+ ya que podemos eliminar el estado 5 obteniendo (C) que es irreducible en $\overline{M_-}$ y conciso respecto a M_+ .

Demostración. Cualquier algoritmo de la forma

$$A = ACM(M_+)$$

REPETIR

$$A' = A$$

$$A = Conciso(Irreducible(A, \overline{M_-}), M_+)$$

HASTA $A = A'$

resuelve el problema y es polinómico respecto a $\|M_+\|$ y $\|M_-\|$ ya que lo son los algoritmos *Conciso* e *Irreducible* y dado que o bien $A' = A$ o bien A tiene estrictamente menos estados que A' , el bucle *REPETIR* se ejecutará como mucho $\|M_+\| + |M_+|$ veces. ■

5. Redes de autómatas cocientes.

Este concepto fue introducido en [12] con el propósito de establecer un marco que permita considerar el problema de la inferencia de lenguajes regulares como una búsqueda dentro de una red booleana construída a partir de un subconjunto finito del lenguaje a aprender. En el mismo artículo los autores enuncian además algunas propiedades de ese espacio de búsqueda.

En este capítulo generalizamos en diversas maneras el concepto de red de autómatas finitos y profundizamos en el estudio de sus propiedades, lo que nos permitirá enunciar algunos teoremas básicos sobre la inferibilidad mediante diferentes tipos de algoritmos de la clase de los lenguajes regulares.

5.1. Red de cocientes de un autómata finito.

Sea $A = \langle Q, \Sigma, \delta, I, F \rangle$ un autómata finito. Como se establece en [12], el conjunto de autómatas cocientes de A está parcialmente ordenado por la relación \ll tal que $A/\pi_1 \ll A/\pi_2$ si y sólo si existe una partición π del conjunto de estados de A/π_1 tal que $(A/\pi_1)/\pi = A/\pi_2$, esto es, si A/π_2 es un autómata cociente de A/π_1 . El par $\langle A, \ll \rangle$ induce además una red, llamada $Lat(A)$, cuyo cero es A y cuyo uno es el autómata finito determinista mínimo para Σ^* .

Es evidente, por la definición de autómata cociente, que $A/\pi_1 \ll A/\pi_2 \Rightarrow L(A/\pi_1) \subseteq L(A/\pi_2)$. La implicación contraria, afirmada en [12] debido quizás a una errata, no es sin embargo cierta. Para demostrarlo considérese el autómata $A = \langle \{0, 1, 2\}, \{a\}, \delta, \{0\}, \{2\} \rangle$ con $\delta = \{ \langle 0, a, 1 \rangle, \langle 1, a, 2 \rangle \}$. Evidentemente $L(A/\pi_{\{0,1\}}) \subseteq L(A/\pi_{\{1,2\}})$ y sin embargo no es cierto que $A/\pi_{\{0,1\}} \ll A/\pi_{\{1,2\}}$ (Véase la figura 24).

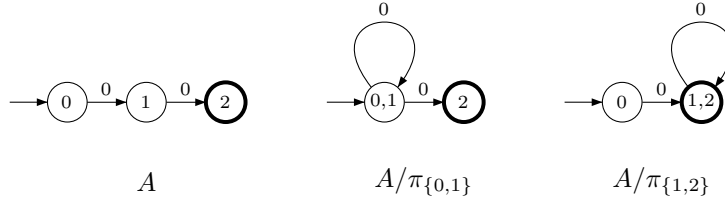


Figura 24: Evidentemente, $L(A/\pi_{\{0,1\}}) \subseteq L(A/\pi_{\{1,2\}})$, sin embargo no es cierto que $A/\pi_{\{0,1\}} \ll A/\pi_{\{1,2\}}$.

La afirmación anterior puede no obstante sustituirse por la siguiente, mas débil:

Lema 67 $L \subseteq L' \Rightarrow \exists A, A' : L(A) = L, L(A') = L', A \ll A'$.

Demostración. Sean $A'' = \langle Q'', \Sigma, \delta'', I'', F'' \rangle$ y $A' = \langle Q', \Sigma, \delta', I', F' \rangle$ los autómatas finitos deterministas completos mínimos de L y L' respectivamente. Sea $A = \langle Q, \Sigma, \delta, I, F \rangle$ tal que $Q = Q'' \times Q', I = I'' \times I', F = F'' \times F'$, y $\delta(\langle q'', q' \rangle, a) = \delta''(q'', a) \times \delta'(q', a)$. Entonces evidentemente $L(A) = L$ y además $A' = A/\pi$ siendo $\pi(\langle q'', q' \rangle) = \{ \langle q'', q' \rangle, q \in Q'' \}$. ■

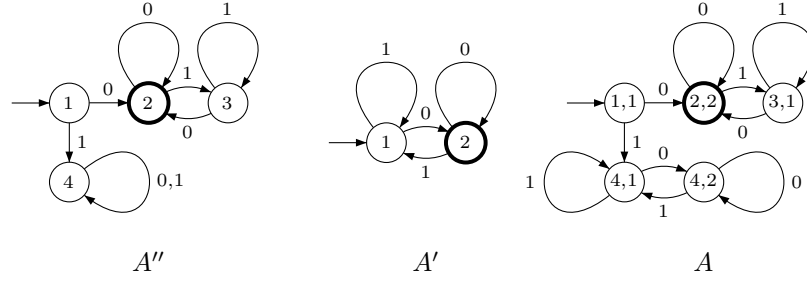


Figura 25: Siendo $L = 0((0 + 1)^*0 + \lambda)$ y $L' = (0 + 1)^*0$, A'' y A' son sus respectivos autómatas finitos deterministas mínimos. Puesto que $L \subseteq L'$ construimos a partir de ellos el autómata A que reconoce a L y del cual se puede obtener como cociente A' mediante la partición $\pi = \{\{(1, 1), (3, 1), (4, 1)\}, \{(2, 2), (4, 2)\}\}$.

Véase un ejemplo en la figura 25.

Siguiendo la notación de [12] diremos que $A/\pi_1 \preceq A/\pi_2$ si existen dos estados q_1 y q_2 de A/π_1 tales que $A/\pi_2 = (A/\pi_1)/\pi_{\{q_1, q_2\}}$. \ll es así la clausura reflexiva y transitiva de \preceq . La profundidad de un autómata A' en la red $Lat(A)$ es n si existen autómatas A_0, \dots, A_n tales que $A_0 = A, A_n = A'$ y para cada entero i , $0 \leq i < n$, $A_i \preceq A_{i+1}$. Puesto que $A_i \preceq A_{i+1}$ implica que el número de estados de A_{i+1} es igual al número de estados de A_i menos uno, se cumple que la profundidad de A' es igual al número de estados de A menos el número de estados de A' .

Lema 68 $A \ll A' \Rightarrow Lat(A') \subseteq Lat(A)$.

Demostración. Trivial. ■

Teorema 69 $A' \in Lat(A)$ si y solo si existe un homomorfismo $h : A \rightarrow A'$ tal que $A' = h(A)$.

Demostración. Sean $A = \langle Q, \Sigma, \delta, I, F \rangle$ y $A' = \langle Q', \Sigma, \delta', I', F' \rangle$. Para demostrar la parte "si" basta con considerar, siendo $A' = A/\pi$, la función h_π que a cada estado q de A le hace corresponder el estado $\pi(q)$ de A' . h_π es evidentemente un homomorfismo y la imagen de A es A' .

Para demostrar la parte "sólo si" consideramos, siendo $A' = h(A)$, la partición π tal que para cada estado $\pi(q) = \{q' : h(q') = h(q)\}$. Dado que en $h(A)$ el estado q' es final o inicial si y sólo si existe q final o inicial respectivamente tal que $q' = h(q)$ y $q'' \in \delta(q', a)$ si y sólo si existen $p', p'' \in Q$ tales que $q'' \in \delta(q', a), h(p') = q', h(p'') = q''$ tenemos que $A' = A/\pi$. ■

Para mas claridad véase un ejemplo en la figura 26.

5.2. Completitud y universalidad estructural.

A continuación exponemos una definición y un teorema extraídos de [12] y adaptados a la definición de autómata finito con la que nosotros trabajamos.

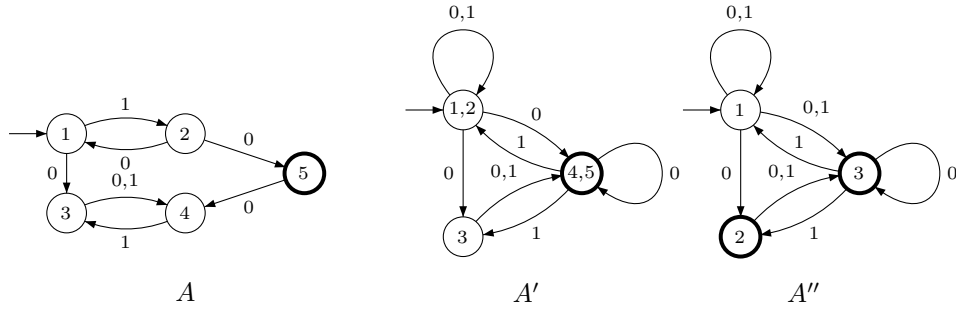


Figura 26: $A \ll A'$ puesto que existe el homomorfismo $h'(1) = h'(2) = \{1, 2\}$, $h'(3) = \{3\}$, $h'(4) = h'(5) = \{4, 5\}$, y $h'(A) = A'$. Sin embargo no es cierto que $A \ll A''$ ya que el único homomorfismo de A en A'' es $h''(1) = h''(2) = 1$, $h''(3) = 2$, $h''(4) = h''(5) = 3$, y $h''(A) \neq A''$.

Definición 70 Un lenguaje finito L es estructuralmente completo respecto a un autómata finito $A = \langle Q, \Sigma, \delta, I, F \rangle$ si $L \subseteq L(A)$ y existe una aceptación de L por A tal que utiliza cada transición de A , cada estado de I es utilizado como estado inicial y cada estado de F es utilizado como estado final.

Teorema 71 Sea L un lenguaje finito. El autómata $A = \langle Q, \Sigma, \delta, I, F \rangle$ pertenece a $\text{Lat}(MCA(L))$ si y solo si L es estructuralmente completo respecto a A .

Demostración. Demostraremos en primer lugar la condición "si". Basta para ello con considerar el homomorfismo de $MCA(L)$ en A tal que a cada estado $\langle x, y \rangle$ de $MCA(L)$ le hace corresponder el estado de A accedido tras procesar x en la aceptación de xy .

Simétricamente, para demostrar la condición "solo si" partimos del hecho de que si A pertenece a $\text{Lat}(MCA(L))$ entonces existe un homomorfismo $h : MCA(L) \rightarrow A$ tal que $A = h(MCA(L))$. La aceptación de L por A tal que en el proceso de cada palabra xy el estado accedido tras procesar x es $h(\langle x, y \rangle)$ cumple las condiciones suficientes para afirmar que L es estructuralmente completo respecto a A . ■

Intuitivamente podría parecer que para todo autómata finito ha de existir algún lenguaje finito estructuralmente completo respecto a él, y de hecho no sabemos que nadie haya mencionado lo contrario. Sin embargo, aunque eso es cierto para los autómatas finitos deterministas aseados, no lo es para los autómatas finitos en general, ni siquiera para aquellos con un solo estado inicial, irreducibles y que no contienen estados superfluos ni tampoco para los autómatas universales de algunos lenguajes:

Lema 72 Existen autómatas finitos para los que no existe ningún lenguaje estructuralmente completo respecto a ellos.

Demostración. Véase la figura 27. ■

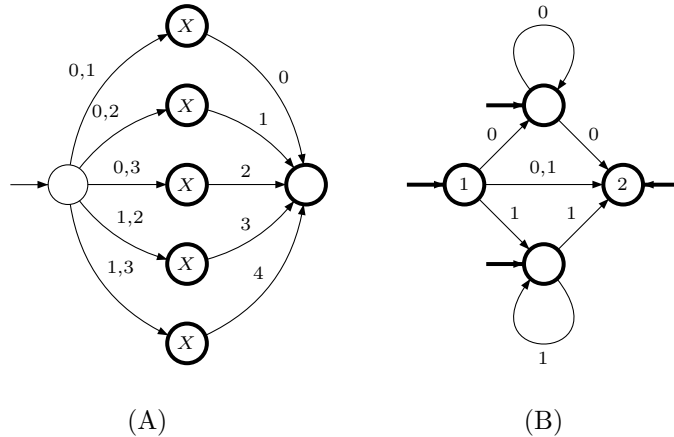


Figura 27: (A) Autómata finito irreducible, sin estados superfluos y con solo un estado inicial. Los estados marcados pueden ser utilizados como finales únicamente en la aceptación de palabras del conjunto $\{0, 1, 2, 3\}$. Dado que este conjunto está formado por cuatro palabras mientras que hay cinco estados marcados, es imposible construir un lenguaje finito en cuya aceptación se utilicen todos ellos como finales. (B) Autómata universal para el lenguaje $0^* + 1^*$. El estado 1 solo puede ser utilizado como final en la aceptación de la palabra λ y el estado 2 solo puede ser utilizado como inicial en la aceptación de esa misma palabra, por lo que no existe ningún lenguaje en cuya aceptación se utilice 1 como final y 2 como inicial.

Sin embargo, evidentemente, para todo autómata finito sin estados inútiles existe un multilenguaje (colección de palabras en la que se admiten repeticiones) finito estructuralmente completo respecto a él. El estudio de la inferencia de lenguajes regulares a partir de multilenguajes y su *SCA* asociado lo dejamos para un trabajo posterior. Veremos en cambio ahora un lema que nos permitirá mantener razonablemente la generalidad sin salirnos del ámbito de los lenguajes.

Definición 73 Diremos que un autómata finito es *L*-inferible si existe algún lenguaje estructuralmente completo respecto a él.

Lema 74 Para todo autómata finito sin estados inútiles ni transiciones superfluas existe otro equivalente *L*-inferible, obtenido a partir de él mediante la eliminación de algunos estados de los conjuntos de estados iniciales y finales.

Demostración. En primer lugar, dado que el autómata no contiene transiciones superfluas, para cada transición existe al menos una palabra cuya aceptación pasa inevitablemente por ella. Y en segundo lugar, si no existe una palabra para cuya aceptación sea indispensable que determinado estado sea inicial o final el

estado puede dejar de serlo sin alterar el lenguaje reconocido por el autómata.

■

En estas condiciones, para no interferir en la legibilidad de algunos enunciados, identificaremos cada autómata finito con el o los subautómatas L -inferibles maximales del mismo, excepto cuando esta identificación pueda afectar a la validez general de un enunciado.

Véase por ejemplo la figura 28.

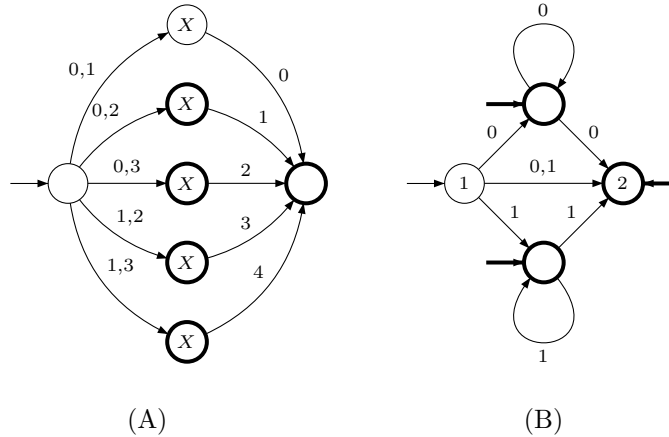


Figura 28: (A) Subautómata L -inferible maximal equivalente al autómata A de la figura anterior. Existen un total de cinco, obtenido cada uno de ellos de la supresión de uno de los estados marcados del conjunto de estados finales. (B) Subautómata L -inferible maximal equivalente al autómata B de la figura anterior. Existen un total de dos, siendo el otro el resultante de eliminar el estado dos del conjunto de estados iniciales.

Así, teniendo presentes estas consideraciones, enunciamos el siguiente teorema:

Teorema 75 *Para todo lenguaje regular L existe un lenguaje finito S tal que todo autómata finito irreducible que acepte a L pertenece a $\text{Lat}(MCA(S))$.*

Demostración. Puesto que cada autómata irreducible para L es un subautómata del autómata universal para L , el conjunto $\{A_i\}_{i \in I}$ formado por todos ellos es un conjunto finito. Sea para cada $i \in I$, S_i un lenguaje estructuralmente completo para A_i . Entonces $S = \bigcup_{i \in I} S_i$ es estructuralmente completo respecto a cada uno de ellos. ■

Véase un ejemplo en la figura 29.

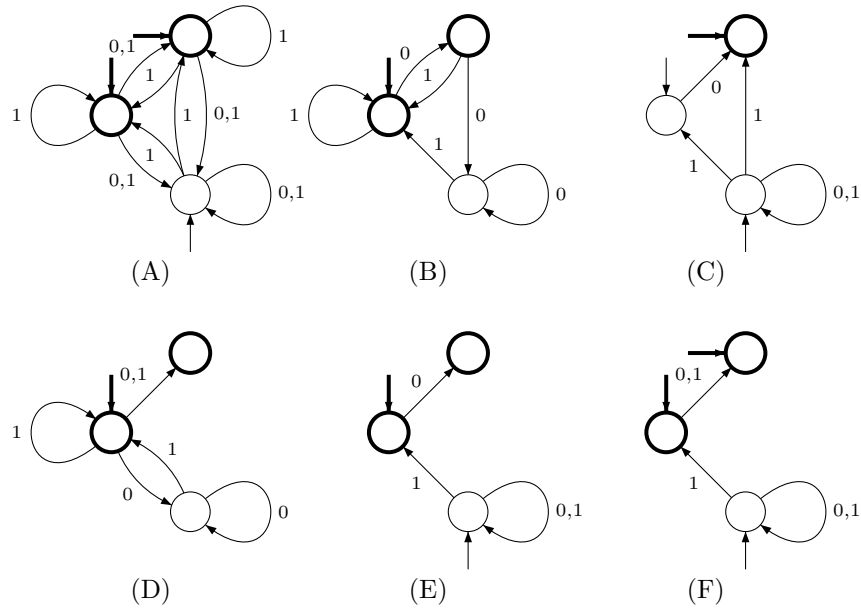


Figura 29: En la figura (A) se muestra el autómata universal U para el lenguaje formado por todas aquellas palabras de ceros y unos que no terminan en dos o más ceros. En las demás figuras se muestran algunos autómatas irreducibles para el mismo lenguaje que son, por tanto, subautómatas de U . Una muestra subestructuralmente completa para U debe ser estructuralmente completa respecto a cada uno de ellos. Nótese que el que la muestra sea estructuralmente completa para U no garantiza que lo sea para cada uno de sus subautómatas. Por ejemplo, $\{10111, 11011, 0110111\}$ es estructuralmente completa para U pero no para los autómatas de las figuras (B), (C), (D), (E) y (F).

Definición 76 Diremos que un lenguaje finito es subestructuralmente completo respecto a un autómata A si lo es respecto a cada uno de sus subautómatas irreducibles equivalentes a él.

Teorema 77 Para todo lenguaje regular L existe un lenguaje finito S tal que todo autómata finito irreducible que acepte a L pertenece a $\text{Lat}(MCA(S))$ y tal que cada autómata irreducible en L que pertenece a $\text{Lat}(MCA(S))$ reconoce L .

Demostración. Sea S_1 un lenguaje finito subestructuralmente completo respecto al autómata universal de L . Sea por otra parte $\{A_i\}_{i \in J}$ el conjunto de subautómatas del autómata universal de L irreducibles en L que reconocen un lenguaje incluido propiamente en L . Este conjunto es evidentemente finito. Para cada uno de estos subautómatas A_i existe al menos una palabra x_i de L que no es aceptada por él. Sea $S_2 = \{x_i\}_{i \in J}$ el lenguaje finito formado por esas palabras. Sea entonces $S = S_1 \cup S_2$.

Por el teorema anterior sabemos que todo autómata finito irreducible que acepte a L pertenece a $\text{Lat}(MCA(S))$.

Sea por otra parte A un autómata de $\text{Lat}(MCA(S))$ irreducible en L . Puesto que $L(A)$ está incluído en L existe un homomorfismo h de A en U , el autómata universal para L . Dado que A es irreducible en L el homomorfismo h es inyectivo y por tanto $h(A)$ es isomorfo a A . A es por tanto un subautómata de U irreducible respecto a L . Como A acepta todas las palabras de S_2 no puede ser uno de los subautómatas irreducibles de U que aceptan un subconjunto propio de L , y por tanto $L(A) = L$. ■

Definición 78 *Llamaremos lenguaje estructuralmente universal de un lenguaje regular L a cualquier subconjunto S de L tal que para cualquier autómata finito A irreducible en L , $A \in \text{Lat}(MCA(S)) \Leftrightarrow L(A) = L$.*

Definición 79 *Llamaremos muestra estructuralmente universal de un lenguaje regular L a cualquier lenguaje finito estructuralmente universal de L .*

Teorema 80 *Para toda muestra estructuralmente universal S de un lenguaje regular L se cumple que todo autómata finito A irreducible en L tal que $S \subseteq L(A)$ reconoce L .*

Demostración. Puesto que A es irreducible en L es isomorfo a un subautómata del autómata universal para L . Por construcción, si A reconociera un subconjunto propio de L habría al menos una palabra en S que no pertenecería a $L(A)$. ■

Teorema 81 *Sea L un lenguaje regular y sean S_1 y S_2 lenguajes finitos tales que $S_1 \subseteq S_2 \subseteq L$. Si S_1 es una muestra estructuralmente universal de L entonces también lo es S_2 .*

Demostración. Trivial. ■

5.2.1. Aplicaciones a la inferencia de L_3 .

Teorema 82 *La clase de los lenguajes regulares es inferible en el límite mediante datos positivos y un oráculo para la inclusión.*

Demostración. Basta con considerar cualquier algoritmo que pruebe a unir en un orden cualquiera pares de estados del MCA de la muestra, conservando como nueva hipótesis el autómata resultante si este es aceptado por el oráculo. El proceso termina cuando ya no se pueden fusionar mas estados tras un número de como máximo $n(n-1)/2$ intentos de fusión siendo n el número de estados del MCA de la muestra. Si la muestra es estructuralmente universal para el lenguaje objetivo se obtendrá como resultado del proceso un autómata finito irreducible que lo reconoce.

Si los datos se van proporcionando secuencialmente basta con, por ejemplo, averiguar si la nueva palabra es aceptada por la hipótesis actual y en caso de

que no lo sea, obtener un nuevo autómata finito añadiendo a aquel el *MCA* del lenguaje formado exclusivamente por la nueva palabra y probando a fusionar cada uno de los nuevos estados con todos los demás del autómata resultante bajo la supervisión del oráculo. El proceso requerirá como máximo $mn + n(n - 1)/2$ intentos de fusión siendo m el número de estados de la hipótesis actual y n la longitud de la nueva palabra mas uno. ■

Véase un ejemplo en la figura 30

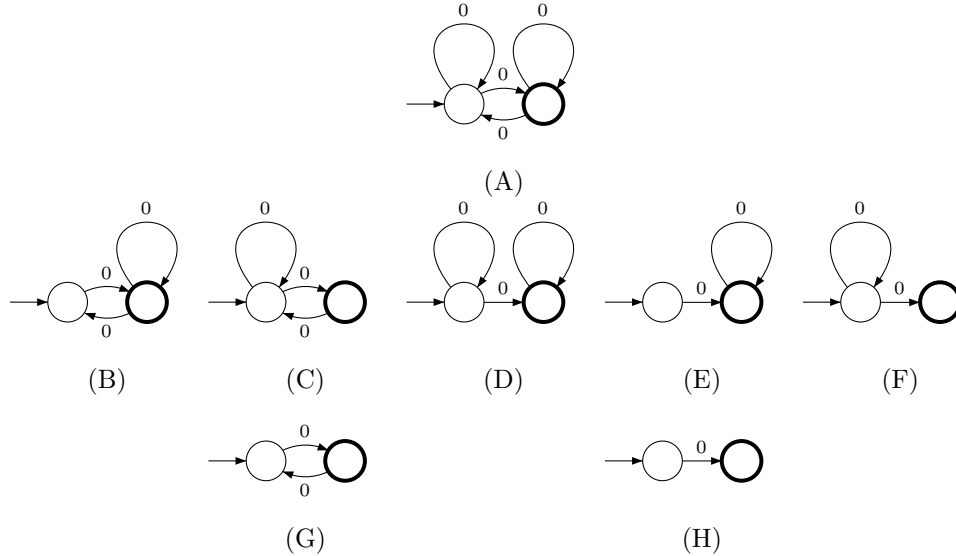


Figura 30: En la figura (A) se muestra el autómata universal \mathbf{U} para el lenguaje 00^* . En las figuras (B)-(F) se muestran todos los autómatas irreducibles para el mismo lenguaje que son, por tanto, subautómatas de \mathbf{U} . La muestra $\{00000\}$ es subestructuralmente completa para \mathbf{U} , pues es estructuralmente completa respecto a cada uno de ellos. En las figuras (G) y (H) tenemos los subautómatas de \mathbf{U} que aceptan un subconjunto propio de 00^* distinto del vacío. La muestra $\{00, 00000\}$ es estructuralmente universal para 00^* , pues no es aceptada por ninguno de estos últimos. Cualquier proceso de mezcla de estados sobre $MCA(\{00, 00000\})$ guiado por un oráculo para la inclusión en 00^* terminará por tanto, cuando no puedan hacerse mas fusiones, en uno de los autómatas (A)-(F).

5.3. Semired de cocientes asociada a un par de autómatas finitos.

Definición 83 *Dados dos autómatas finitos A y A' tales que $L(A) \cap L(A') = \emptyset$ llamaremos semired de cocientes asociada a A y A' al conjunto $SemiLat(A, A')$ formado por aquellos elementos A'' de $Lat(A)$ tales que $L(A'') \cap L(A') = \emptyset$. Por extensión llamaremos $SemiLat(A, L)$ al conjunto formado por aquellos elemen-*

tos A'' de $Lat(A)$ tales que $L(A'') \cap L = \emptyset$.

Puede verse un ejemplo en la figura 31.

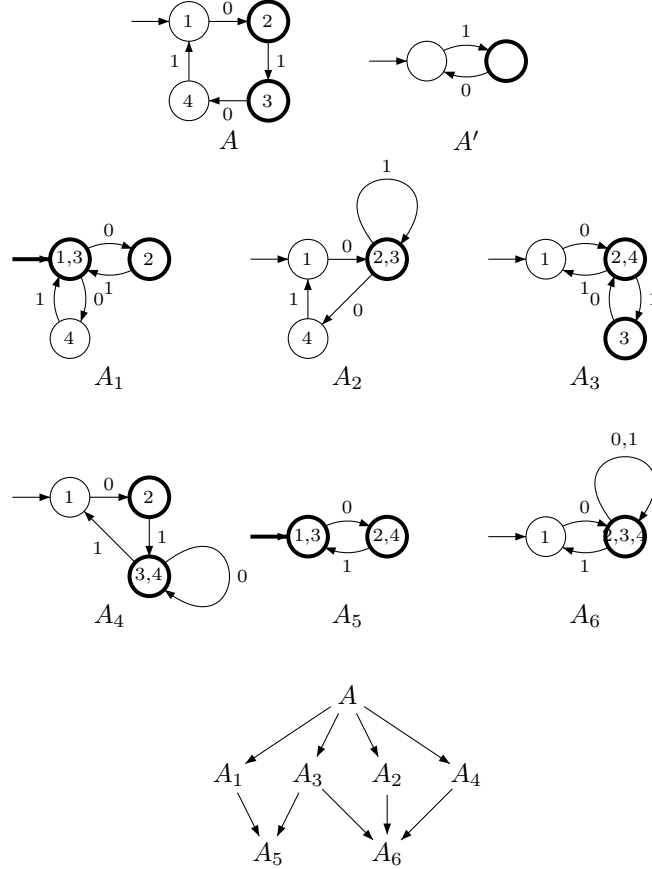


Figura 31: Semired de cocientes asociada a los autómatas A y A' . $BS(A, A') = \{A_5, A_6\}$

Lema 84 $L(A) \cap L(A') = \emptyset$ implica que $SemiLat(A, A')$ forma una semired junto con el orden \ll .

Demostración. Evidentemente $A \in SemiLat(A, A')$ por lo que es suficiente demostrar que si $A_1, A_2 \in Lat(A, A')$, $A_1 \ll A_2$ y $A_2 \in SemiLat(A, A')$ entonces $A_1 \in SemiLat(A, A')$. Esto se deduce trivialmente de que $A_1 \ll A_2$ implica que $L(A_1) \subseteq L(A_2)$ que a su vez implica que $L(A_2) \cap L(A') = \emptyset \Rightarrow L(A_1) \cap L(A') = \emptyset$. ■

Lema 85 $A' \in \text{SemiLat}(A, \overline{L(A)}) \Rightarrow L(A') = L(A)$.

Demostración. $A' \in \text{SemiLat}(A, \overline{L(A)}) \Rightarrow$
 $A \ll A' \wedge L(A') \cap \overline{L(A)} = \emptyset \Rightarrow$
 $L(A) \subseteq L(A') \wedge L(A') \subseteq L(A) \Rightarrow$
 $L(A') = L(A)$. ■

Definición 86 *Dados dos autómatas finitos A y A' tales que $L(A) \cap L(A') = \emptyset$ llamaremos borde de la semired de cocientes asociada a A y A' al conjunto $BS(A, A')$ formado por aquellos elementos A'' de $\text{SemiLat}(A, A')$ tales que no existe $A''' \in \text{SemiLat}(A, A')$ tal que $A''' \neq A''$ y $A'' \ll A'''$. Además, para lenguajes finitos cualesquiera S_1 y S_2 tales que $S_1 \cap S_2 = \emptyset$ llamaremos $BS_{MCA}(S_1, S_2)$, $BS_{PTA}(S_1, S_2)$, etc. a $BS(MCA(S_1), MCA(S_2))$, $BS(PTA(S_1), PTA(S_2))$, etc.*

Lema 87 $A_1 \ll A_2 \Rightarrow \text{SemiLat}(A_2, A') \subseteq \text{SemiLat}(A_1, A')$. $A_1 \ll A_2 \Rightarrow BS(A_2, A') \subseteq BS(A_1, A')$.

Demostración. Trivial. ■

Lema 88 *Todo autómata finito perteneciente a $BS(A, L)$ es un autómata irreducible en \overline{L} .*

Demostración. Se desprende trivialmente de la definición de $BS(A, L)$. ■

Lema 89 *Todo autómata finito perteneciente a $BS(A, \overline{L(A)})$ es un autómata irreducible equivalente a A .*

Demostración. Es equivalente a A por pertenecer a $\text{SemiLat}(A, \overline{L(A)})$ y es irreducible por pertenecer a $BS(A, \overline{L(A)})$. ■

Corolario 90 *A es irreducible si y solo si $\text{SemiLat}(A, \overline{L(A)}) = BS(A, \overline{L(A)}) = \{A\}$.*

Lema 91 *Para todo autómata finito A existe un lenguaje finito L para el que $BS(A, L)$ está formado exclusivamente por autómatas irreducibles equivalentes a A . Concretamente, $BS(A, L) = BS(A, \overline{L(A)})$.*

Demostración. Consideremos el conjunto $\{A_i\}$ formado por aquellos autómatas $A_i \in \text{Lat}(A) - \text{SemiLat}(A, \overline{L(A)})$ tales que existe $A'_i \in \text{SemiLat}(A, \overline{L(A)})$, $A'_i \preceq A_i$, esto es, A_i se obtiene de A'_i mediante la fusión de dos de sus estados. $\{A_i\}$ es un conjunto finito. Además, para cada A_i existe al menos una palabra x_i tal que $x_i \in L(A'_i) = L(A)$, $x_i \notin L(A_i)$. Sea L un lenguaje finito formado por una de estas palabras por cada A_i . Entonces $BS(A, L) = BS(A, \overline{L(A)})$ y por tanto está formado exclusivamente por autómatas irreducibles equivalentes a A . ■

Lema 92 *Para todo autómata finito A y todo lenguaje regular L tal que $L(A) \cap L = \emptyset$ existe un lenguaje finito S tal que $BS(A, L) = BS(A, S)$.*

Demostración. El razonamiento es el mismo que en el lema anterior. ■

Teorema 93 *Para todo lenguaje regular L existen lenguajes finitos S_1 y S_2 tales que $BS_{MCA}(S_1, S_2)$ es el conjunto formado exclusivamente por todos los autómatas finitos irreducibles que reconocen L .*

Demostración. Sea S_1 una muestra estructuralmente universal de L . Entonces para cualquier autómata finito A irreducible en L , $A \in Lat(MCA(S_1)) \Leftrightarrow L(A) = L$. Esto es, en la red $Lat(MCA(S_1))$ están todos los autómatas finitos irreducibles que aceptan L y además esos son los únicos autómatas finitos irreducibles en L que figuran en la red. Por tanto $BS(MCA(S_1), \bar{L})$ es el conjunto formado exclusivamente por todos los autómatas finitos irreducibles que reconocen L . Basta ahora con escojer S_2 de forma que $BS(MCA(S_1), S_2) = BS(MCA(S_1), \bar{L})$. La existencia de tal S_2 está garantizada por el lema anterior. ■

Teorema 94 *Para toda muestra estructuralmente universal S_1 de un lenguaje regular L existe un lenguaje finito S_2 tal que todo autómata finito A irreducible en $\overline{S_2}$ que satisface $S_1 \subseteq L(A)$ reconoce L .*

Demostración. Se desprende directamente de las propiedades de una muestra estructuralmente universal y de los resultados anteriores. ■

Definición 95 *Llamaremos muestra completa universal de un lenguaje regular L a cualquier par de lenguajes finitos S_1 y S_2 tales que $BS_{MCA}(S_1, S_2)$ es el conjunto formado exclusivamente por todos los autómatas finitos irreducibles que reconocen L .*

Veamos a continuación un resultado negativo que en cierta forma limita la caracterización de los lenguajes regulares mediante pares de lenguajes finitos.

Teorema 96 *No es cierto que para cada lenguaje regular L existen lenguajes finitos S_1 y S_2 tales que para lenguajes finitos cualesquiera S'_1, S'_2 , se cumpla que si $S_1 \subseteq S'_1 \subseteq L$ y $S_2 \subseteq S'_2 \subseteq \bar{L}$ entonces necesariamente $BS_{MCA}(S'_1, S'_2)$ contiene exclusivamente autómatas finitos que reconocen L .*

Demostración. Sea por ejemplo $L = (aa)^*$. Supongamos la existencia de los citados S_1 y S_2 y sea a^{2n+1} la palabra de mayor longitud en S_2 . Sea $S'_1 = S_1 \cup \{a^{2n+2}, a^{2n+4}\}$. Entonces podemos unir los estados de $MCA(S_1)$ correspondientes a $\langle a^{2n+2}, \lambda \rangle$ y $\langle a^{2n+3}, a \rangle$ obteniendo así un autómata finito perteneciente a $Lat(MCA(S'_1))$ e incluido en $\overline{S_2}$ que sin embargo acepta la palabra a^{2n+3} que no pertenece a L . Esta palabra será aceptada también por cualquiera de sus cocientes por lo que $BS_{MCA}(S'_1, S_2)$ contendrá al menos un autómata finito que no reconoce L . ■

5.3.1. Aplicaciones a la inferencia de L_3 .

Veremos a continuación que el resultado anterior no plantea restricciones tan fuertes a la inferencia en el límite de \mathcal{L}_3 como pudiera parecer.

Teorema 97 *Dadas una muestra positiva $S_+ = \{x_i\}_{1 \leq i \leq n}$ y otra negativa S_- de un lenguaje, cualquier algoritmo de la forma*

$$A = \langle \emptyset, \Sigma, \emptyset, \emptyset, \emptyset \rangle$$

Desde $i = 1$ Hasta n

Si $x_i \notin L(A)$

$$A = MCA(\{x_1, \dots, x_i\})$$

$A = A/\pi$ para cualquier partición π tal que A/π sea irreducible en $\overline{S_-}$ infiere en el límite \mathcal{L}_3 .

Demostración. Sea $e : \mathbb{N} \rightarrow \Sigma^*$ una biyección. Entonces e induce una enumeración de Σ^* . Sea L un lenguaje regular y sea n_e el menor número tal que las primeras n_e palabras de L en el orden fijado por e forman una muestra estructuralmente universal S_1 de L . Sea S_2 tal que todo autómata finito A irreducible en $\overline{S_2}$ que satisface $S_1 \subseteq L(A)$ reconoce L . Entonces, para cualquier par muestra positiva muestra negativa $\langle S_+, S_- \rangle$ tal que $S_1 \subseteq S_+$ y $S_2 \subseteq S_-$ en que las palabras de S_+ estén ordenadas según e , el anterior algoritmo produce como salida un autómata irreducible para L . Por otra parte, cualquiera que sea el orden que fijemos entre las palabras de nuestra muestra finita, ese orden puede ser considerado como el comienzo de alguna enumeración, y o bien incluye ya una muestra estructuralmente universal, o bien la incluirá eventualmente según se vayan añadiendo a ella nuevas palabras independientemente del orden en que estas se añadan. En el momento en que la muestra positiva es ya estructuralmente universal los datos positivos que se suministren posteriormente no jugarán ningún papel en el proceso, por lo que no afectarán a la muestra negativa necesaria para garantizar la inferencia del lenguaje objetivo. ■

La operación " $A = A/\pi$ para cualquier partición π tal que A/π sea irreducible en $\overline{S_-}$ " requerirá como máximo $n(n-1)/2$ intentos de fusión siendo n el número de estados del $MCA(\{x_1, \dots, x_i\})$.

Comentario 98 *Jugando con cierta imprecisión en el concepto de inferencia en el límite podemos enunciar una proposición aún más general: Supongamos que en el procedimiento anterior los datos positivos se procesan en paquetes de p palabras. El argumento utilizado anteriormente para n palabras sirve igualmente para n paquetes de p palabras. Si nuestra muestra positiva consiste en un número cualquiera p de palabras, el proceso de obtención de un autómata finito consistente e irreducible en el complementario de la muestra negativa mediante fusión de estados en un orden completamente aleatorio, puede ser visto como el primer paso de un algoritmo que en el límite infiere la clase de los lenguajes*

regulares. Después de todo, en cualquier caso práctico siempre dispondremos de un número finito de muestras, y el hecho de que el algoritmo utilizado infiera en el límite la clase \mathcal{L}_3 no garantiza en absoluto que a partir de ellas podamos obtener un autómata para el lenguaje objetivo.

Veamos ahora una versión semiincremental del teorema anterior. Comenzaremos estableciendo el siguiente lema:

Lema 99 *Sea $\langle S_+, S_- \rangle$ una muestra completa universal de un lenguaje L y sea $S_1 \subseteq S_+$. Entonces si $MCA(S_1)/\pi$ es irreducible en $\overline{S_-}$ y $L(MCA(S_1)/\pi) \supseteq S_+$ se cumple que $L(MCA(S_1)/\pi) = L$.*

Demostración. Dado que $\langle S_+, S_- \rangle$ es una muestra completa universal de L , basta con demostrar que $MCA(S_1)/\pi \in Lat(MCA(S_+))$. Para ello es suficiente con observar que podemos obtener $MCA(S_1)/\pi$ a partir de $MCA(S_+)$ de la siguiente manera: Primero construimos $MCA(S_1)/\pi$ utilizando el subautómata $MCA(S_1)$ de $MCA(S_+)$; por último, dado que $L(MCA(S_1)/\pi) \supseteq S_+$, para cada palabra xy de $S_+ - S_1$ existe un camino de aceptación en $MCA(S_1)/\pi$ por lo que podemos unir el estado $\langle x, y \rangle$ de $MCA(S_+)$ con el estado alcanzado en la aceptación de xy tras procesar x , obteniendo como resultado final el autómata $MCA(S_1)/\pi$. ■

Ahora estamos ya en condiciones de demostrar el siguiente teorema:

Teorema 100 *Dadas una muestra positiva $S_+ = \{x_i\}_{1 \leq i \leq n}$ y otra negativa S_- de un lenguaje, cualquier algoritmo de la forma*

$$A = \langle \emptyset, \Sigma, \emptyset, \emptyset, \emptyset \rangle$$

Desde $i = 1$ Hasta n

Si $x_i \notin L(A)$

$$A = A \cup MCA(\{x_i\})$$

$$A = A/\pi \text{ para cualquier partición } \pi \text{ tal que } A/\pi \text{ sea irreducible en } \overline{S_-}$$

infiera en el límite \mathcal{L}_3 .

Demostración. El razonamiento es exactamente el mismo que en el teorema anterior, exceptuando el hecho de que sólo se utilizan en la construcción del autómata aquellas palabras que no son reconocidas por la hipótesis actual. El lema precedente garantiza la convergencia. ■

Véase un ejemplo de la traza de un algoritmo de este tipo en la figura 32.

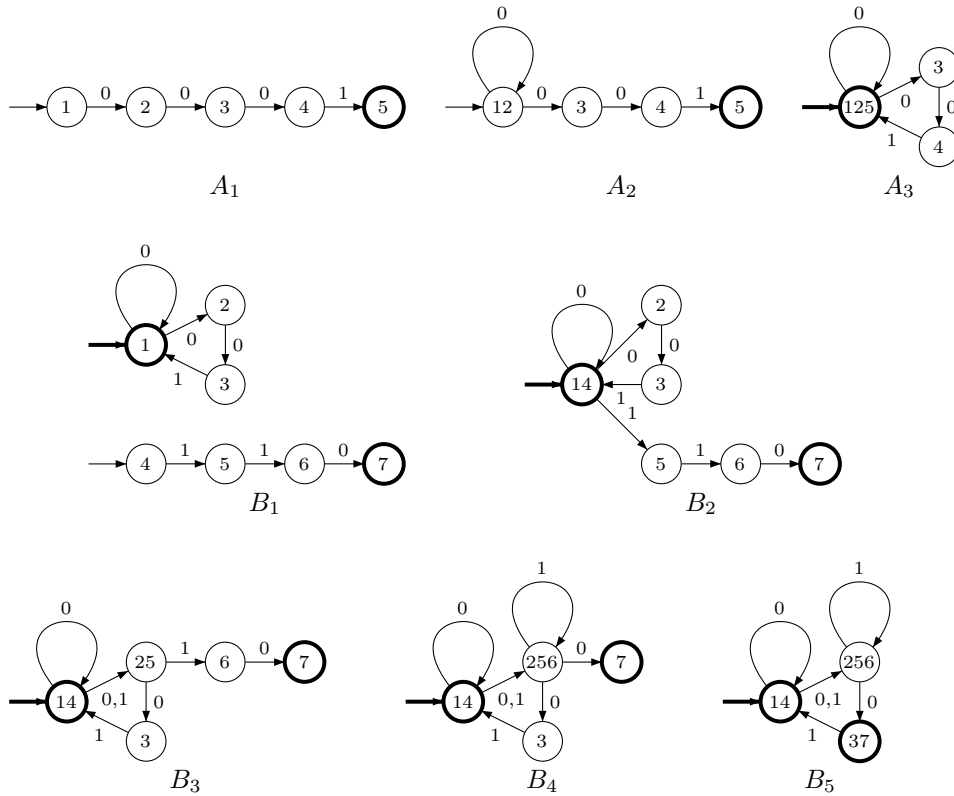


Figura 32: Proceso de inferencia a partir de las muestras $M_+ = \{0001, 001, 00, 0, 110, 11010\}$ y $M_- = \{1, 01, 11, 011, 100\}$. En cada caso se toman los estados en orden y se prueba a unir cada uno de ellos con cada uno de los anteriores, también en orden. A_1 : $MCA(\{0001\})$. A_2 : Unimos el estado 2 con el 1 sin que se produzca incompatibilidad con M_- . Probamos a unir el 3 con el 1 y el 4 con el 1 y con el 3, pero en todos los casos se producen incompatibilidades. A_3 : Unimos el 5 con el 1. Pasamos a la siguiente palabra de la muestra, 001, pero como es aceptada por la hipótesis actual la ignoramos. Lo mismo ocurre con 00 y 0. Llegamos a 110 que no es aceptada. B_1 : Unimos $MCA(\{110\})$ a la hipótesis actual. B_2 : Unimos 4 con 1. B_3 : Tras rechazar la unión de 5 y 1, unimos 5 con 2. B_4 : Tras rechazar la unión de 6 y 1, unimos 6 con 2. B_5 : Tras rechazar la unión de 7 con 1 y con 2, unimos 7 con 3. La última palabra de la muestra es aceptada por la hipótesis actual, por lo que la ignoramos. B_5 es un autómata irreducible compatible con la muestra.

6. Inferencia mediante subautómatas asociados a palabras.

Describimos en esta sección una nueva familia de algoritmos para la inferencia de lenguajes regulares. Cada algoritmo de esta familia, teniendo como entrada una muestra $\langle S_+, S_- \rangle$, calcula para cada palabra x de S_+ al menos un autómata finito que acepta a x y es consistente con S_- . Este autómata es además irreducible en $\overline{S_-}$ y por tanto irreducible. La salida del algoritmo es el autómata formado por la colección de autómatas finitos asociados a cada palabra de S_+ . Cada algoritmo de la familia converge en el límite a un autómata para el lenguaje objetivo.

Presentamos también los experimentos realizados para comparar algunos algoritmos de esta familia con otros algoritmos bien conocidos para la misma tarea. Los resultados obtenidos por nuestros algoritmos son en general mejores, tanto en tasa de error como en el tamaño de la salida. Además, la estructura particular de estos algoritmos resulta en una complejidad menor y por tanto tiempos de ejecución sustancialmente menores a los de los algoritmos clásicos.

6.1. Subautómatas asociados a una palabra en un lenguaje.

Comenzaremos recordando algunos conceptos ya descritos en secciones anteriores que son de especial relevancia en ésta:

Un subautómata de un automata finito $A = \langle Q, \Sigma, \delta, I, F \rangle$ es cualquier autómata finito $A' = \langle Q', \Sigma, \delta', I', F' \rangle$ donde $Q' \subseteq Q$, $I' \subseteq I \cap Q'$, $F' \subseteq F \cap Q'$ y $\delta' \subseteq \delta \cap Q' \times \Sigma \times Q'$. Evidentemente si A' es un subautómata de A , $L(A') \subseteq L(A)$. Evidentemente también, cualquier autómata finito tiene un número finito de subautómatas diferentes. A es irreducible en un lenguaje L si y solo si $L(A) \subseteq L$ y para toda partición π de Q $L(A/\pi) - L \neq \emptyset$. A es irreducible si y solo si es irreducible en $L(A)$.

Recordaremos también un par de proposiciones ya vistas de las que haremos uso en esta sección:

Un autómata finito es irreducible si y sólo si es irreducible en algún lenguaje y, en particular, si y sólo si es irreducible en el complementario de algún lenguaje finito. Por último, todo autómata finito irreducible en un lenguaje L es isomorfo a un subautómata del autómata universal para L .

Pasamos ahora a definir los demás conceptos básicos de esta sección.

Definición 101 Dado un autómata finito A llamaremos descomposición de A a cualquier colección $\langle A_i \rangle_{i \in I}$ de subautómatas de A tal que $L(A) = \bigcup_{i \in I} L(A_i)$.

Definición 102 Dada una palabra $x = a_1 a_2 \dots a_n$, $n \geq 0$, donde los a_i son símbolos, denotaremos mediante A_x al autómata finito determinista mínimo sin estados inútiles para el lenguaje $\{x\}$, esto es, al autómata $\langle Q, \Sigma, \delta, I, F \rangle$ donde, siendo $Q = \{\langle y, z \rangle : yz = x\}$, la función de transición es tal que para $yz = x$, $\delta(\langle y, az \rangle, a) = \{\langle ya, z \rangle\}$ y además $I = \{\langle \lambda, x \rangle\}$, $F = \{\langle x, \lambda \rangle\}$.

Definición. Sea L un lenguaje sobre un alfabeto Σ y sea x una palabra de L . Llamaremos subautómata asociado a x en L a cualquier autómata finito A irreducible en L obtenido a partir de A_x mediante una secuencia cualquiera de fusiones de estados.

Es evidente que para cualquier palabra x y cualquier lenguaje L el número de subautómatas asociados a x en L es finito, así como que para cada autómata A asociado a x en L , $x \in L(A)$. Véase un ejemplo en la figura 33.

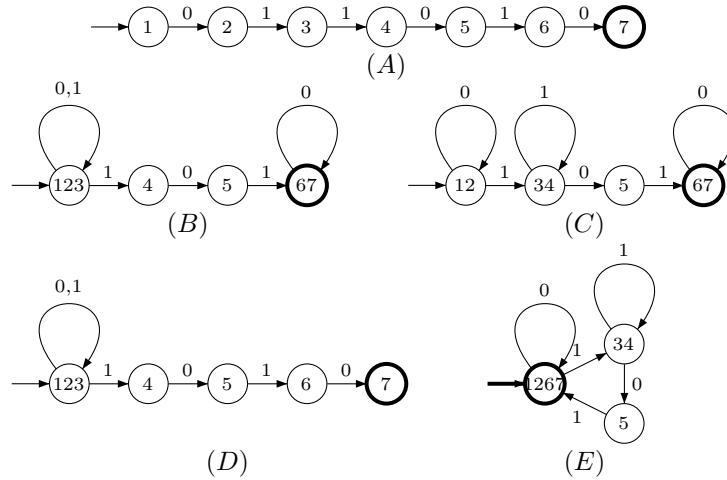


Figura 33: Siendo $L = (0+1)^*101(0+1)^*$ y $x = 011010$ vemos: (A) Autómata A_x . (B) y (C) Subautómatas asociados a x en L . (D) Este autómata no es un subautómata asociado a x en L pues no es irreducible en L . (E) Este autómata tampoco es un subautómata asociado a x en L pues el lenguaje que acepta no está incluido en L .

Lema 103 Sea L un lenguaje regular y sea $x \in L$. Cualquier subautómata asociado a x en L puede obtenerse conociendo un número finito de palabras de \overline{L} .

Demostración. Se deduce directamente del hecho de que un autómata finito es irreducible si y solo si es irreducible en el complementario de algún lenguaje finito. Además, sea $\langle u, v, w \rangle$ una factorización de x , esto es, $x = uvw$. Si los estados de A_x $\langle u, vw \rangle$ y $\langle uv, w \rangle$ no pueden ser fusionados basta con conocer cualquier palabra de $uv^*w - L$ para evitar su fusión. Por tanto el número mínimo de palabras suficiente para garantizar la obtención de un subautómata asociado a x en L está acotado por $|x|^2$. ■

Teorema 104 Para todo lenguaje regular L existe un subconjunto finito M tal que L es la unión de los lenguajes reconocidos por los subautómatas asociados a las palabras de M en L .

Demostración. La existencia del subconjunto M está garantizada por el hecho de que cada palabra x de L es aceptada por cualquier subautómata asociado a x en L . La finitud de M se deduce de que cada subautómata asociado a una palabra x en L es irreducible en L y por tanto isomorfo a algún subautómata de U , el autómata universal de L , junto con el hecho de que el número de subautómatas de U es finito. ■

Véase un ejemplo en la figura 34.

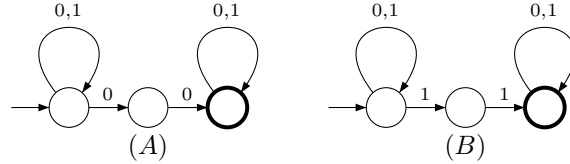


Figura 34: Siendo $L = (0 + 1)^*(00 + 11)(0 + 1)^*$ y $M = \{010001, 011101\}$ se cumple que L es la unión de los lenguajes reconocidos por los subautómatas asociados a las palabras de M en L , ya que entre los asociados a 010001 está (A) y entre los asociados a 011101 está (B).

6.2. Una familia de algoritmos para la inferencia de \mathcal{L}_3 mediante autómatas finitos.

6.2.1. La familia WASRI.

Presentamos ahora una familia (Word Associated Subautomata Regular Inference) de algoritmos basados en los conceptos anteriormente expuestos. Cada uno de los miembros de WASRI infiere en el límite la clase de los lenguajes regulares.

Esta familia se corresponde con el sencillo esquema que describimos a continuación, siendo L el lenguaje a inferir.

Sea $A = \langle Q, \Sigma, \delta, I, F \rangle$ con $Q = \delta = I = F = \emptyset$

Mientras $\exists x \in L - L(A)$

Obtengase al menos un autómata finito asociado a x en L

Para cada $A' = \langle Q', \Sigma, \delta', I', F' \rangle$ obtenido

$$A = \langle Q \cup Q', \Sigma, \delta \cup \delta', I \cup I', F \cup F' \rangle.$$

Las uniones indicadas deben suponerse uniones disjuntas.

Teorema 105 *Cualquier algoritmo de la familia WASRI infiere en el límite la clase de los lenguajes regulares.*

Demostración. Sabemos por una parte que cada lenguaje regular L es la unión de los lenguajes reconocidos por los subautómatas asociados en L a las palabras de un subconjunto finito de L , y por otra, que el número de palabras pertenecientes a \bar{L} que necesitamos conocer para obtener esos autómatas es también finito. Estos dos hechos garantizan la convergencia del proceso, pues después de un tiempo finito esas palabras aparecerán y L será identificado. ■

Los distintos algoritmos de la familia WASRI se diferencian entre sí en conceptos tales como:

- El número de subautómatas inferidos para cada palabra.
- El orden a seguir en el proceso de fusión de estados para la obtención de cada subautómata.
- El criterio de selección final entre los subautómatas inferidos.
- etc.

Veamos a continuación la descripción de uno de los miembros mas básicos de la familia.

6.2.2. WASRI1.

Sea $S = \langle S_+, S_- \rangle$ una muestra de un lenguaje, donde $S_+ = \{x_1, \dots, x_n\}$. El algoritmo WASRI1 obtiene un autómata finito A compatible con S de la siguiente manera:

Sea $A = \langle Q, \Sigma, \delta, I, F \rangle$ con $Q = \delta = I = F = \emptyset$
 Desde $i = 1$ hasta $i = n$
 Si $x_i \in L(A)$
 $A_i = \langle \emptyset, \Sigma, \emptyset, \emptyset, \emptyset \rangle$
 Si $x_i \notin L(A)$
 Sea $A_i = A_{x_i}$ con $Q_i = \{q_{i1}, \dots, q_{im}\}$
 Desde $j = 2$ hasta $j = m$
 Desde $k = 1$ hasta $k = j - 1$
 Si $q_{ik} \in Q_i$
 Si $L(A_i / \pi_{\{q_{ik}, q_{ij}\}}) \cap S_- = \emptyset$
 $A_i = A_i / \pi_{\{q_{ik}, q_{ij}\}}$
 $A = A \cup A_i$
 $A = \langle \emptyset, \Sigma, \emptyset, \emptyset, \emptyset \rangle$
 Desde $i = n$ hasta $i = 1$
 Si $(L(A_i) \cap S_+) - L(A) \neq \emptyset$
 $A = A \cup A_i$

Por $A = A \cup A_i$ queremos significar que siendo $A = \langle Q, \Sigma, \delta, I, F \rangle$ y $A_i = \langle Q_i, \Sigma, \delta_i, I_i, F_i \rangle$ el autómata A pasa a ser $\langle Q \cup Q_i, \Sigma, \delta \cup \delta_i, I \cup I_i, F \cup F_i \rangle$ donde, una vez más, las uniones se presuponen disjuntas.

6.3. Dos ejemplos.

Presentamos en este apartado dos ejemplos de ejecución encaminados a facilitar la comprensión de *WASRI1*. El primero describe la mayoría de situaciones que pueden producirse, esto es, además de la mezcla de estados refleja también el hecho de que algunas palabras de la entrada pueden ser reconocidas por el autómata hipótesis actual y que algunos de los subautómatas obtenidos puede ser eliminado posteriormente si el autómata resultante acepta todas las palabras de la muestra positiva. El segundo ejemplo muestra que este algoritmo puede tratar algunas situaciones de una forma mucho más eficiente que el resto de algoritmos con los que ha sido comparado.

6.3.1. Ejemplo de ejecución.

Supongamos que la entrada a *WASRI1* es $S_+ = \{0, 000011, 001, 0101010\}$ y $S_- = \{01000010\}$.

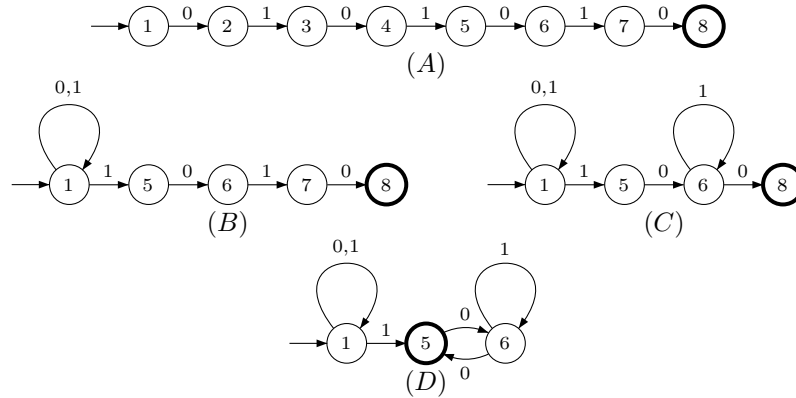


Figura 35: Mezclas sucesivas realizadas por el algoritmo *WASRI1* durante el proceso de la palabra 0101010 siendo su entrada $S_+ = \{0101010, 0, 001, 000011\}$ y $S_- = \{01000010\}$.

Por ser más ilustrativo, comenzaremos por mostrar el proceso de la palabra 0101010 como si fuera la primera de la muestra. El autómata $A_{0101010}$ se muestra en la figura 35(A). Los estados 2,3 y 4 pueden unirse con el estado 1 ya que el autómata resultante no acepta la muestra negativa. El resultado de estas primeras uniones está en la figura 35(B). Siguiendo el mismo orden, los siguientes estados que pueden ser mezclados son 6 y 7, pues las posibles uniones

previas producirían un autómata que acepta la muestra negativa. El autómata resultante es el de la figura 35(C). Por último, la mezcla de los estados 5 y 8 produce el autómata de la figura 35(D), irreducible en $\overline{S_-}$.

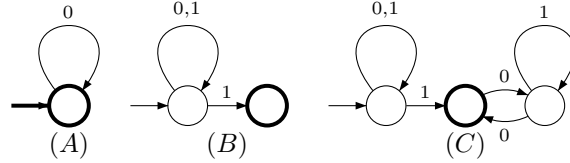


Figura 36: Autómata finito producido por WASRI1 para la entrada $S_+ = \{0101010, 0, 001, 000011\}$ y $S_- = \{01000010\}$ antes del borrado del subautómata superfluo (B).

Consideremos ahora la muestra completa $S_+ = \{0, 000011, 001, 0101010\}$. El autómata obtenido para la primera palabra, 0, se muestra en la figura 36(A). A continuación la palabra 000011 produce como salida el autómata de la figura 36(B). La palabra 001 no produce salida, pues es aceptada por la hipótesis actual, el autómata formado por los subautómatas de la figura 36(A)y(B). A continuación, la palabra 0101010 produce como salida el autómata de la figura 36(C).

Por último, como el autómata resultante del borrado del subautómata correspondiente a la palabra 000011 todavía reconoce todas las palabras de S_+ , el algoritmo *WASRI1* produce como resultado final el autómata formado por los subautómatas (A) y (C) de la figura 36.

6.3.2. Un ejemplo bonito.

Supongamos que $L = \{x \in 0^* : |x| \text{ es múltiplo de } 2, 3 \text{ o } 5\}$ es el lenguaje objetivo. El autómata finito determinista mínimo para L tiene el mismo número de estados que su RFSA canónico (30 estados). Si la entrada para *WASRI1* es $S_+ = \{0^2, 0^3, 0^5\}$ y $S_- = \{0, 0^{11}\}$, produce como salida el autómata descrito en la figura 37, que reconoce L , mientras que algoritmos como *RPNI* o *DeLeTe2* están lejos de la convergencia cuando procesan esa entrada.

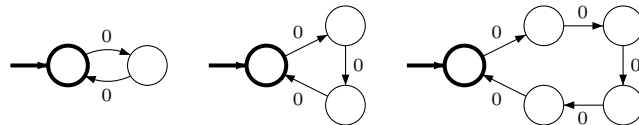


Figura 37: Autómata finito producido por WASRI1 para la entrada $S_+ = \{0^2, 0^3, 0^5\}$ y $S_- = \{0, 0^{11}\}$.

En el capítulo dedicado a la experimentación compararemos el comportamiento de algunos algoritmos de la familia *WASRI* con el de otros algoritmos clásicos entre los que se encuentran aquellos reconocidos como los que mejores resultados ofrecen. Veremos que los resultados obtenidos por *WASRI* son en general mejores tanto en tasa de error como en el tamaño de la salida. Además, el hecho de que *WASRI* procese por separado cada palabra de la muestra positiva disminuye en buena medida su complejidad temporal, lo que le permite procesar en un tiempo razonable entradas que para los demás algoritmos suponen ejecuciones excesivamente largas.

7. Experimentación.

7.1. Las muestras.

El objetivo de esta sección no es proporcionar algoritmos de inferencia refinados para su comparación con otros similares ni, mucho menos, algoritmos diseñados para su aplicación a algún problema en particular. Se persigue únicamente mostrar a grandes rasgos que algoritmos basados en los esquemas descritos en los capítulos anteriores se comportan en el caso general de forma similar, algo mejor en algunos casos y algo peor en otros, que aquellos algoritmos basados en esquemas tradicionales. Utilizaremos como referencia para la comparación dos algoritmos clásicos, el RPNI y el DeLeTe2, que infieren respectivamente autómatas finitos deterministas y RFSAs. Este último ha demostrado comportarse mejor que los demás algoritmos clásicos de inferencia cuando los lenguajes objetivo vienen dados por un proceso de generación pseudoaleatoria de autómatas finitos o expresiones regulares, mientras que el RPNI y sus derivados le superan si la fuente de lenguajes objetivo es un proceso de generación pseudoaleatoria de autómatas finitos deterministas.

Las muestras de aprendizaje y test utilizadas en la primera fase de la experimentación fueron extraídas, y están todavía disponibles, de la página web "<http://www.grappa.univlille3.fr/~lemay/>" de Aurélien Lemay, uno de los autores de DeLeTe2. Están divididas en dos bloques en función de la fuente de las muestras: Las denominadas *er_n* provienen de expresiones regulares mientras que las denominadas *nfa_n* provienen de autómatas finitos con un solo estado inicial. En ambos casos el valor numérico de *n* (50, 100, 150, o 200) indica el número de palabras en la muestra de aprendizaje. Cada uno de estos ocho experimentos consiste en un total de 30 lenguajes diferentes a ser aprendidos y utiliza como test una muestra de mil palabras etiquetadas en función de su pertenencia o no al lenguaje objetivo. El éxito del proceso de aprendizaje en cada uno de los ocho experimentos se mide en función de la tasa media de reconocimiento, esto es la media para los 30 lenguajes de que consta cada experimento de los porcentajes de coincidencia en la clasificación de las palabras de la muestra de test realizada por el autómata obtenido como resultado del mismo.

Sin embargo, estos experimentos presentan ciertas características que no son del todo de nuestro gusto: Tanto las muestras de aprendizaje como las de test contienen palabras repetidas, las muestras de aprendizaje y las de test no son disjuntas entre sí, los autómatas objetivo son diferentes para cada tamaño de las muestras de aprendizaje, por lo que estas no son incrementales, lo cual dificulta la observación de la evolución en el comportamiento del algoritmo de aprendizaje respecto al número de muestras, etc. Decidimos por ello utilizar en una segunda etapa un sistema de experimentación ligeramente diferente: Para cada uno de los 240 lenguajes del experimento anterior (120 procedentes de expresiones regulares y 120 de autómatas finitos) obtuvimos muestras de aprendizaje de tamaños 100, 200, 300, 400 y 500, cada una de las cuales contiene a la anterior, y muestras de test de tamaño 1000 disjuntas de las de aprendizaje.

Denominamos a estas muestras ER_n y NFA_n donde una vez mas ER y NFA indican la procedencia y n su tamaño.

7.2. *WASRI2*.

En la experimentación hemos utilizados dos algoritmos diferentes, uno por cada uno de los esquemas descritos.

El primero de ellos, correspondiente al esquema *WASRI*, al que denominaremos *WASRI2*, obtiene para cada muestra dos autómatas finitos y finalmente elige aquel de los dos que tiene menos estados. El primero de estos autómatas se construye aplicando el esquema *WASRI* en orden canónico a cada palabra que no es aceptada por la hipótesis actual, esto es, para la palabra x_i el prefijo λ corresponde al estado q_{i_0} mientras que al prefijo x_i le corresponde el estado $q_{i_{|x_i|}}$ y se prueba a unir cada estado con cada uno de los anteriores en orden de índice menor a mayor hasta que se encuentra una fusión compatible con la muestra. Por último se eliminan aquellos subautómatas que resultan superfluos para la aceptación de toda la muestra positiva.

El segundo autómata se obtiene en forma similar, salvo que en este caso los estados de cada subautómata se ordenan en orden inverso, de forma que al prefijo λ corresponde al estado $q_{i_{|x_i|}}$ (que pese a su nombre será el inicial) mientras que al prefijo x_i le corresponde el estado q_{i_0} (que será final).

Evidentemente el esquema permitiría la obtención de no solo estos dos autómatas, sino también la de muchos otros mediante cualquier otra ordenación de los estados o la fusión de pares de estados seleccionados de forma aleatoria, etc. En cualquier caso el proceso finalizaría con la selección entre todas las hipótesis obtenidas de aquella con menor tamaño.

7.3. *LPEF*.

El segundo algoritmo, correspondiente al esquema general de fusión de estados, al que nos referiremos como *LPEF* (Lista de Pares de Estados Fusionables) es mas complejo. Al igual que en el caso anterior se construye un subautómata para cada palabra no aceptada por la hipótesis actual. La construcción del subautómata se realiza de la siguiente manera: Para cada par de estados fusionables se calcula el número de pares de estados fusionables que contiene el autómata resultante de su fusión y se efectúa la fusión del par de estados para los que ese valor es maximal. El proceso continúa mientras quedan pares de estados fusionables. Una vez obtenido el subautómata el proceso continúa intentando unir, en orden secuencial, cada estado del nuevo subautómata con cada estado del autómata que constituía la hipótesis anterior. Por último se eliminan los estados superfluos del autómata resultante obteniéndose así la nueva hipótesis.

7.4. Resultados

Los resultados de los experimentos realizados se muestran en la siguiente tabla:

	<i>RPNI</i>		<i>DeLeTe2</i>		<i>WASRI2</i>		<i>LPEF</i>	
	Tasa Rec.	Tam.	Tasa Rec.	Tam.	Tasa Rec.	Tam.	Tasa Rec.	Tam.
<i>er_50</i>	76.4	9.7	81.3	32.4	89.2	15.9	92.0	7.5
<i>er_100</i>	80.6	14.2	91.4	30.7	93.0	25.4	93.5	12.1
<i>er_150</i>	84.5	15.4	92.0	61.0	94.9	25.7	94.5	16.2
<i>er_200</i>	91.1	13.3	95.7	47.7	95.8	35.5	97.4	11.3
<i>nfa_50</i>	64.8	14.3	69.3	71.3	74.8	39.3	71.5	22.2
<i>nfa_100</i>	68.3	21.8	74.4	149.1	76.5	79.8	75.3	42.2
<i>nfa_150</i>	71.2	28.1	76.7	218.3	77.3	121.1	76.7	57.2
<i>nfa_200</i>	71.7	33.4	78.9	271.3	81.2	148.1	78.6	60.4
<i>ER_100</i>	83.4	12.4	91.7	30.2	93.4	25.3	94.4	10.7
<i>ER_200</i>	93.9	11.6	97.0	24.5	95.7	33.4	97.4	11.2
<i>ER_300</i>	96.3	11.2	97.8	31.4	96.2	47.0	98.1	12.7
<i>ER_400</i>	97.5	11.0	98.5	27.4	97.1	51.3	98.8	11.6
<i>ER_500</i>	98.1	11.0	98.8	29.9	97.6	48.6	99.0	11.8
<i>NFA_100</i>	66.5	20.3	74.0	98.8	74.7	82.9	74.3	37.0
<i>NFA_200</i>	69.3	32.4	77.8	220.9	77.1	170.4	77.5	64.7
<i>NFA_300</i>	72.9	40.9	80.9	322.1	79.7	267.2	79.7	90.1
<i>NFA_400</i>	74.6	49.8	82.7	421.3	81.1	350.1	81.3	108.1
<i>NFA_500</i>	76.8	55.9	84.3	512.6	83.5	431.2	83.6	125.8

Las columnas Tasa Rec. indican el tanto por ciento medio de acierto en los diferentes casos que componen cada experimento, mientras que las columnas Tam. indican el valor medio en el número de estados. Puede apreciarse que los resultados obtenidos por *WASRI2* y *LPEF* compiten razonablemente con los de *DeLeTe2* y superan ampliamente a los de *RPNI*, aunque, como ya hemos dicho, esto último no es así cuando los lenguajes objetivo se derivan de la obtención de autómatas finitos deterministas mediante métodos pseudoaleatorios. En cualquier caso, parece razonable esperar que algoritmos mas sofisticados derivados de estos esquemas puedan competir en ciertas tareas con los utilizados actualmente.

8. Conclusión.

Comenzaremos este apartado enumerando los resultados de este trabajo que consideramos mas relevantes y lo finalizaremos describiendo algunos de los problemas que quedan abiertos para futuros trabajos.

Entre los resultados obtenidos destacamos los siguientes:

Se ha introducido el concepto de irreducibilidad relativa, definida de manera que un autómata finito A es irreducible en un lenguaje L si y solo si $L(A) \subseteq L$ y la fusión de dos estados cualesquiera de A da lugar a un autómata A' tal que $L(A') \not\subseteq L$. Se ha demostrado que, siendo U el autómata universal de L , el autómata A es irreducible en L si y sólo si existe algún homomorfismo de A en U y, además, todo homomorfismo de A en U es inyectivo. De aquí se desprende que un autómata finito A es irreducible (en $L(A)$) si y solo todo homomorfismo de A sobre el autómata universal para $L(A)$ es inyectivo y que por tanto el tamaño máximo para cualquier autómata finito irreducible está acotado por 2^n siendo n el número de estados del autómata finito determinista mínimo equivalente a A .

Se ha definido muestra estructuralmente universal de un lenguaje regular L como cualquier muestra positiva S de L tal que un autómata es un autómata irreducible para L si y solo si es un autómata cociente irreducible en L del aceptor canónico maximal de S . Se ha demostrado que para todo lenguaje regular existe una muestra estructuralmente universal y que si S es una muestra estructuralmente universal para L y $S \subseteq S' \subseteq L$ entonces también lo es S' . A partir de este resultado se ha descrito un esquema de inferencia incremental mediante datos positivos y un oráculo para la inclusión que aprende en el límite la clase de los lenguajes regulares representados mediante autómatas finitos.

Así mismo se ha definido muestra completa universal de un lenguaje regular L como cualquier muestra $\langle S_1, S_2 \rangle$ de L tal que $BS_{MCA}(S_1, S_2)$ es el conjunto formado exclusivamente por todos los autómatas irreducibles que reconocen L , esto es, un autómata irreducible reconoce L si y solo si es un autómata cociente del aceptor canónico maximal de S_1 irreducible en el complementario de S_2 . De este resultado se deduce que cualquier proceso de fusión de estados en el aceptor maximal canónico de una muestra positiva, compatible con una muestra negativa y que respete ciertos principios bastante débiles, es de hecho, un sistema para la inferencia en el límite de la clase de los lenguajes regulares mediante representación no determinista. Basándonos en este esquema hemos desarrollado algunos algoritmos sencillos que compiten con éxito con los algoritmos de referencia en la actualidad.

Por último, se ha definido subautómata asociado a una palabra x en el lenguaje L como cualquier autómata finito irreducible en L obtenido como cociente del aceptor canónico maximal de $\{x\}$ y se ha demostrado que para todo lenguaje regular L existe una muestra finita S tal que L es la unión de los lenguajes reconocidos por los subautómatas asociados a las palabras de M en L . A partir de este resultado se ha descrito un esquema para la inferencia en el límite mediante representación no determinista de la clase de los lenguajes regulares y se han implementado algoritmos que siguen este esquema con la

particularidad de que su complejidad depende solo linealmente del número de palabras de la muestra y cuadráticamente de la longitud de cada una de las mismas, lo que permite utilizar estos algoritmos en tareas que por su tamaño serían inabordables para otros algoritmos de mayor complejidad. Además se ha comparado satisfactoriamente el comportamiento de alguno de estos algoritmos con el de otros algoritmos para la misma tarea, particularmente en el caso en que el lenguaje objetivo viene representado por expresiones regulares generadas pseudoaleatoriamente.

En cuanto a futuras líneas de ampliación de este trabajo mencionaremos las siguientes: Desarrollo de algoritmos eficientes para la inferencia tanto de la clase de los lenguajes regulares como de algunas de sus subclases mas significativas, estudio de algunos de los eurísticos ya existentes para estas tareas para, a la luz de estos resultados, averiguar si son realmente algoritmos o si pueden llegar a serlo mediante la introducción de pequeñas modificaciones, el estudio, similar al efectuado para la irreducibilidad y la concisión, de otras propiedades deseables en los autómatas finitos, como por ejemplo la primalidad, etc.

Referencias

- [1] A. Arnold, A. Dicky and M. Nivat. A note about minimal non-deterministic automata. Bull. EATCS 47, pp 166-169, 1992.
- [2] G. I. Álvarez, J. Ruiz, A. Cano and P. García. Non deterministic regular positive negative inference NRPNI. Proc. of the XXXI Conferencia latinoamericana de informática, pp 239-249, 2005.
- [3] J. A. Brzozowski. Canonical regular expressions and minimal state graphs for definite events. Mathematical theory of automata, MRI symposia series 12, pp 520-561, 1962.
- [4] C. Carrez. On the minimization of non-deterministic automaton. Laboratoire de calcul de la faculté des sciences de l'université de Lille, 1970.
- [5] J. M. Champarnaud and F. Coulon. NFA Reduction algorithms by mean of regular inequalities. Theoretical computer science 327, pp 241-253, 2004.
- [6] F. Coste and D. Fredouille. Unambiguous automata inference by means of state-merging methods. ECML 2003.
- [7] J. H. Conway. Regular algebra and finite machines. Chapman and Hall, 1971.
- [8] C. Câmpeanu, N. Sântean and S. Yu. Mergible states in large NFA. Theoretical computer science 330, pp 23-34, 2005.
- [9] F. Denis, A. Lemay and A. Terlutte. Learning regular languages using nondeterministic finite automata. ICGI 2000, LNAI 1891, pp 39-50, 2000.
- [10] F. Denis, A. Lemay and A. Terlutte. Residual finite state automata. STACS 2001, LNAI 2010, pp 144-157, 2001.
- [11] F. Denis, A. Lemay and A. Terlutte. Learning regular languages using RFSAs. Theoretical computer science 313, pp 267-294, 2004.
- [12] P. Dupont, L. Miclet and E. Vidal. What is the search space of the regular inference?. ICGI 1994, LNCS 862, pp 25-37, 1994.
- [13] J. A. Feldman. Some decidability results on grammatical inference and complexity. Information and control 20, pp 244-262, 1972.
- [14] E. M. Gold. Language identification in the limit. Information and control 37, pp 447-474, 1967.
- [15] E. M. Gold. Complexity of automaton identification from given data. Information and control 37, pp 302-320, 1978.
- [16] P. García and M. Vázquez de Parga. A note about mergible states in large NFA. Bull. of the EATCS 87, pp 181-184, 2005.

- [17] M. A. Harrison. Introduction to formal language theory. Addison-Wesley, 1978.
- [18] C. de la Higuera. Characteristic sets for polynomial grammatical inference. *Machine Learning* 27, pp 125-137, 1997.
- [19] J. E. Hopcroft and J. D. Ullman. Introduction to automata theory, languages and computation. Addison-Wesley, 1979.
- [20] L. Ilie, G. Navarro and S. Yu. On NFA reductions. LNCS 3113, pp 112-124, 2004.
- [21] L. Ilie and S. Yu. Algorithms for computing small NFAs. LNCS 2420, pp 328-340, 2002.
- [22] L. Ilie and S. Yu. Reducing NFAs by invariant equivalences. *Theoretical computer science* 306, pp 373-390, 2003.
- [23] T. Kameda and P. Weiner. On the state minimalization of nondeterministic finite automata. *IEEE transactions on computers*, vol C-19, numb 7, pp 617-627, 1970.
- [24] K. J. Lang. Random DFA's can be approximately learned from sparse uniform examples. *Proc. of the fifth annual ACM workshop on computational learning theory*, pp 45-52, 1992.
- [25] K. J. Lang, B. A. Pearlmutter and R. A. Price. Results of the Abbadingo one DFA learning competition and a new evidence-driven state merging algorithm. *ICGI 98, LNAI 1433*, pp 1-12, 1998.
- [26] S. Lombardy and J. Sakarovitch. On the star height of rational languages. A new presentation for two old results. *Proc. of Words, languages and combinatorics III*. World scientific, pp 266-285, 2003.
- [27] O. Matz and A. Potthoff. Computing small nondeterministic finite automata. *BRICS notes series, May 95*, pp 74-78, 1995.
- [28] J. Oncina and P. García. Inferring regular languages in polynomial update time. *Pattern recognition and image analysis*, pp 49-61, 1992.
- [29] L. Polák. Syntactic semiring and universal automaton. *Proc. of Developments in language theory, LNCS 2710*, pp 411-422, 2003.
- [30] L. Polák. Minimalizations of NFA using the universal automaton. LNCS 3317, pp 325-336, 2005.
- [31] M. O. Rabin and D. Scott. Finite automata and their decision problems. *IBM journal of research and development* 3, pp 114-125, 1959.
- [32] B. Traktenbrot and Y. Barzdin. Finite automata: Behavior and synthesis, 1973.

- [33] L. G. Valiant. A theory of the learnable. *Communications of the ACM*, vol 27, numb 11, pp 1134-1142, 1984.
- [34] R. M. Wharton. Aproximate language identification. *Information and control* 33, pp 253-255, 1974.
- [35] S.Yu. Handbook of formal languages, Regular languages, vol 1, chap 2. Springer Verlag, 1997.