Propiedades de los lenguajes regulares

Propedéutico de Ciencias Computacionales

Plan

Section 1

Propiedades de los Lenguajes Regulares

Propiedades de los Lenguajes Regulares

Existen diferentes herramientas que se pueden utilizar sobre los lenguajes regulares:

- El lema de Pumping: cualquier lenguaje regular satisface el *pumping lemma*, el cual se puede usar para probar que un lenguaje no es regular.
- Propiedades de cerradura: se pueden construir autómatas a partir de componentes usando operaciones, v.g., dado un lenguaje L y M construir un autómata para $L \cap M$.
- Propiedades de decisión: análisis computacional de autómatas, v.g., probar si dos autómatas son equivalentes.
- Técnicas de minimización: útiles para construir máquinas más pequeñas.

Propiedades de los Lenguajes Regulares

- La clase de lenguajes conocidos como lenguajes regulares tienen al menos 4 descripciones: *DFA*, *NFA*, $\epsilon - NFA$ y *RE*.
- No todos los lenguajes son regulares, por ejemplo, $L = \{0^n 1^n | n \geq 1\}$.
- Si suponemos que el lenguaje es regular y es aceptado por un DFA A de k estados, si lee al menos k0's se tiene que cumplir que dos estados se repitan, esto es que para $i < j, p_i = p_j$. Llamemos a este estado q. Si $\delta(q, 1^i) \in F$ entonces el máquina acepta erróneamente $0^j 1^i$, y si $\notin F$ entonces la máquina rechaza erróneamente $0^i 1^i$.
- El problema es que el DFA tiene que tener historia de cuántos 0's lleva para poder aceptar el mismo número de 1's y que este número es variable.

Section 2

Lema de Pumping

Lema de Pumping

Si L es un lenguaje regular, entonces existe una constante n tal que cada cadena $w \in L$, de longitud n o más, puede ser escrita como $w = xyz$, donde:

1. $y \neq \epsilon$
2. $|xy| \leq n$
3. Para toda $i \geq 0, wy^iz$ también está en L. Notese que $y^i = y$ repetida i veces; $y^0 = \epsilon$.

Lo que dice es que si tenemos una cadena con una longitud mayor al número de estados del autómata, entonces una cadena no vacía y puede ser repetida (*pumped*) un número arbitrario de veces.

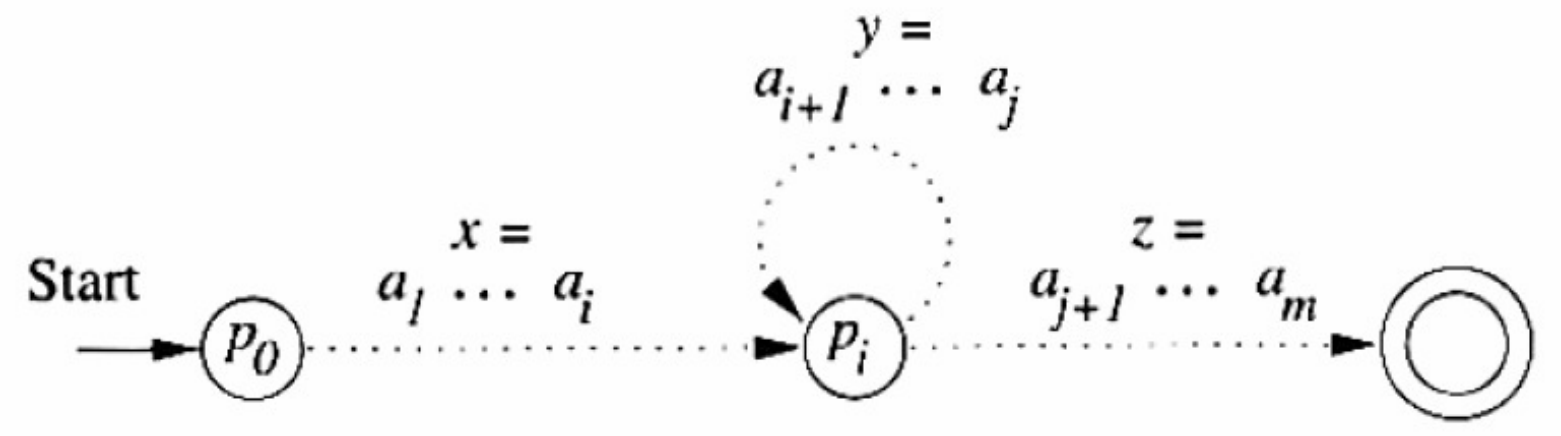
Prueba del Lema de Pumping

- Como se da por hecho que L es regular, debe haber un DFA A tal que $L = L(A)$. Si A tiene n estados; escogemos esta n para el lema de pumping.
- Sea w una cadena de longitud $\geq n$ en L, como en $w = a_1 a_2 \ldots a_m$, donde $m \geq n$.
- Sea q_i el estado en que A esta después de leer los primeros i símbolos de w.
- $q_0 =$ estado de inicio, $q_1 = \delta(q_0, a_1), q2 = \delta'(q_0, a_1 a_2)$, etc.

Prueba del Lema de Pumping

- Como sólo hay n estados diferentes, dos de $q_0, q_1, \ldots, q_n$ deben ser los mismos; digamos $q_i = q_j$, donde $0 \leq i < j \leq n$.
- Sea $x = a_1 \ldots a_i; y = a_{i+1} \ldots a_j; z = a_{j+1} \ldots a_m$. Entonces, si repetimos el ciclo desde q_i a q_i con la etiqueta $a_{i+1} \ldots a_j$ cero veces por una ocasión, o más, se puede probar que xy^iz es aceptado por A.

Lema de Pumping



Uso del Lema de Pumping

El Lema de Pumping se utiliza para mostrar que un lenguaje L no es regular.

- Se inicia asumiendo que L es regular.
- Luego, debe haber alguna n que sirva como la constante de PL. Puede que no sepamos el valor de n, pero podemos seguir con el resto del "juego" con n como un parámetro.
- Escogemos una w que sabemos que está en L. Típicamente, w depende de n.
- Aplicando el PL, sabemos que w puede descomponerse en xyz, satisfaciendo las propiedades del PL. De nuevo, puede que no sepamos como descomponer w, así que utilizaremos x, y, z como parámetros.
- Derivamos una contradicción escogiendo i (la cual puede depender de n, x, y, y/ó z) tal que xy^iz no está en L.

Ejemplo

Considere el lenguaje de cadenas con el mismo número de 0's y 1's. Por el pumping lemma, $w = xyz, |xy| \leq n, y \neq \epsilon$ y $xy^k z \in L$.

$$w = \underbrace{000\ldots}_{x}\underbrace{\ldots 0}_{y}\underbrace{0111\ldots 11}_{z}$$

En particular, $xz \in L$, pero xz tiene menos 0's que 1's.

Ejemplo 2

Supongamos que $L = 1^p : p$ es primo es regular. Sea n el parámetro del pumping lemma y seleccionemos un primo $p \geq n + 2$.

$$w = \overbrace{\underbrace{111\ldots}_{x}\underbrace{\ldots}_{y}\underbrace{1}_{y}\underbrace{111\ldots11}_{z}}^{p}$$

$|y| = m$
Ahora $xy^{p-m}z$ está en L.
$|xp^{p-m}z| = |xz| + (p-m)|y| = p - m + (p-m)m = (1+m)(p-m)$
Que no es un número primo, a menos que uno de los factores sea 1. Pero: $y \neq \epsilon \Rightarrow 1 + m > 1$ y
$m = |y| \leq |xy| \leq n, p \geq n + 2 \Rightarrow p - m \geq n + 2 - n = 2$

Problemas

Problema 1: Considere el problema 0^n10^n y demuestre que no es regular.

Problemas

Problema 2: Considere que el conjunto de cadenas de 0's cuya longitud es un cuadrado perfecto; formalmente
$L = \{0^i | i \text{ es un cuadrado}\}$.
Suponemos que L es regular. Entonces hay una n constante que satisface las condiciones del PL.
Considere $w = 0^{n^2}$, que seguramente estará en L.
Entonces $w = xyz$, donde $|xy| \leq n$ y $y \neq \epsilon$
Por PL $xyyz$ está en L. Pero la longitud de $xyyz$ es más grande que n^2 y no más grande que $n^2 + n$.
Sin embargo, el próximo cuadrado perfecto después de n^2 es $(n+1)^2 = n^2 + 2n + 1$.
Así, $xyyz$ no es de longitud cuadrada y no está en L.
Como hemos derivado una contradicción, la única asunción que no se ha probado –que L es regular– debe ser una falla, y tenemos una "prueba por contradicción" que L no es regular.

Section 3

Propiedades de Cerradura

Propiedades de Cerradura

Algunas operaciones sobre lenguajes regulares garantizan producir lenguajes regulares:

- Unión: $L \cup M$ lenguajes con cadenas en L, M o en ambos.
- Intersección: $L \cap M$ lenguajes con cadenas en ambos.
- Complemento: $\overline{L}$ cadenas que no están en L.
- Diferencia: $L \backslash M$ o $L - M$.
- Inversión: $L^R = \{w^R : w \in L\}$
- Cerradura: L^*
- Concatenación: $L.M$
- Homomorfismo (substitución): $h(L) = \{h(w) \in L\} h$ es un homomorfismo.
- Homomorfismo inverso (substitución inversa): $h^{-1}(L) = \{w \in \Sigma : h(w) \in L, h : \Sigma \rightarrow\}$ es un homomorfismo.

Propiedades de Cerradura

- **Unión**: la unión de lenguajes regulares es regular. Sea $L = L(E)$ y $M = L(F)$. Entonces $L(E + F) = L \cup M$, por la definición de "+" en RE.
- **Complemento**: Si L es un lenguaje regular sobre Σ, entonces también lo es $\overline{L} = \Sigma^* \ L$. Todos los estados son de aceptación excepto los F.
- **Ejemplo**: Sea L definido por el siguiente DFA (el lenguaje de cadenas que terminan en 01):

Uso de Propiedades

- Las cadenas de un número diferente de 0's y 1's es difícil probarlo con el *pumping lemma*. Sin embargo, ya probamos que $L = 0^n1^n$ no es regular. $M = 0^n1^m, n \neq m$ es $\overline{L}$. Como L no es regular su complemento tampoco.
- **Intersección**: Si L y M son regulares, entonces también $L \cap M$. Usando las leyes de Morgan: $L \cap M = \overline{\overline{L} \cup \overline{M}}$.
- Para esto también se puede construir un autómata que simula A_L y A_M en paralelo y llega a un estado de aceptación si A_L y A_M también lo hacen.

Propiedades

- **Diferencia**: $L \backslash M$ lenguaje en L pero no en M, $L \backslash M = L \cap \overline{M}$.
- **Inversión**: w^R es la cadena invertida w. Ejemplo: $0010^R = 0100$.
- Inversión de un lenguaje L es el lenguaje con todas las cadenas de L invertidas. Por ejemplo:
  $L = \{001, 10, 111\}$, $L^R = \{100, 01, 111\}$.
  Se puede probar que $L(E^R) = (L(E))^R$.
  En particular, $E^R = E$ para $\emptyset$ conjunto vacío y a.
  Si $E = F + G$, entonces, $E^R = F^R + G^R$
  Si $E = F \cdot G$, entonces $E^R = G^R \cdot F^R$

Ejemplo

- Por ejemplo, $L(E_1) = \{01, 111\}$ y $L(E_2) = \{00, 10\}$.
- $L(E_1)L(E_2) = \{0100, 0110, 11100, 11110\}$
- $L^R(E_1) = \{10, 111\}$ y $L^R(E_2) = \{00, 01\}$
- $L^R(E_1)L^R(E_2) = \{0010, 00111, 0110, 01111\} = (L(E_1)L(E_2))^R$
- Si $L = (0 + 1)0^*$, $L^R = (0^*)^R(0 + 1)^R = 0^*(0 + 1)$
- Si $E = F^*$, entonces $E^R = (F^R)^*$

Homomorfismo

- Un *homomorfismo* sobre Σ es una función $h : \Sigma^* \to \Theta^*$, donde Σ y Θ son alfabetos.
- Sea $w = a_1 a_2 \dots a_n \in \Sigma$. Entonces $h(w) = h(a_1)h(a_2)\dots h(a_n)$ y $h(L) = \{h(w) \in L\}$.
- **Ejemplo**: sea $h : \{0,1\}^* \to \{a,b\}^*$ definido como $h(0) = ab$, y $h(1) = \epsilon$. Entonces $h(0011) = abab$ y $h(L(10^*1)) = L((ab)^*)$.
  $h(L)$ es regular si L es regular.
  Sea $L = L(E)$ para una RE E, queremos probar que $L(h(E)) = h(L)$.

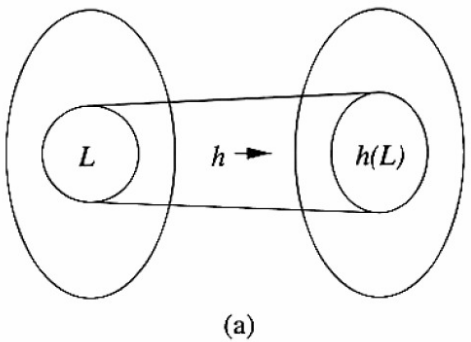
Prueba

Base: E es ϵ ó $\emptyset$, $h(E) = E$ y $L(h(E)) = L(E) = h(L(E))$.
E es un solo símbolo, a. Entonces $L(E) = a$, y
$L(h(E)) = L(h(a)) = \{h(a)\} = h(L(E))$.
Inducción: Hay tres casos, dependiendo de si
$R = R_1 + R_2, R = R_1R_2$, ó $R = R_1^*$.

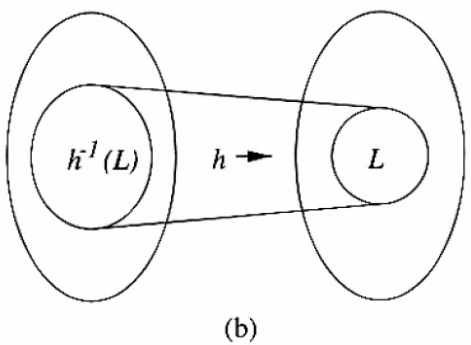
- $L = E + F, L(h(E + F)) = L(h(E) + h(F)) = L(h(E)) \cup L(h(F)) = h(L(E)) \cup h(L(F)) = h(L(E) \cup L(F)) = h(L(E + F))$
- $L = E \cdot F, L(h(E \cdot F)) = L(h(E)) \cdot L(h(F)) = h(L(E)) \cdot h(L(F)) = h(L(E) \cdot L(F))$
- $L = E^*, L(h(E^*)) = L(h(E)^*) = h(L(E))^* = h(L(E^*))$

Homomorfismo Inverso

Sea $h : \Sigma^* \to \Theta^*$ un homomorfismo. Sea $L \subseteq \Theta^*$, entonces $h^{-1}(L) = \{w | w \in \Sigma^* : h(w) \in L\}$.



(a)



(b)

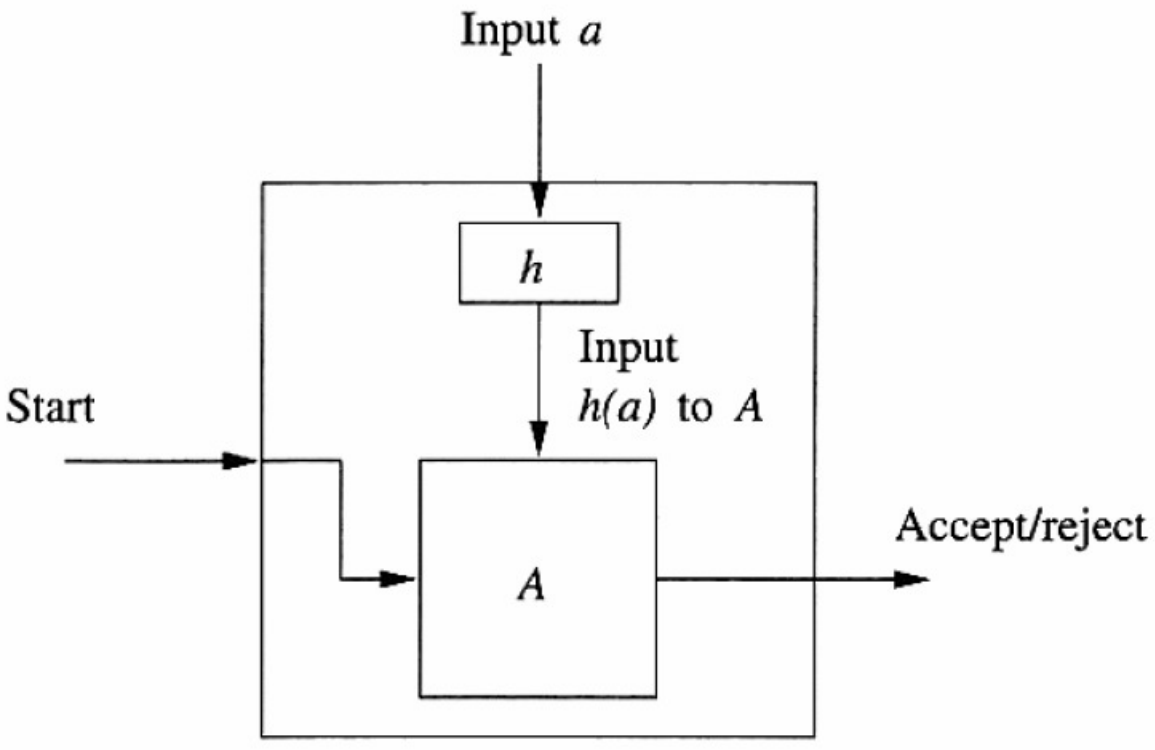
Ejemplo

- Sea L el lenguaje de la expresión regular $(00 + 1)^*$. Sea h un homomorfismo definido por: $h(a) = 01$ y $h(b) = 10$.
- En particular: $h^{-1}(L) = (ba)^*$ (que combinación de a's y b's me dan el mismo lenguaje?)
- $h(ba) = 1001$ y $h(w)$ que son n repeticiones de 1001 están en L.
- Por otro lado, si w empieza solo con a (01), termina con b (10), tiene dos a's seguidas (0101) o dos b's seguidas (1010), no se cumple

Homomorfismo Inverso

- Sea $h : \Sigma^* \to \Theta^*$ un homomorfismo. Sea $L \subseteq \Theta^*$ un lenguaje regular, entonces $h^{-1}(L)$ es regular.
- El DFA del homomorfismo inverso usa los estados del DFA original, pero cambia los símbolos de entrada de acuerdo a la función h antes de decidir el siguiente estado.
- En la prueba lo importante es que los estados de aceptación de los dos autómatas son los mismos y que $h(a)$, para cualquier símbolo a dentro del alfabeto del DFA original puede ser ϵ, un símbolo o muchos símbolos.
  $\gamma(q, a) = \hat{\delta}(q, h(a))$.
  La prueba se hace por inducción sobre $|w|$.
  $\hat{\gamma}(q_0, w) = \hat{\delta}(q_0, h(w))$.

Homomorfismo Inverso



Ejemplos

Si h del alfabeto $\{0, 1, 2\}$ al alfabeto $\{a, b\}$ se define por: $h(0) = a; h(1) = ab; h(2) = ba$:

1. Qué es $h(0120)$?
   $aabbaa$
2. Si L es el lenguaje $L(01^*2)$ cual es $h(L)$?
   $a(ab)^*ba$
3. Si L es el lenguaje $\{ababa\}$ cual es $h^{-1}(L)$?
   Para obtener esa cadena podemos tener 110 o 022 o 102:
   $\{110, 022, 102\}$

Propiedades de decisión

- Algunas de las preguntas que nos podemos hacer acerca de lenguajes son si el lenguaje es vacío, si una cadena particular pertenece al lenguaje o si dos descripciones definen el mismo lenguaje.
- También podemos cuestionarnos acerca del costo computacional requerido para resolver estas preguntas o por ejemplo el requerido para hacer la conversión entre una representación a otra.

Análisis de Complejidad

Transformar un ϵ-NFA a un DFA:

- Si el ϵ-NFA tiene n arcos. Para calcular $ECLOSE(p)$ se requieren seguir a lo más n^2 arcos y el DFA tiene a lo más 2^n estados.
- Para cada símbolo y cada subconjunto el calcular la función de transición para todos los estados, requiere a lo más n^3 pasos, lo cual nos da una complejidad total de $O(n^3 2^n)$.
- En general el número de estados del DFA es de orden lineal (digamos s), por lo que en la práctica la complejidad se reduce a $O(n^3 s)$.

Análisis de Complejidad

Otras Transformaciones:

- **Transformar un DFA a un NFA**: Sólo se requiere poner corchetes a los estados, lo cual nos da O(n).
- **Transformar un FA a una RE**: $O(n^3 4^n)$. Es todavía peor si el FA es NFA. Si lo convertimos primero a un DFA nos da: $O(n^3 4^{n^3 2^n})$.
- **Transformar de una RE a un FA**: se puede construir un autómata en n pasos. Si eliminamos transiciones ϵ toma $O(n^3)$. Si se requiere un DFA puede tomar un número exponencial de pasos.

Análisis de Complejidad

- **Decidir si un lenguaje es vacío**: el probar si existe un camino entre un estado inicial a uno final o de aceptación, es simplemente un problema de ver si un nodo está conectado en un grafo, lo cual tiene una complejidad de $O(n^2)$.
- **Probar por pertenencia a un lenguaje regular**: ver si una cadena es miembro del lenguaje. Si la cadena w es de longitud n para un DFA, esto es de complejidad $O(n)$. Si es un NFA de s estados, entonces la complejidad es: $O(ns^2)$. Si es un ϵ-NFA entonces la complejidad es $O(ns^3)$.

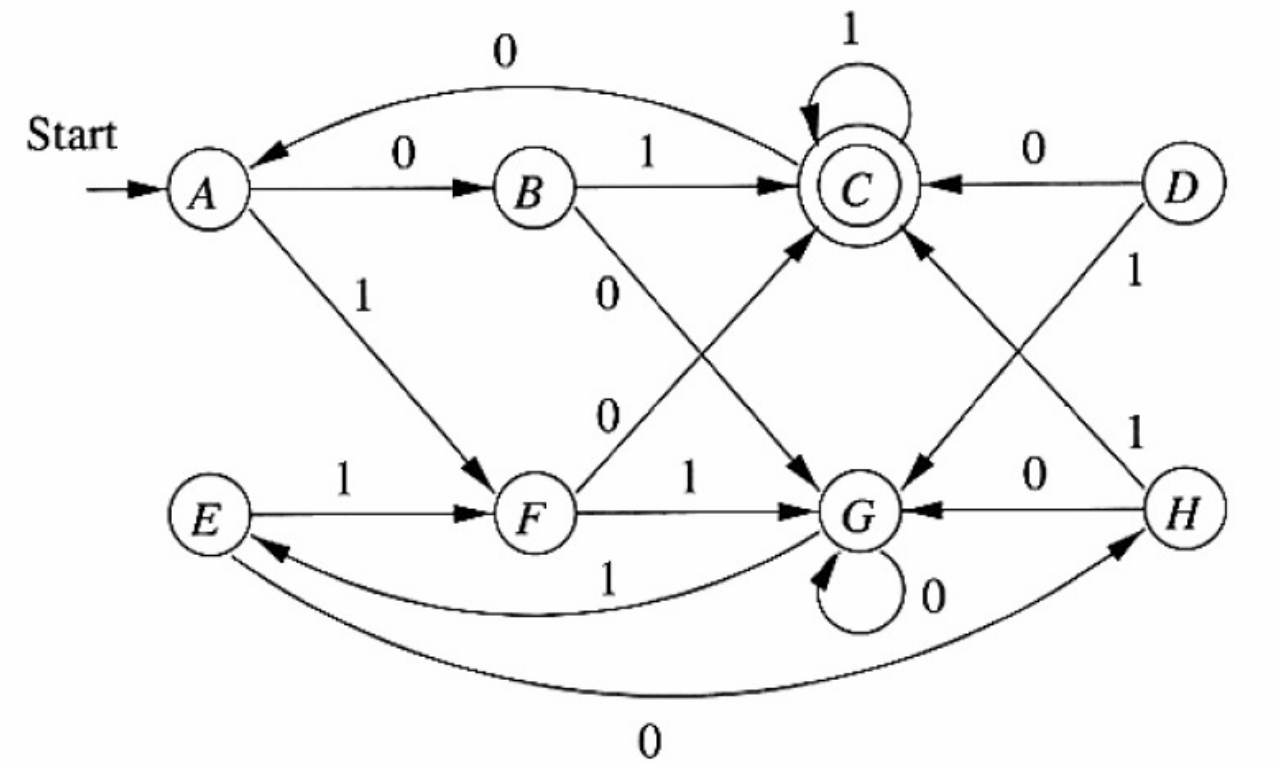
Section 4

Equivalencia y minimización de autómatas

Equivalencia y minimización de autómatas

- Lo que queremos saber es si dos autómatas diferentes definen el mismo lenguaje.
- Primero definiremos lo que son estados equivalentes.
- Dos estados p y q dentro de un autómata son *equivalentes*: $p \equiv q \Leftrightarrow \forall w \in \Sigma^* : \hat{\delta}(p, w) \in F \Leftrightarrow \hat{\delta}(q, w) \in F$.
- Si no, entonces se dice que son *distinguibles*. Osea que p y q son distinguibles si: $\exists w : \hat{\delta}(p, w) \in F \wedge \hat{\delta}(q, w) \notin F$ o viceversa.

Ejemplo



Ejemplo

$$\hat{\delta}(C,\epsilon) \in F, \hat{\delta}(G,\epsilon) \notin F \Rightarrow C \neq G$$
$$\hat{\delta}(A,01) = C \in F, \hat{\delta}(G,01) = E \notin F \Rightarrow A \neq G$$
$$\hat{\delta}(A,\epsilon) = A \notin F, \hat{\delta}(E,\epsilon) = E \notin F$$
$$\hat{\delta}(A,1) = F = \hat{\delta}(E,1) \therefore \hat{\delta}(A,1x) = \hat{\delta}(E,1x) = \hat{\delta}(F,1x)$$
$$\hat{\delta}(A,00) = G = \hat{\delta}(E,00)$$
$$\hat{\delta}(A,01) = C = \hat{\delta}(E,01)$$
$$\therefore A \equiv E$$

Table-Filling Algorithm

- También podemos encontrar los pares equivalentes usando el algoritmo de llenado de tabla (*table-filling algorithm*).
- Base: Si $p \in F \wedge q \neq F \Rightarrow p \neq q$
- Inducción: Si $\exists a \in \Sigma : \delta(p, a) \neq \delta(q, a) \Rightarrow p \neq q$

Ejemplo

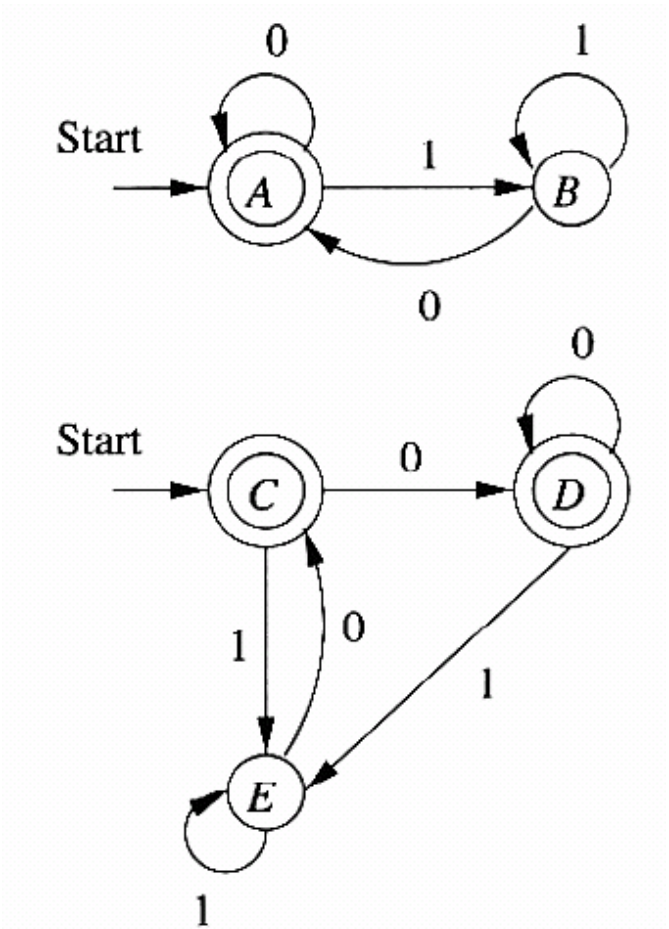
Por ejemplo, para el DFA anterior:

	A	B	C	D	E	F	G
B	x						
C	x	x					
D	x	x	x				
E		x	x	x			
F	x	x	x		x		
G	x	x	x	x	x	x	
H	x		x	x	x	x	x

Prueba de equivalencia entre lenguajes regulares

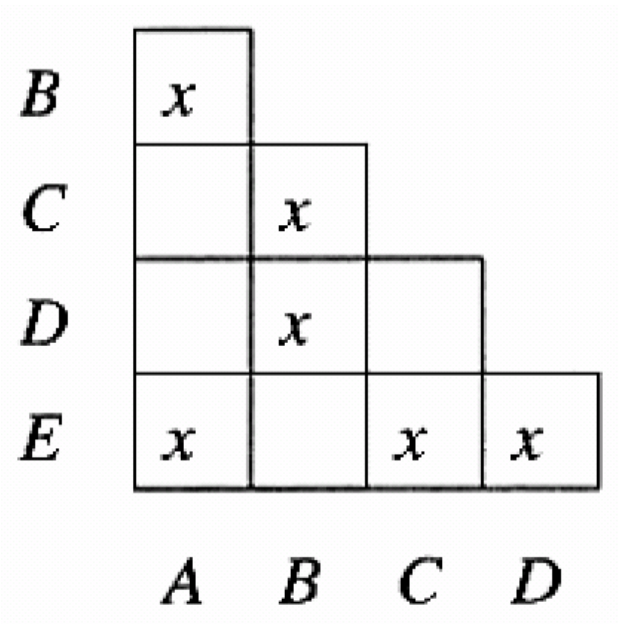
- Sea L y M dos lenguajes regulares (dados en alguna forma).
- Convierte L y M a DFA's
- Junta los dos DFA's
- Prueba si los dos estados iniciales son equivalentes, en cuyo caso $L = M$. Si no son equivalentes, entonces $L \neq M$.

Ejemplo



Ejemplo

Los dos DFA's aceptan: $L(\epsilon + (0+1)^*0)$. Si los consideramos a los dos como un solo autómata, el algoritmo de llenado de tabla nos da:



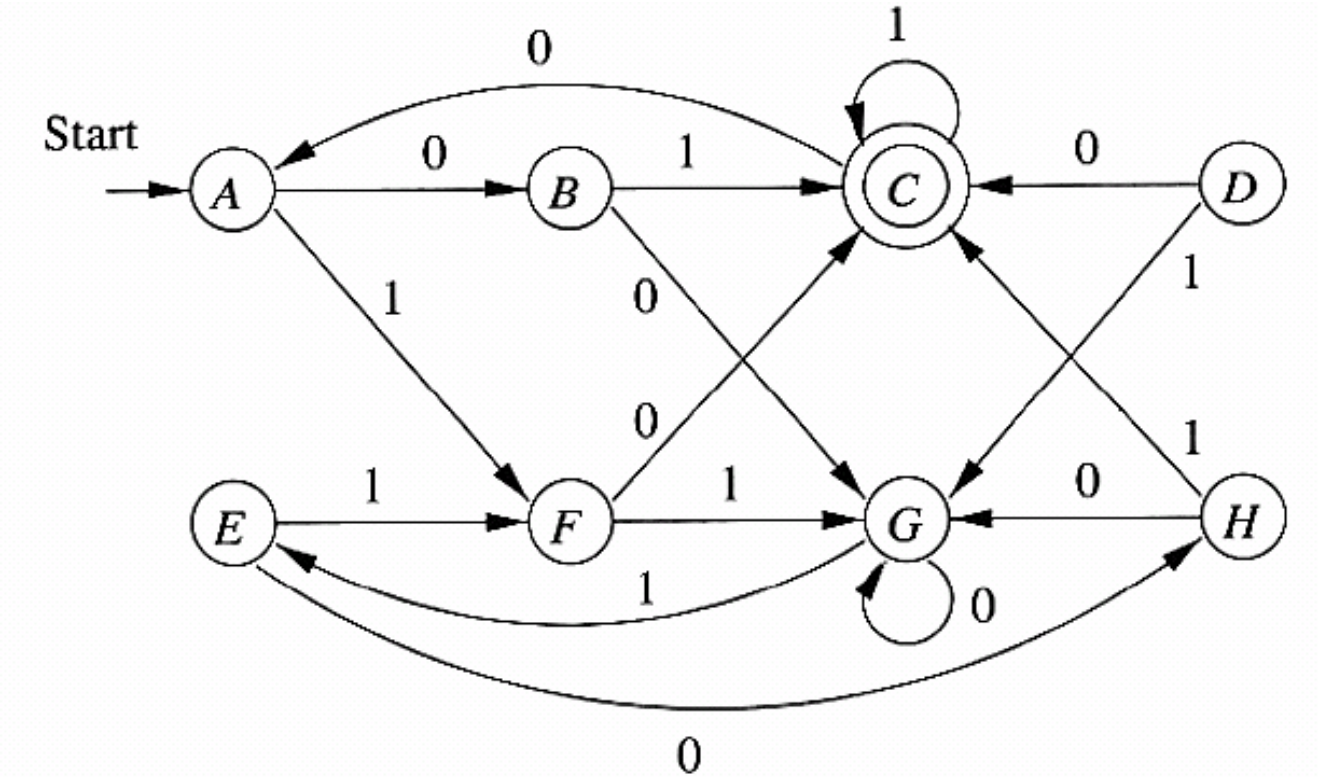
Lo que nos deja los siguientes pares de estados equivalentes: $\{A, C\}, \{A, D\}, \{C, D\}$ y $\{B, E\}$. Como A y C son equivalentes, entonces los dos autómatas son equivalentes.

Minimización de un DFA

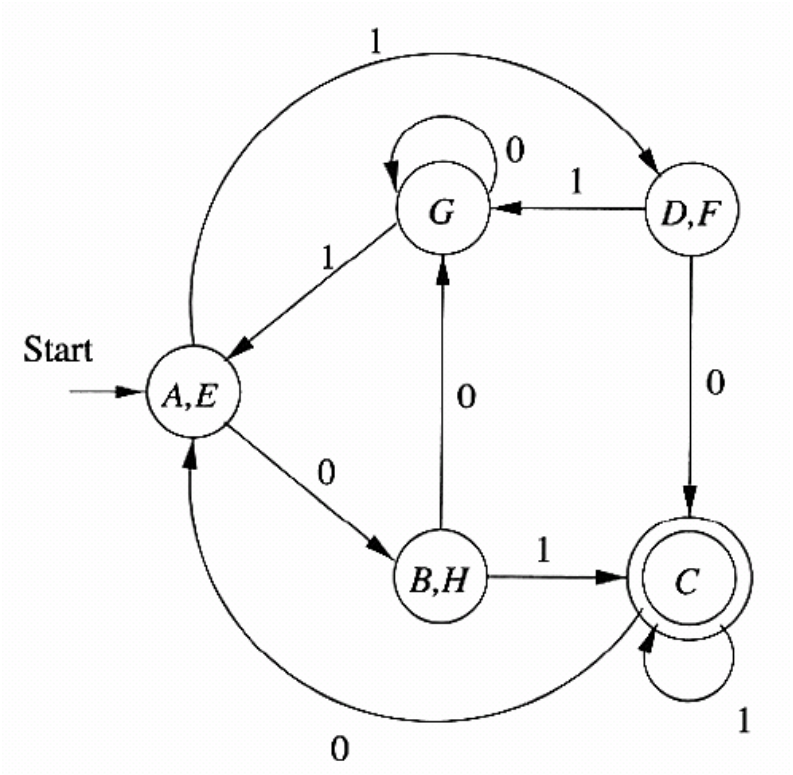
- Una consecuencia de encontrar estados equivalentes, es que podemos reemplazar los estados equivalentes por un solo estado representado por su unión.
- Por otro lado, la relación de equivalencia cumple con ser reflexiva, simétrica y transitiva.
- La prueba de equivalencia de estados es transitiva, osea que si encontramos que p y q son equivalentes y q y r son equivalentes, entonces p y r son equivalentes.

Ejemplo

Minimizar el siguiente autómata usando los estados equivalentes.



Ejemplo



Nota: no podemos aplicar el algoritmo de llenado de tabla a un NFA.

Ejemplo

Suponga que se tiene la siguiente tabla de transción:

	0	1
$\rightarrow A$	B	A
B	A	C
C	D	B
$*D$	D	A
E	D	F
F	G	E
G	F	G
H	G	D

- Hacer la tabla de estados distinguibles
- Construir un DFA de estados mínimos equivalente

Material basado en el libro Introduction to Automata Theory Languages and Computation [Hopcroft et al., 2006].

Bibliography I


Hopcroft, J. E., Motwani, R., and Ullman, J. D. (2006).
Automata theory, languages, and computation.
International Edition, 24.