TECNOLÓGICO NACIONAL DE MEXICO

INSTITUTO TECNOLÓGICO DE IZTAPALAPA

INTEGRANTES:

CUANENEMI CUANALO MARIO ALBERTO 181080030
FERMIN CRUZ ERIK 181080007
GUTIERREZ ARELLANO RAFAEL 181080022
PEREZ ARMAS FAUSTO ISAAC 181080037

ISC-6AM

LENGUAJES Y AUTOMATAS I

M.C. ABIEL TOMÁS PARRA HERNÁNDEZ

SEP 2020 / FEB 2021

ACTIVIDAD SEMANA 7

Cuananemi Cuanalo Mario Alberto

DFA • El lenguaje de un DFA es el conjunto de todas las cadenas que el DFA acepta • Dada una cadena (e.g., s1, s2, . . . , sn) el DFA empieza en su estado inicial (e.g., q0), consulta si existe una transicion de ´ q0 con el primer s´ımbolo (s1) a otro estado (e.g., q1) y si existe (i.e., δ(q0, s1) = q1) se mueve al estado descrito en la transicion. ´ • Procesa el siguiente s´ımbolo de la cadena (i.e., s2) y as´ı continua. • Si logra procesar toda la cadena y el estado al que llega es uno de los estados finales, entonces se dice que el automata acepta esa cadena.

Ejemplo Un Automata A que acepta ´ L = {x01y|x ∧ y ∈ {0, 1} ∗} • El DFA acepta cadenas que tienen 01 en alguna parte de la cadena • El lenguaje del DFA es el conjunto de cadenas que acepta {w|w tiene la forma “x01y” para algunas cadenas x y y que consisten solo de 0’s y 1’s ´ }

Diferencia entre DFA y NFA:

NO SEÑOR.	DFA	NFA
1	DFA son las siglas de Deterministic Finite Automata.	NFA son las siglas de Nondeterministic Finite Automata.
2	Para cada representación simbólica del alfabeto, solo hay una transición de estado en DFA.	No es necesario especificar cómo reacciona la NFA según algún símbolo.
3	DFA no puede utilizar la transición de cadena vacía.	NFA puede utilizar la transición de cadena vacía.
4	DFA puede entenderse como una sola máquina.	NFA puede entenderse como múltiples pequeñas máquinas que computan al mismo tiempo.
5	En DFA, el siguiente estado posible se establece claramente.	En NFA, cada par de estado y símbolo de entrada puede tener muchos estados siguientes posibles.
6	DFA es más difícil de construir.	NFA es más fácil de construir.
7	DFA rechaza la cadena en caso de que termine en un estado diferente del estado de aceptación.	NFA rechaza la cuerda en caso de que todas las ramas mueran o rechacen la cuerda.
8	El tiempo necesario para ejecutar una cadena de entrada es menor.	El tiempo necesario para ejecutar una cadena de entrada es mayor.
9	Todos los DFA son NFA.	No todos los NFA son DFA.
10	DFA requiere más espacio.	NFA requiere menos espacio que DFA.

Teoria de Myhill-Nerode

Sea Ent un alfabeto finito. Según vimos anteriormente la noción de *lenguaje regular* puede presentarse mediante el reconocimiento por autómatas finitos, sean deterministas o no, o mediante la representación por expresiones regulares. Otras presentaciones equivalentes de esta noción se dan en la siguiente:

Proposición 6.1 (Teorema de Myhill-Nerode) Sea $L \subset Ent^*$ un lenguaje arbitrario. Las siguientes aseveraciones son equivalentes a pares:

1.
L es regular.
2.

L es la unión de algunas clases de equivalencia de una relación de equivalencia de índice finito, congruente por la derecha con la concatenación de palabras.
3.
La relación $1. \Rightarrow 2.$ tal que

$$L = L(AF$$

es una relación de índice finito.

Demostración: Supongamos *L* regular. Sea $AF = (Q, Ent, tran, q_0, F)$ un autómata finito tal que . Consideremos la función

$$\equiv_K$$

y su kernel:

$$L = \bigcup [\sigma]_{\equiv_K}.$$

Hemos visto que `` $2. \Rightarrow$ '' es una relación de equivalencia de índice finito (de hecho este índice está acotado por la cardinalidad de *Q*), congruente por la derecha. Se tiene además

$$R \subset (Ent^*)^2$$

Así pues, se cumple 2.

$\mathcal{F} \subset Q/R$ Sea $L = \bigcup [\sigma]$ una relación de equivalencia de índice finito, congruente por la derecha con la concatenación de palabras, tal que para un subconjunto se tiene que $\sigma R \tau$. Afirmamos que

$$\forall v \in Ent^* \tag{47}$$

En efecto, si $\sigma v R \tau v$ entonces, $(\sigma \cdot v \in L \Leftrightarrow$, al ser *R* congruente por la derecha, $\sigma \equiv_L \tau$.

Como *L* es la unión de algunas clases de *R*, esto da que $\equiv_L$. Así pues, $3. \Rightarrow 1.$. De la relación 4.36 vemos que toda clase de equivalencia respecto a `` $Q =$ '' es la unión de

clases de equivalencia respecto a *R*. Por tanto el índice de `` $Q =$ '' no puede exceder el de *R*. Como este último es finito el de `` $Q =$ '' es también finito.

$q_0 = [nil]$ `` $Q =$ '' es una relación de equivalencia de índice finito, congruente por la derecha. Definamos $AF = (Q, Ent, tran, q_0, F)$ haciendo $F = \{[\sigma] | \sigma \in L\}$,

$tran : ([\sigma], \quad q_1, q_2 \in Q$ y . Es evidente que una palabra es reconocida por AF si y sólo si esa palabra está en *L*. Así pues, $L = L(AF)$ y,consecuentemente, es regular.

Para un autómata finito $AF = (Q, Ent, tran, q_0, F)$ diremos que dos estados son *indistinguibles* si para cualquier palabra $\sigma \in Ent^*$, $\equiv_I$

Si dos estados no fueran indistinguibles, entonces habría una palabra $\sigma \in Ent^*$ tal que

$$\sigma, \tau \in Ent^*$$

Diremos que tal palabra *distingue* a los estados q_1,q_2. La relación de indistinguibilidad es una relación de equivalencia. La denotaremos como `` '' . Sea $L = L(AF)$ el lenguaje reconocido por el autómata AF . Para cualesquiera dos palabras $AFM_L = (Er$ rige la equivalencia siguiente

$$(3. \Rightarrow 1.)$$

Por tanto el autómata $AFM_I = (Q^{con}/\equiv_I)$ construído en la demostración de la implicación $h : T(\sigma) \mapsto$ del teorema de Myhill-Nerode 4.6.1 es isomorfo al autómata cociente AFM_L y, por consiguiente, es una imagen homomorfa de la parte conexa del autómata AF, AFM_I es un homomorfismo. De hecho, $\{tran(q_1,$ es una imagen homomorfa de cualquier autómata que reconozca a *L*. Así pues, resulta de inmediato la **Proposición 6.2** El autómata mínimo que reconoce a un lenguaje regular *L* es $AFM_I = (Q^{con}/\equiv_I)$.

Minimización de autómatas

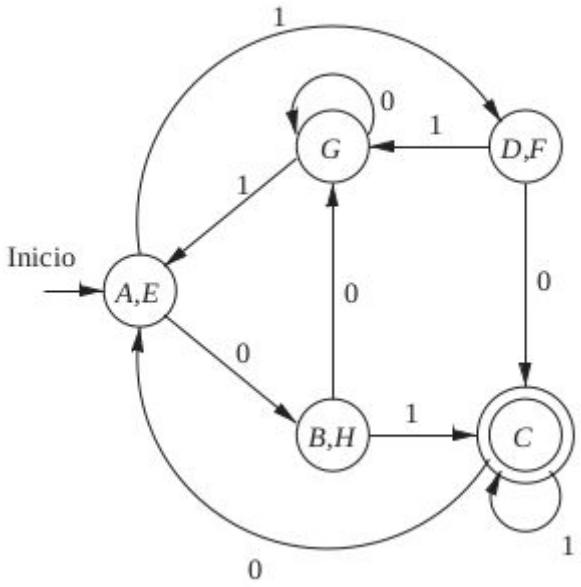
Construcción de un AFDt con un número de estados mínimo que sea equivalente a un AFDt dado. Definiciones previas: • Estados accesibles: q0 es accesible q accesible ⇒ ∀s∈Σ, δ(q, s) es accesible • Estados k-equivalentes o k-indistinguibles: p ≡k q ∀x∈Σ≤k (δ*(p, x)∈F ↔ δ*(q, x) ∈ F) • Estados equivalentes o indistinguibles: p ≡ q ∀k ∀x∈Σ≤k (δ*(p, x)∈F ↔ δ*(q, x) ∈F), es decir, ∀x∈Σ* (δ*(p, x)∈F ↔ δ*(q, x)∈F)

Construcción del AFDt mínimo N a partir del AFDt M = (Q, Σ, δ, q0, F) 1) Eliminar estados inaccesibles 2) Determinar las clases de estados equivalentes: p ≡0 q ⇔ (p∈F ↔ q∈F) p ≡k+1 q ⇔ (p ≡k q ∧ ∀s∈Σ δ(p, s) ≡k δ(q, s)) 3) Construcción del AFD N = (P, Σ, γ, p0, G) con P = Q/≡ siendo ≡ es la menor ≡k tal que ≡k coincide con ≡k+1 p0 = [q0] γ([p], s) = [δ(p, s)] G = {[p]: p∈F}

Minimización de un AFD

Dos pasos:

1. Eliminación de los estados no alcanzables
2. Búsqueda de estados equivalentes
3. Construcción de un autómata a partir de los grupos de estados equivalentes



Ejercicio 1

Considerar el automata siguiente:

	0	1
->	A	B	A
	B	A	C
	C	D	B
*	D	D	A
	E	D	F
	F	G	E
	G	F	G
	H	G	D

1. Dibuje la tabla de estados distinguibles para este autómata.
2. Construya el AFD equivalente con el número mínimo de estados.

Ejercicio 2

Mismo ejercicio con:

	0	1
->	A	B	E
	B	C	F
*	C	D	H

	D	E	H
	E	F	I
*	F	G	B
	G	H	B
	H	I	C
	I	A	E

Fermin Cruz Erik

DFA: Se refiere a un autómata finito determinista. Se dice que un autómata finito es determinista, si corresponde a un símbolo de entrada, y en este hay un solo estado resultante, es decir, solo hay una transición.

Un autómata finito determinista está formado por cinco tuplas y se representa como:

Donde:

Q: Un conjunto finito no vacío de estados presentes en el control finito (q0, q1, q2,…).

Σ: Un conjunto finito no vacío de símbolos de entrada.

δ: Es una función de transición que toma dos argumentos, un estado y un símbolo de entrada, devuelve un solo estado.

q0: Es el estado inicial, uno de los estados en Q.

F: Es un conjunto no vacío de estados finales / estados de aceptación del conjunto que pertenece a Q.

NFA: Se refiere a un autómata finito no determinista. Se dice que un autómata finito no es determinista, si hay más de una posible transición desde un estado en el mismo símbolo de entrada.

Un autómata finito no determinista también está formado por cinco tuplas y se representa de la siguiente manera:

Donde:

Q: Un conjunto de estados finitos no vacíos.

Σ: Un conjunto de símbolos de entrada finitos no vacíos.

δ: Es una función de transición que toma un estado de Q y un símbolo de entrada y devuelve un subconjunto de Q.

qo: Estado inicial de NFA y miembro de Q.

F: Un conjunto no vacío de estados finales y miembro de Q .

TEOREMA DE NERODE

Caracteriza los lenguajes regulares como subconjuntos del monoide que son satuados por una congruencia de índice infinito.

Sea ‘S’ un semigrupo. Un subconjunto X de S se llama reconocible si está saturado por una congruencia en ‘S’ de índice finito. Denotamos por Rec (S) el conjunto formado por todos los subconjuntos de ‘S’ reconocibles. Sea ‘S’ un semigrupo y ‘X’ un subconjunto de ‘S’. Se define la congruencia sintáctica de X como s ≈Xt⇔ ∀u, v∈S1(usv∈X⇔utv∈X)

MINIMIZACIÓN DE UNA NFA

Si bien una búsqueda exhaustiva puede minimizar un NFA, no existe un algoritmo de tiempo polinomial para minimizar los NFA generales a menos que P = PSPACE, una conjetura no resuelta en la teoría de la complejidad computacional que se cree ampliamente que es falsa.

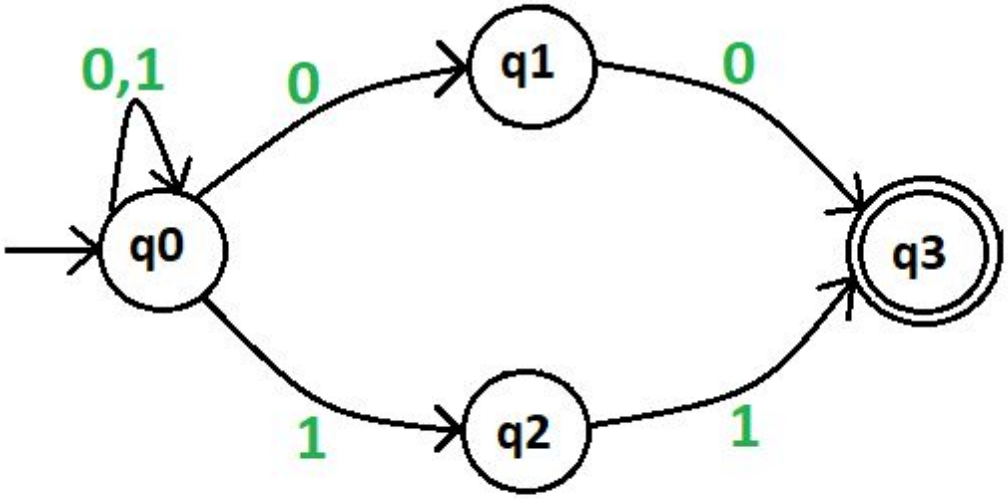
Sin embargo, existen métodos de minimización de NFA que pueden ser más eficientes que la búsqueda por fuerza bruta.

Gutierrez Arellano Rafael

Autómata finito NO-Determinista.

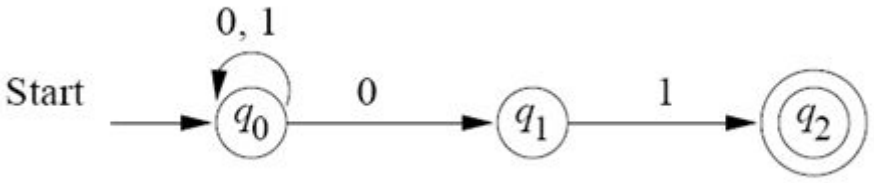
Este tipo de autómata es aquel que permite al autómata finito (AF) tiene 0 o más estados consecutivos para cada estado y su entrada.
Con lo anterior podemos decir que un NFA puede estar en varios estados a la vez o que se puede que ‘adivina' o que estado debe ir.

NFA

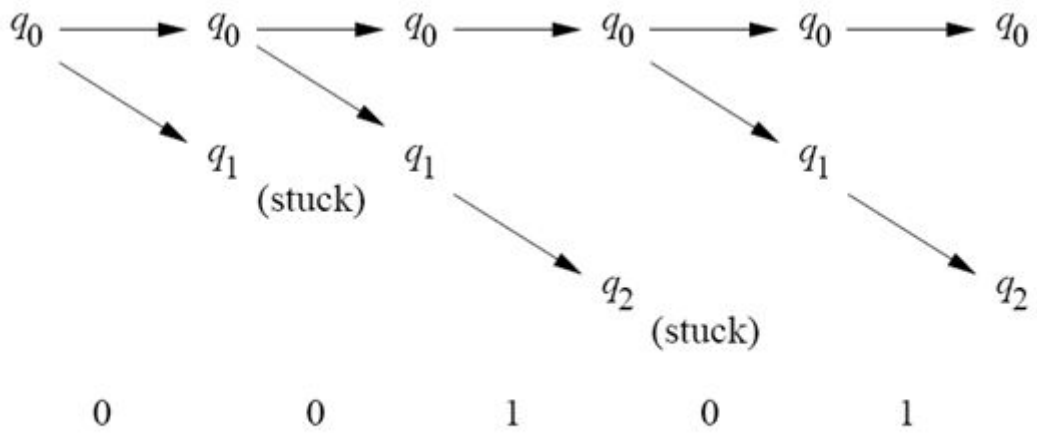


Los autómatas finitos se pueden representar mediante grafos particulares, también llamados diagramas de estados finitos

Por ejemplo, el siguiente autómata acepta todas las cadenas que terminan en 01:



Lo que pasa al procesar como entrada a 00101 es:



EN SU FUNCIONAMIENTO:

1. En el comienzo del proceso de reconocimiento de una cadena de entrada.
2. el autómata finito se encuentra en el *estado inicial* y a medida que procesa cada símbolo de la cadena va cambiando de estado de acuerdo a lo determinado por la *función de transición*.
3. Cuando se ha procesado el último de los símbolos de la cadena de entrada, el autómata se detiene en el estado final del proceso.
4. Si el estado final en el que se detuvo es un estado *de aceptación*, entonces la cadena pertenece al lenguaje reconocido por el autómata; en caso contrario, la cadena no pertenece a dicho lenguaje.

Otra manera de describir el funcionamiento de un autómata finito es mediante el uso de tablas de transiciones o matrices de estados.

AUTÓMATA DFA

Autómata finito determinista, un autómata finito determinista (DFA), también conocido como aceptor finito determinista (DFA), máquina de estado finito determinista (DFSM) o autómata de estado finito determinista (DFSA) - es una máquina de estados finitos que acepta o rechaza una determinada cadena de símbolos, mediante la ejecución de una secuencia de estados determinada únicamente por la cadena.

Complemento

Los autómatas finitos deterministas siempre están completos : definen una transición para cada estado y cada símbolo de entrada.

Si bien esta es la definición más común, algunos autores usan el término autómata finito determinista para una noción ligeramente diferente: un autómata que define como máximo una transición para cada estado y cada símbolo de entrada; se permite que la función de transición sea parcial . [5] Cuando no se define ninguna transición, dicho autómata se detiene.

Los DFA son equivalentes en potencia de cálculo a los autómatas finitos no deterministas (NFA). Esto se debe a que, en primer lugar, cualquier DFA también es un NFA, por lo que un NFA puede hacer lo que un DFA puede hacer. Además, dada una NFA, utilizando la construcción del conjunto de potencias se puede construir una DFA que reconozca el mismo lenguaje que la NFA, aunque la DFA podría tener un número exponencialmente mayor de estados que la NFA. [13] [14]Sin embargo, aunque los NFA son computacionalmente equivalentes a los DFA, los problemas mencionados anteriormente no se resuelven necesariamente de manera eficiente también para los NFA. El problema de no universalidad de los NFA es el PSPACE completo, ya que hay NFA pequeños con la palabra de rechazo más corta en tamaño exponencial. Un DFA es universal si y sólo si todos los estados son estados finales, pero esto no es válido para los NFA. Los Problemas de Igualdad, Inclusión y Minimización también son PSPACE completos ya que requieren formar el complemento de una NFA lo que resulta en una explosión exponencial de tamaño.

MINIMIZACIÓN.

Pasos a la minimización

1. Eliminación de los estados no alcanzables
2. Búsqueda de estados equivalentes
3. Construcción de un autómata a partir de los grupos de estados equivalentes

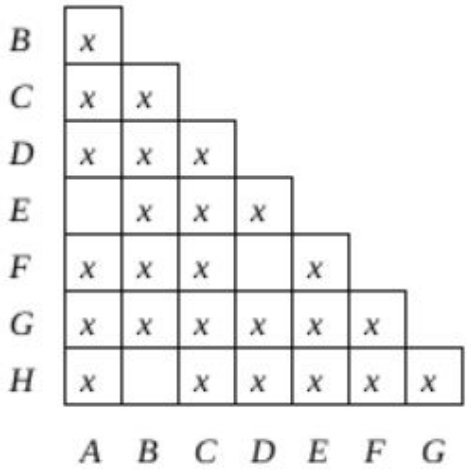
   EJEMPLO:

	0	1
-> A	B	A
B	A	C
C	D	B
* D	D	A
E	D	F
F	G	E
G	F	G
H	G	D

La idea es ir marcando dentro de una tabla qué pares de estados no son equivalentes. Cuando terminamos de ejecutar este algoritmo, las casillas no marcadas de la tabla nos indican los estados equivalentes.

primer paso: marcar como distinguibles los pares de estados finales/no finales.

paso iterativo: si r y s son dos estados ya marcados como distinguibles, y si para un símbolo de entrada a tenemos que $\delta(p, a) = r$ y $\delta(q, a) = s$, entonces marcamos p y q como distinguibles.



Perez Armas Fausto Isaac

NFA

Un NFA puede estar en varios estados a la vez o se puede ver que “adivina” a qué estado ir.

NFA Formal

Formalmente, un NFA es una quíntupla $A=(Q,S,d,q_0,F)$, donde todo es un **DFA**, pero

- $d(q,a)$ es un conjunto de estados en lugar de un solo estado. De hecho puede ser vacío, tener un solo estado o tener más estados.

Un NFA acepta, al igual que un DFA, lenguajes regulares

Por ejemplo, para el NFA que acepta cadenas que acaban en 01 su función de

	0	1
$\rightarrow q_0$	$\{q_0, q_1\}$	$\{q_0\}$
q_1	$\emptyset$	$\{q_2\}$
$\star q_2$	$\emptyset$	$\emptyset$

Como puede observarse, todo se especifica en conjuntos.

Lenguajes de un NFA

El lenguaje aceptado por un NFA, A, es: L(A)={w: $(q_0,w)\cap F\neq\emptyset$}

Equivalencia entre un DFA y un NFA

Un NFA es normalmente más fácil de definir, aunque al mismo tiempo, para cualquier NFA N existe un DFA D tal que L(D)=L(N) y viceversa.

Para esto se usa la construcción de subconjunto que muestra un ejemplo de cómo un autómata se puede construir a partir de otro.

Construcción de Subconjunto

1. Para cada **NFA** existe un **DFA** equivalente (acepta el mismo lenguaje).
2. Pero el **DFA** puede tener un número exponencial de estados.

Sea $N = (Q_N, S_N, d_N, q_0, F_N)$ un **NFA**. El DFA equivalente construido a partir del subconjunto de construcción es $D = (Q_D, S, d_D, \{q_0\}, F_D\}$, donde:

1. $|Q_D| = 2^{|QN|}$; i.e., Q_D es el conjunto de todos los subconjuntos de Q_N.
2. F_D es el conjunto de conjuntos S en Q_D tal que S Ç F_N ¹ Æ.
3. Para cualquier y ,

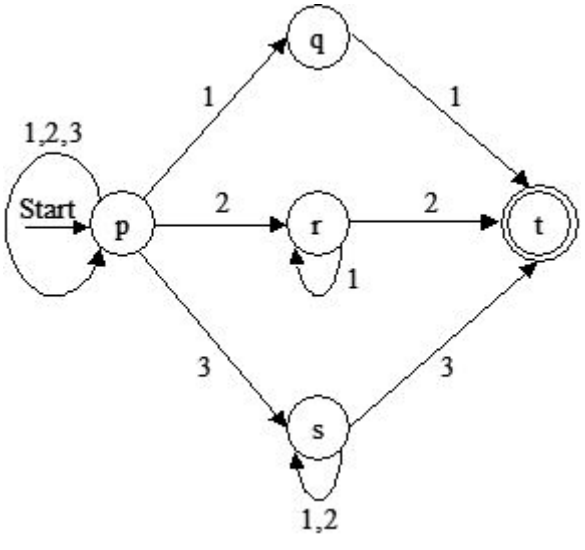
O sea, la unión de todos los estados a partir de con entrada
$d_D(\{q_1,q_2,...,q_k\},a) = d_N(p_1,a)$ È $d_N(p_2,a)$ È ... È $d_N(p_k,a)$.

La función de transición d_D del NFA anterior es:

	0	1
$\emptyset$	$\emptyset$	$\emptyset$
$\rightarrow \{q_0\}$	$\{q_0, q_1\}$	$\{q_0\}$
$\{q_1\}$	$\emptyset$	$\{q_2\}$
$\star\{q_2\}$	$\emptyset$	$\emptyset$
$\{q_0, q_1\}$	$\{q_0, q_1\}$	$\{q_0, q_2\}$
$\star\{q_0, q_2\}$	$\{q_0, q_1\}$	$\{q_0\}$
$\star\{q_1, q_2\}$	$\emptyset$	$\{q_2\}$
$\star\{q_0, q_1, q_2\}$	$\{q_0, q_1\}$	$\{q_0, q_2\}$

Al existir 3 estados, tenemos 8 subconjuntos. Esto mismo lo podemos poner como:

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



Ejemplo: Subconjunto de Construcción del NFA Previo

Un truco práctico importante utilizado por analizadores léxicos y otros procesadores de texto es ignorar los (frecuentemente muchos) estados que no son accesibles desde el estado de inicio (i.e., no hay ruta que lleve a ellos).

Para el ejemplo anterior de NFA, de los 32 subconjuntos posibles, solo 15 son accesibles. Calculando las transiciones "por demanda" obtenemos el siguiente d_D:

DFA

Con la propiedad de no-determinismo se agrega eficiencia al describir una aplicación

- Permite programar soluciones en un lenguaje de más alto nivel
- Hay un algoritmo para compilar un N-DFA en un DFA y poder ser ejecutado en una computadora convencional

Extensión del N-DFA para hacer saltos de un estado a otro espontáneamente

- Con la cadena vacía como entrada
- ÎN-DFA
- Estos autómatas también aceptan lenguajes regulares

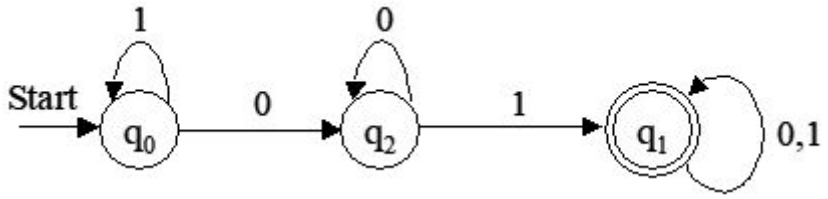
Ejemplo: Un Autómata *A* que acepta $L = \{x01y : x, y \in \{0,1\}^*\}$

- El DFA acepta cadenas que tienen 01 en alguna parte de la cadena
- El lenguaje del DFA es el conjunto de cadenas que acepta
- {w | w tiene la forma *x*01*y* para algunas cadenas *x* y *y* que consisten sólo de 0's y 1's}

El Autómata $A = (\{q_0, q_1, q_2\}, \{0,1\}, d, q_0, \{q_1\})$

Autómata representado con una tabla de transiciones:

Autómata representado con un diagrama de transiciones:



Teorema de Myhill-Nerode

En informática teórica, y en particular en teoría de lenguajes formales y autómatas, el teorema de Myhill-Nerode da una condición necesaria y suficiente para que un lenguaje formal sea un lenguaje racional, es decir, reconocible por un autómata finito. . Este teorema lleva el nombre de John Myhill y Anil Nerode, quienes lo demostraron en 1958 (Nerode 1958).
El interés de esta afirmación es que la condición necesaria y suficiente se relaciona con el lenguaje mismo y no requiere la construcción de un autómata. La prueba, por otra parte, es constructiva y permite obtener de forma eficaz un autómata que además resulta mínimo.
Si L es un lenguaje regular, entonces, por definición, hay un DFA A que lo reconoce, con solo un número finito de estados. Si hay n estados, entonces particione el conjunto de todas las cadenas finitas en n subconjuntos, donde el subconjunto S i es el conjunto de cadenas que, cuando se dan como entrada al autómata A , hacen que termine en el estado i . Por cada dos cadenas X y Y que pertenecen al mismo subconjunto, y para cada opción de una tercera cadena z , autómata A alcanza el mismo estado en la entrada XZ , ya que llega en la entrada yz , y por lo tanto debe aceptar ya sea tanto de las entradas xz e yz o rechazar ambos. Por lo tanto, ninguna cadena z puede ser una extensión distintiva de x y y , así que deben estar relacionados por R L . Por lo tanto, s i es un subconjunto de una clase de equivalencia de R L . Combinando este hecho con el hecho de que cada miembro de una de estas clases de equivalencia pertenece a uno de los conjuntos S i , esto da una relación de muchos a uno de los estados de A a las clases de equivalencia, lo que implica que el número de clases de equivalencia es finito y como máximo n .
En la otra dirección, suponga que R L tiene un número finito de clases de equivalencia. En este caso, es posible diseñar un autómata finito determinista que tenga un estado para cada clase de equivalencia. El estado de inicio del autómata corresponde a la clase de equivalencia que contiene la cadena vacía, y la función de transición de un estado X en el símbolo de entrada y lleva al autómata a un nuevo estado, el estado correspondiente a la clase de equivalencia que contiene la cadena xy , donde x es una cadena arbitrariamente elegido en la clase de equivalencia de X . La definición de la relación Myhill-Nerode implica que la función de transición está bien definida: no importa qué cadena representativa x se elija para el estado X , se obtendrá el mismo valor de función de transición. Un estado de este autómata es aceptable si la clase de equivalencia correspondiente contiene una cadena en L ; en este caso, nuevamente, la definición de la relación implica que todas las cadenas de la misma clase de equivalencia también deben pertenecer a L , porque de lo contrario la cadena vacía sería una cadena distintiva para algunos pares de cadenas de la clase. Así, la existencia de un autómata finito que reconoce L implica que la relación Myhill-Nerode tiene un número finito de clases de equivalencia, a lo

sumo igual al número de estados del autómata, y la existencia de un número finito de clases de equivalencia implica la existencia de un autómata con tantos estados.

MINIMIZACIÓN

Minimización de autómatas

Construcción de un AFDt con un número de estados mínimo que sea equivalente a un AFDt dado.

Definiciones previas:

- Estados accesibles: q0 es accesible
  q accesible $\Rightarrow \forall s \in \Sigma, \delta(q, s)$ es accesible
- Estados k-equivalentes o k-indistinguibles: $p \equiv_k q$

  $\forall x \in \Sigma^{\leq k} (\delta^*(p, x) \in F \leftrightarrow \delta^*(q, x) \in F)$
- Estados equivalentes o indistinguibles: $p \equiv q$

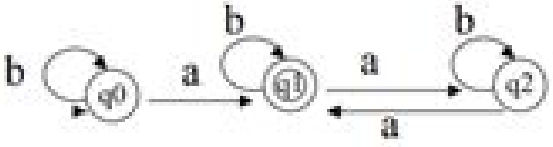
  $\forall k \; \forall x \in \Sigma^{\leq k} (\delta^*(p, x) \in F \leftrightarrow \delta^*(q, x) \in F)$, es decir,
  $\forall x \in \Sigma^* (\delta^*(p, x) \in F \leftrightarrow \delta^*(q, x) \in F)$

Minimización de autómatas

Construcción del AFDt **mínimo N** a partir del AFDt **M = (Q, Σ, δ, q_0, F)**

1) Eliminar estados inaccesibles
2) Determinar las clases de estados equivalentes:
   $p \equiv_0 q \Leftrightarrow (p \in F \leftrightarrow q \in F)$
   $p \equiv_{k+1} q \Leftrightarrow (p \equiv_k q \wedge \forall s \in \Sigma \; \delta(p, s) \equiv_k \delta(q, s))$
3) Construcción del AFD **N = (P, Σ, γ, p_0, G)** con
   P = Q/≡ siendo ≡ es la menor $\equiv_k$ tal que $\equiv_k$ coincide con $\equiv_{k+1}$
   $p_0 = [q_0]$
   $\gamma([p], s) = [\delta(p, s)]$
   $G = \{[p]: p \in F\}$

Ejemplo 1



Etapa 0 : $q0 \equiv_0 q2$ (ambos $\in F$), pero q0 y q1 no son equivalentes
Clases a nivel 0: [q0,q2] [q1]

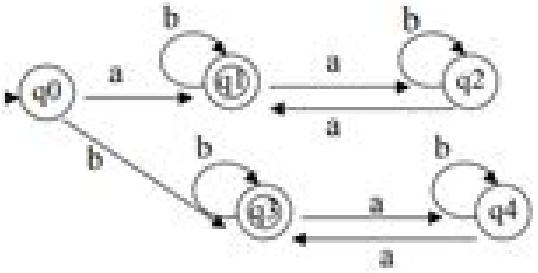
Etapa 1 : $q0 \equiv_1 q2$ porque $\delta(q0, a) \equiv_0 \delta(q2, a)$ y $\delta(q0, b) \equiv_0 \delta(q2, b)$

Clases a nivel 1: [q0,q2] [q1]

Por tanto $\equiv = \equiv_0$

AFD minimo:

Ejemplo 2



q0,q2	a	q1
	b	q3, q2 **no** equiv.
q0,q4	a	q1,q3
	b	q3, q4 **no** equiv.
q2,q4	a	q1,q3
	b	q2,q4
q1,q3	a	q2,q4
	b	q1,q3

Etapa 0 : [q0,q2,q4] [q1,q3]
Etapa 1 : [q0] [q2,q4] [q1,q3]
Etapa 2 : [q0] [q2,q4] [q1,q3]

Por tanto $\equiv = \equiv_1$

AFD mínimo :