Automatas Finitos

Propedéutico de Ciencias Computacionales

Plan

Section 1

Introducción a los Autómatas

Introducción a los Autómatas

Autómata: Conjunto de estados + Control $\rightarrow$ Cambio de estados en respuesta a una entrada.
Tipo de Control:

- Determinístico: Para cada entrada, hay sólo un estado al que el autómata puede ir desde el estado en que se encuentre.
- No determinístico: Un autómata finito es no-determinístico cuando se permite que el **AF** tenga 0 o más estados siguientes para cada par estado-entrada.

Autómatas No-Deterministas

Si añadimos la propiedad de no-determinismo, no añadimos poder al autómata. Osea que no podemos definir ningún lenguaje que no se pueda definir con el autómata determinístico.
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

Autómatas No-Deterministas

- Extensión del N-DFA para hacer saltos de un estado a otro espontáneamente, con la cadena vacía (ϵ) como entrada: ϵN-DFA. Estos autómatas también aceptan lenguajes regulares.

Section 2

Definición formal de un Autómata Finito Determinístico

Definición Formal de un Autómata

Un AF se representa como una 5-tupla: $A = (Q, \Sigma, \delta, q_0, F)$.
Donde:

1. Q: Un conjunto finito de *estados*.
2. Σ: Un conjunto finito de *símbolos de entrada* (alfabeto)
3. q_0: El estado *inicial/de comienzo*.
4. F: cero o más *estados finales/de aceptación*.

Función de Transición

5. δ: *Función de transición*. Esta función:
   - Toma un estado y un símbolo de entrada como argumentos.
   - Regresa un estado.
   - Una "regla" de δ se escribe como $\delta(q, a) = p$, donde q y p son estados y a es un símbolo de entrada.
   - Intuitivamente: Si el **AF** está en un estado q, y recibe una entrada a, entonces el **AF** va al estado p (nota: puede ser al mismo estado $q = p$).

DFA

- El *lenguaje* de un DFA es el conjunto de todas las cadenas que el DFA acepta
- Dada una cadena (e.g., $s_1, s_2, \ldots, s_n$) el DFA empieza en su estado inicial (e.g., q_0), consulta si existe una transición de q_0 con el primer símbolo (s_1) a otro estado (e.g., q_1) y si existe (i.e., $\delta(q_0, s_1) = q_1$) se mueve al estado descrito en la transición.
- Procesa el siguiente símbolo de la cadena (i.e., s_2) y así continua.
- Si logra procesar toda la cadena y el estado al que llega es uno de los estados finales, entonces se dice que el autómata acepta esa cadena.

Ejemplo

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

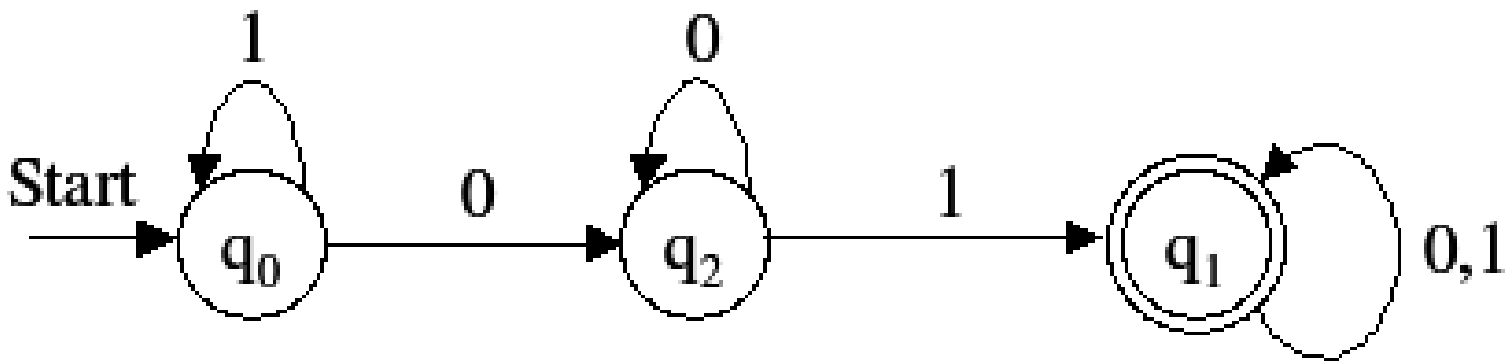
- El DFA acepta cadenas que tienen 01 en alguna parte de la cadena
- El lenguaje del DFA es el conjunto de cadenas que acepta $\{w | w$ tiene la forma "$x01y$" para algunas cadenas x y y que consisten sólo de 0's y 1's $\}$.

Tabla de Transiciones

El autómata anterior puede ser representado con una tabla de transiciones, definido como $A = (\{q_0, q_1, q_2\}, \{0, 1\}, \delta, q_0, q_1)$, de la siguiente forma:

	0	1
$\rightarrow q_0$	q_2	q_0
$\star q_1$	q_1	q_1
q_2	q_2	q_1

Diagrama de Transiciones



- Cada estado tiene un nodo asociado
- Cada transición de estados tiene un arco asociado etiquetado con el/los símbolos correspondientes
- Existe una etiqueta de inicio para el estado inicial y un doble círculo asociado a los estados finales

Convenciones

Por convención se utilizan:

- a, b, etc., o dígitos para los símbolos de entrada
- $u, v, \ldots, z$ para las cadenas de símbolos de entrada
- q, p, etc., para los estados

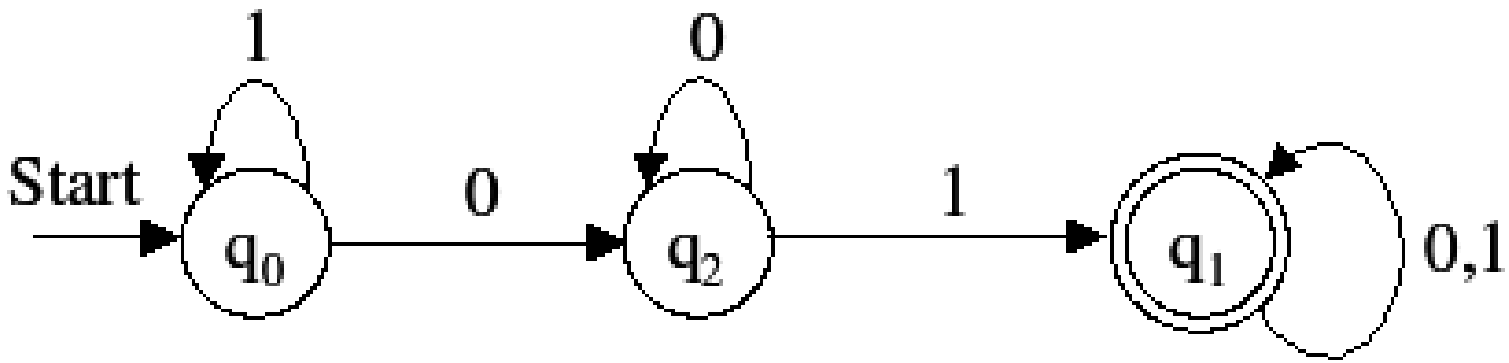
Funciones de transición extendidas ($\hat{\delta}$)

Intuitivamente, un **FA** acepta una cadena $w = a_1 a_2 \ldots a_n$ si hay una ruta en el diagrama de transiciones que:

1. Empieza en el estado de inicio,
2. Termina en un estado de aceptación, y
3. Tiene una secuencia de etiquetas $a_1, a_2, \ldots, a_n$.

Ejemplo

Ejemplo: El siguiente **AF** acepta la cadena 01101:



Función de Transición

Formalmente, podemos extendemos la función de transición δ a $\hat{\delta}(q, w)$, donde w puede ser cualquier cadena de símbolos de entrada:

- Base: $\hat{\delta}(q, \epsilon) = q$ (i.e., nos quedamos en el mismo lugar si no recibimos una entrada).
- Inducción: $\hat{\delta}(q, w) = \delta(\hat{\delta}(q, x), a)$, donde x es una cadena, y a es un solo símbolo (i.e., ver a dónde va el AF con x, luego buscar la transición para el último símbolo a partir de ese estado).
- Si $\hat{\delta}(q, x) = p$ entonces $\hat{\delta}(q, w) = \delta(p, a)$
- $\hat{\delta}$ representa rutas.

Función de Transición

- **Aceptación de Cadenas**: Un **AF** $A = (Q, \Sigma, \delta, q_0, F)$ acepta la cadena w si $\hat{\delta}(q_0, w)$ está en F.
- **Lenguaje de un AF**: Un **AF** acepta el lenguaje $L(A) = \{w | \hat{\delta}(q_0, w) \in F\}$.

Ejemplo:

Un DFA que acepta todas y sólo las cadenas que tienen un número par de 0's y también un número par de 1's

Ejemplo (cont.)

Ejemplo (cont.)

Representación tabular del autómata anterior:

		0	1
$\star \rightarrow$	q_0	q_2	q_1
	q_1	q_3	q_0
	q_2	q_0	q_3
	q_3	q_1	q_2

Problema 2.2.1

Cada vez que entra una canica en alguno de las dos entradas (A o B) cae y mueve la compuerta. La salida D es éxito y la D fracaso. Modelarlo como un DFA.

Problema 2.2.4

Diseñe un DFA que acepte:

- Todas las cadenas que acaban en 00
- Todas las cadenas con 000 en algún lugar
- Todas las cadenas con una sucadena 001

Section 3

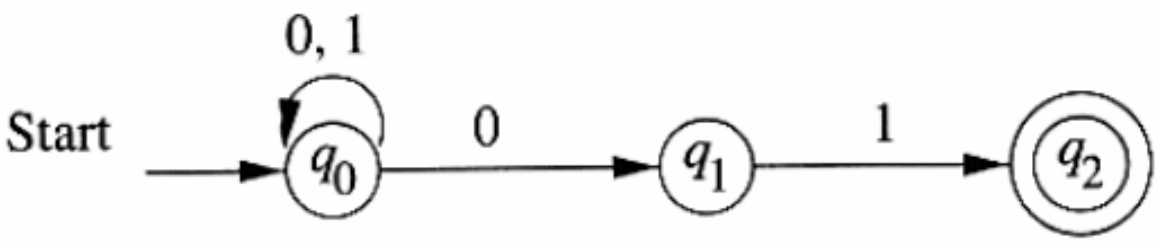
Autómata Finito No-Determinístico

Autómata Finito No-Determinístico

Un autómata finito es no-determinístico cuando se permite que el AF tenga 0 o más estados siguientes para cada par estado-entrada:

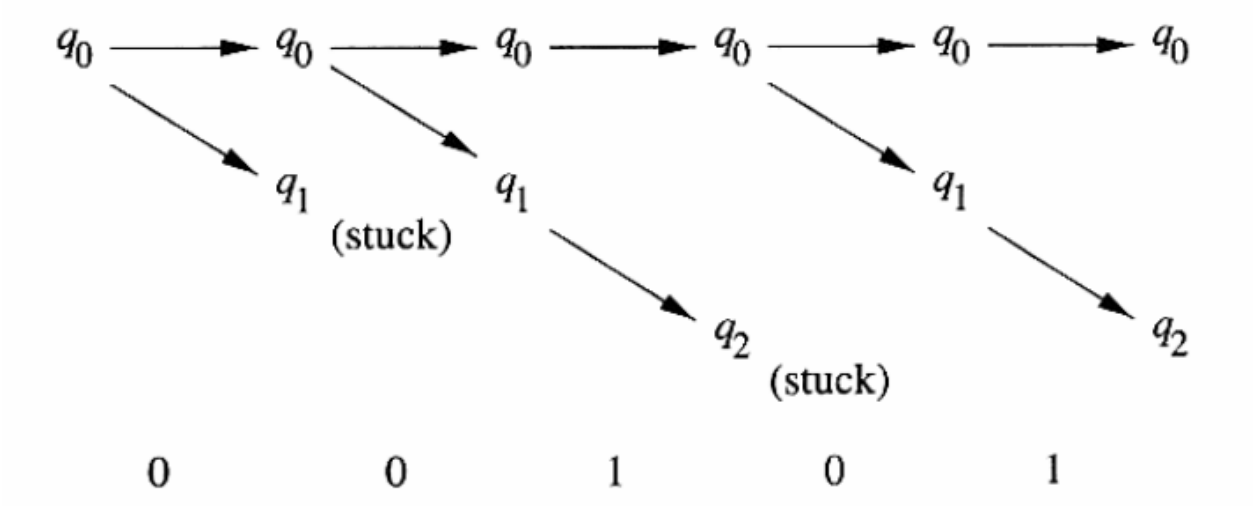
Autómata Finito No-Determinístico

- En el ejemplo anterior, se puede apreciar que de q_0 se puede ir a q_0 ó a q_1 con la entrada "0", y esto hace al AF ser no-determinista.
- Un NFA puede estar en varios estados a la vez o se puede ver que "adivina" a qué estado ir.
- Por ejemplo, el siguiente autómata acepta todas las cadenas que terminan en 01:



Autómata Finito No-Determinístico

Lo que pasa al procesar como entrada a 00101 es:



Autómata Finito No-Determinístico

- Un NFA es una herramienta importante para diseñar procesadores de cadenas, e.g., *grep*, analizadores léxicos, etc. Es fácil diseñar NFAs que encuentren secuencias de palabras en texto.
- **NFA**: Formalmente, un NFA es una quíntupla $A = (Q, \Sigma, \delta, q_0, F)$, donde todo es un DFA, pero $\delta(q, a)$ nos regresa 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

Ejemplo

Por ejemplo, para el NFA que acepta cadenas que acaban en 01 su función de transición δ es:

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

Como puede observarse, todo se especifica en conjuntos.

Extensión a $\hat{\delta}$

Similarmente a un DFA, podemos definir la función de transición extendida $\hat{\delta}$ como sigue:

- Base: $\hat{\delta}(q, \epsilon) = q$
- Inducción: Supongamos w es de la forma $w = xa$, donde a es el símbolo terminal y x es el resto de w. Supongamos también que: $\hat{\delta}(q, x) = \{p_1, p_2, \ldots, p_k\}$.
  $\bigcup_{i=1}^{k} \delta(p_i, a) = \{r_1, r_2, \ldots, r_m\}$.
  Entonces $\cup\delta(q, w) = \{r_1, r_2, \ldots, r_m\}$.
- En otras palabras calculamos $\hat{\delta}(q, w)$ primero calculando $\hat{\delta}(q, x)$ y después siguiendo cualquier transición de algunos de esos estados etiquetada con a.

Ejemplo:

Por ejemplo, podemos calcular $\hat{\delta}(q_0, 00101)$ para el autómata anterior:

$\hat{\delta}(q_0, \epsilon) = \{q_0\}$

$\hat{\delta}(q_0, 0) = \delta(q_0, 0) = \{q_0, q_1\}$

$\hat{\delta}(q_0, 00) = \delta(q_0, 0) \cup \delta(q_1, 0) \cup \emptyset = \{q_0, q_1\}$

$\hat{\delta}(q_0, 001) = \delta(q_0, 1) \cup \delta(q_1, 1) = \{q_0\} \cup \{q_2\} = \{q_0, q_2\}$

$\hat{\delta}(q_0, 0010) = \delta(q_0, 0) \cup \delta(q_2, 0) = \{q_0, q_1\} \cup \emptyset = \{q_0, q_1\}$

$\hat{\delta}(q_0, 00101) = \delta(q_0, 1) \cup \delta(q_1, 1) = \{q_0\} \cup \{q_2\} = \{q_0, q_2\}$

Que tiene un estado final.

Lenguajes de un NFA

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

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

Para cada NFA existe un DFA equivalente (acepta el mismo lenguaje). Pero el DFA puede tener un número exponencial de estados.
Sea $N = (Q_N, \Sigma, \delta_N, q_0, F_N)$ un NFA. El DFA equivalente construido a partir del subconjunto de construcción es $D = (Q_D, \Sigma, \delta_D, \{q_0\}, F_D)$, donde:

- $|Q_D| = 2^{|Q_N|}$; i.e., Q_D es el conjunto de todos los subconjuntos de Q_N.
- F_D es el conjunto de conjuntos S en Q_D tal que $S \cap F_N \neq \emptyset$.
- Para cualquier $S \subseteq Q_N$ y $a \in \Sigma$, $\delta_D(S, a) = \bigcup_{p \in S} \delta_N(p, a)$, osea, la unión de todos los estados a partir de p con entrada a. $\delta_D(\{q_1, q_2, \ldots, q_k\}, a) = \delta_N(p_1, a) \cup \delta_N(p_2, a) \cup \ldots \cup \delta_N(p_k, a)$.

Construcción de Subconjunto

La función de transición δ_D del NFA anterior es:

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

Construcción de Subconjunto

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
$\star C$	A	D
$\star H$	E	F

Construcción de Subconjunto

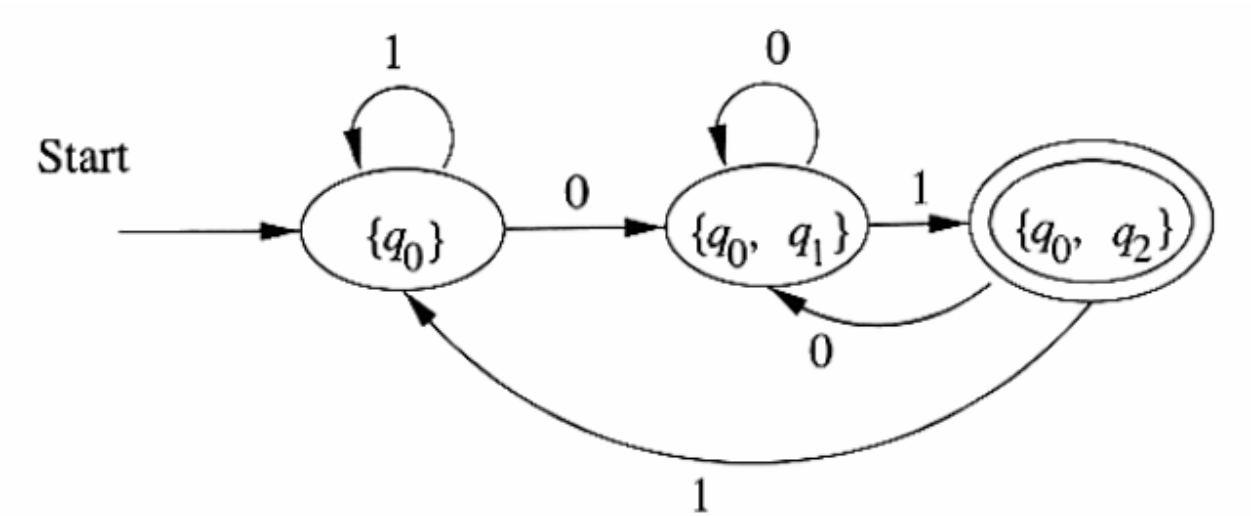
- Lo cual es un DFA (simplemente cambiando la notación). También es importante notar que no todos los estados pueden ser alcanzados. En particular, sólo los estados B, E y F son accesibles, por lo que los demás los podemos eliminar.
- Una forma de no construir todos los subconjuntos para después encontrar que sólo unos cuantos son accesibles, es construir la tabla sólo para los estados accesibles (*lazy evaluation*).

Ejemplo (cont.)

Para el ejemplo anterior: $\delta_D(\{q_0\}, 0) = \{q_0, q_1\}$
$\delta_D(\{q_0\}, 1) = \{q_1\}$
$\delta_D(\{q_0, q_1\}, 0) = \{q_0, q_1\}$
$\delta_D(\{q_0, q_1\}, 1) = \{q_0, q_2\} = \delta_N(q_0, 1) \cup \delta_N(q_1, 1)$
$\delta_D(\{q_0, q_2\}, 0) = \{q_0, q_1\}$
$\delta_D(\{q_0, q_2\}, 1) = \{q_0\}$

Ejemplo (cont.)

Lo que nos queda:



Prueba de Equivalencia

Teorema clave: inducción de $|w|$ (la prueba está en el libro): $\hat{\delta}_D(\{q_0\}, w) = \hat{\delta}_N(q_0, w)$. Lo que queremos probar es que si $D = (Q_D, \Sigma, \delta_D, \{q_0\}, F_D)$ es construido a partir del NFA $N = (Q_N, \Sigma, \delta_N, q_0, F_N)$ usando construcción de subconjuntos, entonces $L(D) = L(N)$.

Prueba de Equivalencia

- Queremos probar por inducción en w que $\hat{\delta}_D(\{q_0\}, w) = \hat{\delta}_N(q_0, w)$.
- Las dos funciones de transición regresan conjuntos de estados de Q_N, pero la determinística lo interpreta como uno solo de sus estados Q_D.
- *Base*: $w = \epsilon$, en este caso $\hat{\delta}_D(\{q_0\}, \epsilon) = \hat{\delta}_N(q_0, \epsilon) = \{q_0\}$.

Prueba de Equivalencia

- *Inducción*: Tomamos w de longitud $n+1$ y suponemos que se cumple el enunciado para n, o sea que $\hat{\delta}_D(\{q_0\}, x) = \hat{\delta}_N(q_0, x)$.
- Sean estos dos conjuntos de estados $= \{p_1, p_2, \ldots, p_k\}$.
- Dividimos a w en xa. La definición de $\hat{\delta}$ para el NFA nos dice que: $\hat{\delta}_N(q_0, w) = \bigcup_{i=1}^{k} \delta_N(p_i, a)$.
- Por la construcción de subconjuntos: $\delta_D(\{p_1, p_2, \ldots, p_k\}, a) = \bigcup_{i=1}^{k} \delta_N(p_i, a)$

Prueba de Equivalencia

- Usando esto, y que $\hat{\delta}_D(\{q_0\}, x) = \{p_1, p_2, \ldots, p_k\}$ tenemos que:
  $\hat{\delta}_D(\{q_0\}, w) = \delta_D(\hat{\delta}_D(\{q_0\}, x), a) = \delta_D(\{p_1, p_2, \ldots, p_k\}, a) = \bigcup_{i=1}^{k} \delta_N(p_i, a)$
- Tanto D como N aceptan w cuando contiene un estado en FN.
- Consecuencia: $L(D) = L(N)$.

Ejemplos

Ejemplo: Convertir el siguiente NFA a un DFA

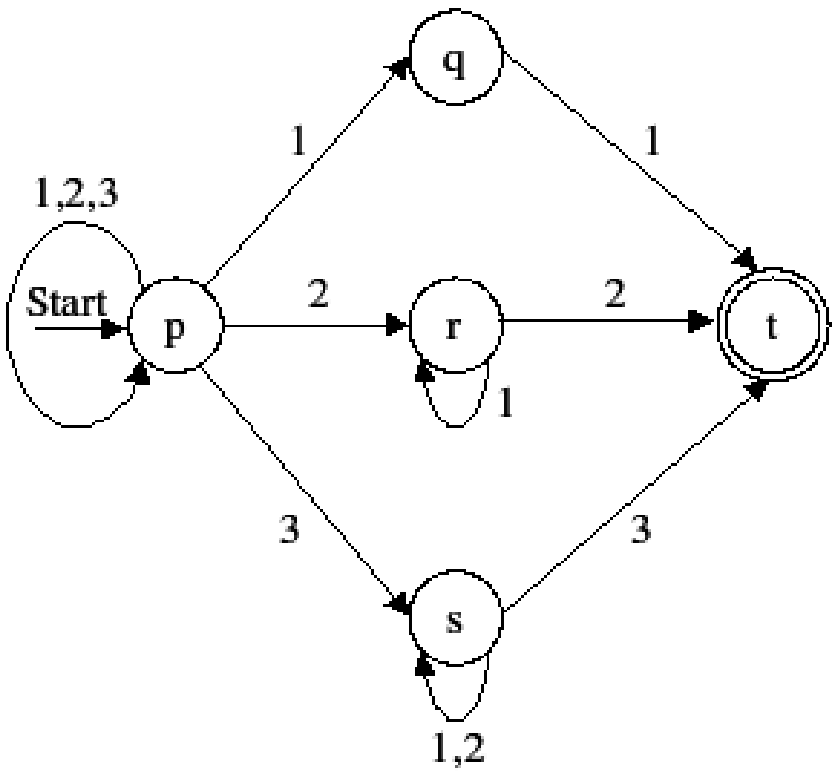
	0	1
$\rightarrow p$	$\{p, q\}$	$\{p\}$
q	$\{r\}$	$\{r\}$
r	$\{s\}$	$\emptyset$
$*s$	$\{s\}$	$\{s\}$

Ejemplo

En este ejemplo un tanto imaginario, se diseñará un **NFA** para aceptar cadenas sobre el alfabeto $\{1, 2, 3\}$ de tal manera que el último símbolo aparezca previamente, sin ninguna intervención de un símbolo más alto entre esa previa aparición del símbolo, e.g., $\ldots 11, \ldots 21112, \ldots 312123$.

- Truco: Utilizar el estado de inicio con el significado "Creo que todavía no se ha visto el símbolo que corresponde al símbolo final".
- Otros tres estados representando una elección de que el símbolo con que acaba la cadena se ha visto y se recuerda de que símbolo se trata.

Ejemplo



Ejemplo (cont.)

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 δ_D:

Ejemplo (cont.)

	1	2	3
$\rightarrow \{p\}$	$\{pq\}$	$\{pr\}$	$\{ps\}$
$\{pq\}$	$\{pqt\}$	$\{pr\}$	$\{ps\}$
$\{*pqt\}$	$\{pqt\}$	$\{pr\}$	$\{ps\}$
$\{pr\}$	$\{pqr\}$	$\{prt\}$	$\{ps\}$
$\{*prt\}$	$\{pqr\}$	$\{prt\}$	$\{ps\}$
$\{ps\}$	$\{pqs\}$	$\{prs\}$	$\{pst\}$
$\{*pst\}$	$\{pqs\}$	$\{prs\}$	$\{pst\}$
$\{prs\}$	$\{pqrs\}$	$\{prst\}$	$\{pst\}$
$\{*prst\}$	$\{pqrs\}$	$\{prst\}$	$\{pst\}$
$\{pqs\}$	$\{pqst\}$	$\{prs\}$	$\{pst\}$
$\{*pqst\}$	$\{pqst\}$	$\{prs\}$	$\{pst\}$
$\{pqr\}$	$\{pqrt\}$	$\{prt\}$	$\{ps\}$
$\{*pqrt\}$	$\{pqrt\}$	$\{prt\}$	$\{ps\}$
$\{pqrs\}$	$\{pqrst\}$	$\{prst\}$	$\{pst\}$
$\{*pqrst\}$	$\{pqrst\}$	$\{prst\}$	$\{pst\}$

Ejemplo

Ejemplo: Diseñe un NFA que reconozca los siguientes conjuntos de cadenas: *abc*, *abd* y *aacd*, suponiendo que el alfabeto es $\{a, b, c, d\}$

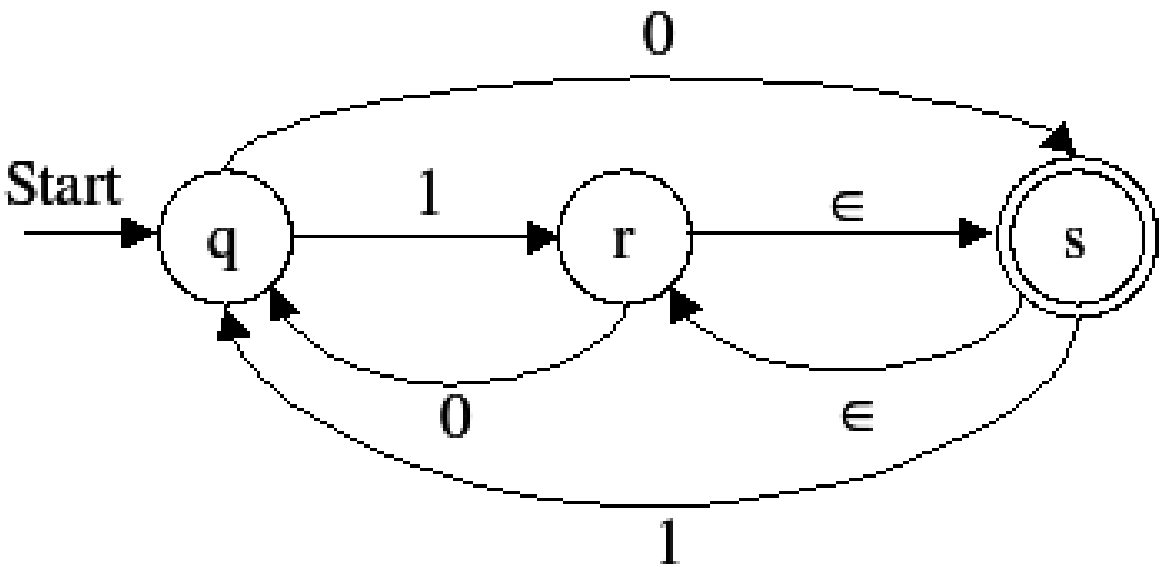
Section 4

Autómatas Finitos y Lenguajes Formales

Autómata Finito con Transiciones-ϵ

- Sea ϵ una etiqueta en arcos.
- No hay ningún cambio extra: la aceptación de w todavía se da como la existencia de la ruta desde un estado de inicio a un estado de aceptación con etiqueta w.
- Pero ϵ puede aparecer en los arcos, y significa que puede existit una transición con una cadena vacía (i.e., no tiene una contribución visible para w).
- Se les denota $\epsilon-$NFA.

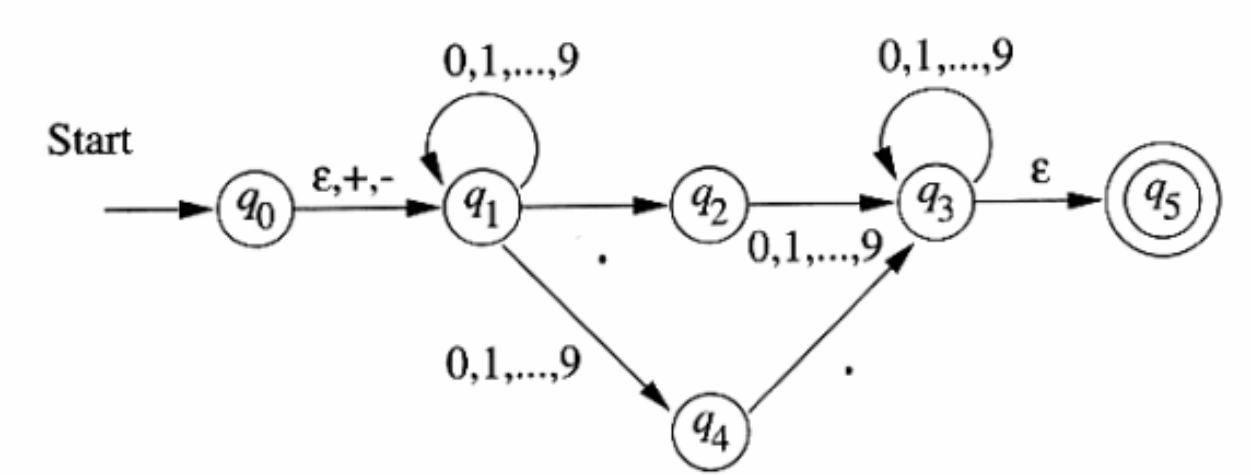
Ejemplo



Ejemplo (cont.)

- 001 es aceptado siguiendo la ruta q, s, r, q, r, s, con la etiqueta $0\epsilon01\epsilon = 001$.
- Podemos diseñar un autómata que acepte cadenas de números que tengan un signo al inicio opcional, seguido posiblemente de una cadena de decimales, seguido de un punto decimal y posiblemente de otra cadena de decimales.

Ejemplo



ϵ-NFA

- Más formalmente: Un ϵ-NFA es una quíntupla $(Q, \Sigma, \delta, q_0, F)$, donde δ es una función de $Q \times \Sigma \cup \{\epsilon\}$ al conjunto potencia de Q.
- Hay que evitar que ϵ sea parte del alfabeto Σ

Tabla de Transición

La tabla de transición del ϵ-NFA del ejemplo anterior es:

	ϵ	$+, -$	$.$	$0, 1, \ldots, 9$
q_0	$\{q_1\}$	$\{q_1\}$	$\emptyset$	$\emptyset$
q_1	$\emptyset$	$\emptyset$	$\{q_2\}$	$\{q_1, q_4\}$
q_2	$\emptyset$	$\emptyset$	$\emptyset$	$\{q_3\}$
q_3	$\{q_5\}$	$\emptyset$	$\emptyset$	$\{q_3\}$
q_4	$\emptyset$	$\emptyset$	$\{q_3\}$	$\emptyset$
q_5	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$

Section 5

Eliminación de las Transiciones-ϵ

Eliminación de las Transiciones-ϵ

Las transiciones-ϵ son una conveniencia, pero no incrementan la potencia de los FA's. Para eliminar las transiciones-ϵ:

1. Calcular la cerradura transitiva sólo para los arcos ϵ. Del ejemplo anterior: $q \longrightarrow \{q\}; r \longrightarrow \{r, s\}; s \longrightarrow \{r, s\}$.
2. Si un estado p puede alcanzar al estado q por medio de arcos ϵ, y existe una transición de q a r en la entrada a (no ϵ), entonces añádase una transición de p a r con la entrada a.
3. Convertir el estado p en un estado de aceptación siempre y cuando p pueda alcanzar algún estado de aceptación q por medio de arcos ϵ.
4. Eliminar todas las transiciones-ϵ.

Ejemplo

Transiciones Extendidas

De la misma forma como lo hicimos anteriormente, podemos definir las transiciones extendidas para ϵ-NFA.

- *Base*: $\hat{\delta}(q, \epsilon) = ECLOSE(q)$
- *Inducción*: $\hat{\delta}(q, xa) = \bigcup_{p \in \delta(\hat{\delta}(q,x),a)} ECLOSE(p)$

  Por ejemplo, $\hat{\delta}(q_0, 5.6)$ es: $\hat{\delta}(q_0, \epsilon) = \{q_0, q_1\} = ECLOSE(q_0)$
  $\delta(q_0, 5) \cup \delta(q_1, 5) = \{q_1, q_4\}$
  $ECLOSE(q_1) \cup ECLOSE(q_4) = \{q_1, q_4\}$
  $\delta(q_1, .) \cup \delta(q_4, .) = \{q_2, q_3\}$
  $ECLOSE(q_2) \cup ECLOSE(q_3) = \{q_2, q_3, q_5\} = \hat{\delta}(q_0, 5.6)$
  $\delta(q_2, 6) \cup \delta(q_3, 6) \cup \delta(q_5, 6) = \{q_3\}$
  $\hat{\delta}(q_0, 5.6) = ECLOSE(q_3) = \{q_3, q_5\}$

Equivalencias

- Como antes, el lenguaje aceptado por un ϵ-NFA, E, es: $L(E) = \{w|\hat{\delta}(q_0, w) \cap F \neq \emptyset\}$, osea todas las cadenas w que van de un estado inicial q_0 a al menos un estado final.
- Se puede demostrar que se puede construir un DFA a partir de un ϵ-NFA siguiendo un esquema parecido al de construcción de subconjuntos visto para NFA.

Equivalencias

- $Q_D = \{S | S \subseteq Q_E \wedge S = ECLOSE(S)\}$
- $q_D = ECLOSE(q_0)$
- $F_D = \{S | S \subseteq Q_D \wedge S \cap F_E \neq \emptyset\}$
- $\delta_D(S, a) = \{\bigcup ECLOSE(p) | p \in \delta(t, a), t \in S\}$
- Lo que se calcula para todos los $a \in \Sigma$ y conjuntos $S \in Q_D$.

Equivalencias

- Se puede demostrar que un lenguaje L es aceptado por algún ϵ-NFA E si y solo si L es aceptado por un DFA.
- Para esto, hacia un sentido es fácil (cambiando un DFA a un ϵ-NFA)
- Para el otro sentido se hace lo mismo que hemos hecho antes, probando con el caso base y el caso inductivo, donde partimos de $w = xa$, suponemos que es verdad para x y probamos para w, solo que ahora tomando la cerradura ϵ o ECLOSE para los estados.

Ejemplo

Considere el siguiente ϵ-NFA:

	ϵ	a	b	c
$\rightarrow p$	$\emptyset$	$\{p\}$	$\{q\}$	$\{r\}$
q	$\{p\}$	$\{q\}$	$\{r\}$	$\emptyset$
$*r$	$\{q\}$	$\{r\}$	$\emptyset$	$\{p\}$

a) Calcula las ϵ-closure para cada estado

b) Cuales son todas las cadenas de 3 o menos caracteres aceptadas por el autómata

c) Convertirlo a un DFA

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.