Ciencias de la Computación I

Gramáticas Regulares

Expresiones Regulares

Gramáticas

- Intuitivamente una gramática es un conjunto de reglas para formar correctamente las frases de un lenguaje.

- Por ejemplo, gramática del castellano (o de cualquier idioma) nos permite:

- **Identificar cuándo una frase es sintácticamente correcta**

  "Juan es un buen estudiante" **"Es un Juan estudiante buen"**

- **Generar todas las posibles frases sintácticamente correctas**

En esta materia estudiaremos

Gramáticas Formales → GENERADORAS de Lenguajes Formales

Gramáticas Formales

Una gramática formal es una cuadrupla G = (N, T, P, S)

N = conjunto finito de símbolos no terminales

T = conjunto finito de símbolos terminales

$N \cap T = \emptyset$

S = símbolo distinguido o axioma $S \notin (N \cup T)$

P = conjunto finito de reglas de producción **(permiten generar cadenas a partir de S)**

$\alpha \rightarrow \beta$

$\alpha = \varphi A \rho$ $\quad A \in N \cup \{S\}$

$\beta = \varphi \omega \rho$ $\quad \varphi, \omega, \rho \in (N \cup T)^*$

De acuerdo al formato de las reglas se pueden definir 4 tipos de gramáticas y sus correspondientes lenguajes

Gramáticas Regulares

Reglas de producción

1) $S \longrightarrow aA$
2) $A \longrightarrow aA$
3) $A \longrightarrow b$

- S es el símbolo distinguido a partir del cual se comienza a generar
- S se reemplaza por aA ó
- A se reemplaza por aA ó
- A se reemplaza por b
- a, b son símbolos del Alfabeto

Derivaciones

$S \Rightarrow aA \Rightarrow ab$ $\quad ab \in L$

$S \Rightarrow aA \Rightarrow aaA \Rightarrow aab$ $\quad aab \in L$

$S \Rightarrow aA \Rightarrow aaA \Rightarrow aaaA \Rightarrow aaab$ $\quad aaab \in L$

Se pueden generar infinitas cadenas $\quad \ldots..$

$L = \{a^n b / n > 0\}$

Gramáticas Regulares (Tipo 3)

- **Generan los lenguajes regulares (reconocidos por Autómatas Finitos)**
- **Se definen como una cuadrupla G = <N, T, P, S>**

N = conjunto finito de símbolos no terminales

T = conjunto finito de símbolos terminales

S = símbolo distinguido o axioma $S \notin (N \cup T)$

P = conjunto finito de reglas de producción

Formato reglas de producción de Gramáticas Regulares

Lineal a derecha	**Lineal a izquierda**
$A \rightarrow aB$	$A \rightarrow Ba$
$A \rightarrow a$	$A \rightarrow a$
$S \rightarrow \varepsilon$ (para generar la cadena vacía)	$S \rightarrow \varepsilon$ (para generar la cadena vacía)
$A \in N \cup \{S\}$ $a \in T$ $B \in N$	$A \in N \cup \{S\}$ $a \in T$ $B \in N$

Gramáticas Regulares (Tipo 3)

Ejemplo 1:

Sea $G_1 = (\{A\}, \{a, b\}, S, P_1)$ donde $P_1 = \{ S \rightarrow aA, A \rightarrow aA, A \rightarrow b \}$

G_1 es una gramática regular lineal a derecha que genera el lenguaje:

$L = \{a^n b / n > 0 \}$

Ejemplo 2

Sea $G_2 = (\{A\}, \{a, b\}, S, P_2)$ donde $P_2 = \{ S \rightarrow Ab,$

$A \rightarrow Aa,$

$A \rightarrow a \}$

Derivación

$S \Rightarrow Ab \Rightarrow ab$ $\quad ab \in L$

$S \Rightarrow Ab \Rightarrow Aab \Rightarrow aab$ $\quad aab \in L$

$S \Rightarrow Ab \Rightarrow Aab \Rightarrow Aaab \Rightarrow aaab$ $\quad aaab \in L$

Se pueden generar infinitas cadenas .....

G_2 genera $L = \{a^n b / n > 0\}$ $\quad L(G_2) = L(G_1)$

Gramáticas Regulares (Tipo 3)

Derivación inmediata (lineal a derecha):

$\omega \Rightarrow \beta$ La cadena **β** se obtiene de la cadena **ω** en un paso usando las reglas de P. Si $\omega = \alpha A$ y $\beta = \alpha\delta$ entonces:

$\alpha A \Rightarrow \alpha\delta$ sii existe en P la regla $A \rightarrow \delta$ y $\alpha \in T^*$ $A \in N \cup \{S\}$

siendo $\delta = aB$ o $\delta = a$ $a \in T$ $B \in N$

Cuando $A = S$ puede ser $\delta = \varepsilon$

Ejemplo Si G = ({A}, {a, b}, S, P) donde P = { $S \rightarrow aA$, $A \rightarrow aA$, $A \rightarrow b$ }

$S \Rightarrow aA \Rightarrow aaA \Rightarrow aaaA \Rightarrow aaab$

$\alpha\ A \Rightarrow \alpha\ \delta$

En un paso se aplicó la regla $A \rightarrow aA$

Gramáticas Regulares (Tipo 3)

Derivación: La cadena **β** se obtiene de la cadena **ω** en cero o más pasos usando las reglas de P. Se define la clausura reflexiva y transitiva de $\Rightarrow$

$\alpha_1 \Rightarrow \alpha_2 \Rightarrow \ldots \Rightarrow \alpha_n$ decimos que $\alpha_1 \overset{*}{\Rightarrow} \alpha_n$ para $\alpha_i \in (N \cup T)^*$

Ejemplo

$S \Rightarrow aA \Rightarrow aaA \Rightarrow aaaA \Rightarrow aaab$ $\rightarrow S \overset{*}{\Rightarrow} aaab$

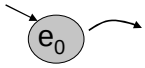
En varios pasos

Lenguaje generado por una gramática regular G = (N, T, P, S):

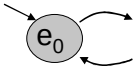
$L(G) = \{ x / x \in T^* \text{ y } S \overset{*}{\Rightarrow} x \}$

Es decir, una cadena $\in$ L(G) si:
1) La cadena está formada por símbolos terminales únicamente
2) La cadena puede ser derivada a partir de S

Pasaje de Autómata Finito a Gramática Regular

1) 

S



S, A

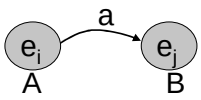
Si tiene arcos salientes

Si tiene arcos salientes, y entrantes

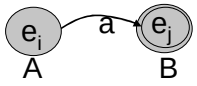
2) 

B

al resto de los estados asociar un no terminal

3) 

agregar la regla $A \rightarrow aB$

4) 

agregar las reglas $A \rightarrow aB$
$A \rightarrow a$

5) 

agregar la regla $S \rightarrow \varepsilon$

Expresiones Regulares

Se denominan Expresiones Regulares (ER) sobre un alfabeto A, a las expresiones que se pueden construir a partir de las siguientes reglas:

- $\varnothing$ es ER que describe el lenguaje vacío

- ε es ER que describe el lenguaje $\{\varepsilon\}$ (el lenguaje que contiene sólo la cadena vacía)

-Para cada símbolo $a \in A$, a es ER que describe el lenguaje {a}

- Si r y s son ER que describen los lenguajes L(r) y L(s) respectivamente:

  r **+** s es ER que describe el lenguaje $L(r) \cup L(s)$

  r **.** s es ER que describe el lenguaje L(r) **.** L(s)

  r^* es ER que describe el lenguaje $L(r)^*$

• Precedencia de operadores (de mayor a menor): *, **.** , +

• Se pueden usar paréntesis

Expresiones Regulares

Las ER describen a los lenguajes regulares (aquellos reconocidos por autómatas finitos y generados por gramáticas regulares).

Ejemplos:

Dado el alfabeto A = {a, b}

ER	Lenguaje que describe
r = a + b	L(r) = {a, b}
r = ab	L(r) = {ab}
r = a^*b	L(r) = {$a^n b$ / $n \geq 0$ }
r = $(a + b)^*b$	L(r) = { x / x $\in$ {a, b}* y x termina en b}
r = $(a + b)^*ab\,(a + b)^*$	L(r) = { x / x $\in$ {a, b}* y x contiene ab}

Expresiones Regulares

Ejemplos:

1) ¿Qué lenguaje describe ER r_1 = digito . dig^* ?

donde digito = {1, 2, 3, 4, 5, 6, 7, 8, 9} y dig = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}

2) ¿Qué lenguaje describe ER r_2 = letra. $(letra + dig)^*$?

donde letra = {a, b, …, z} y dig = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}

3) ¿Qué lenguaje describe ER $r_3 = ab^* + a$?

4) ¿Qué lenguaje describe ER $r_4 = ab^*$?

5) ¿Qué puede decirse de r_3 y r_4?

Expresiones Regulares

Expresiones regulares equivalentes:

Dos ER r_1 y r_2 son equivalentes $r_1 \equiv r_2$ si $L(r_1) = L(r_2)$

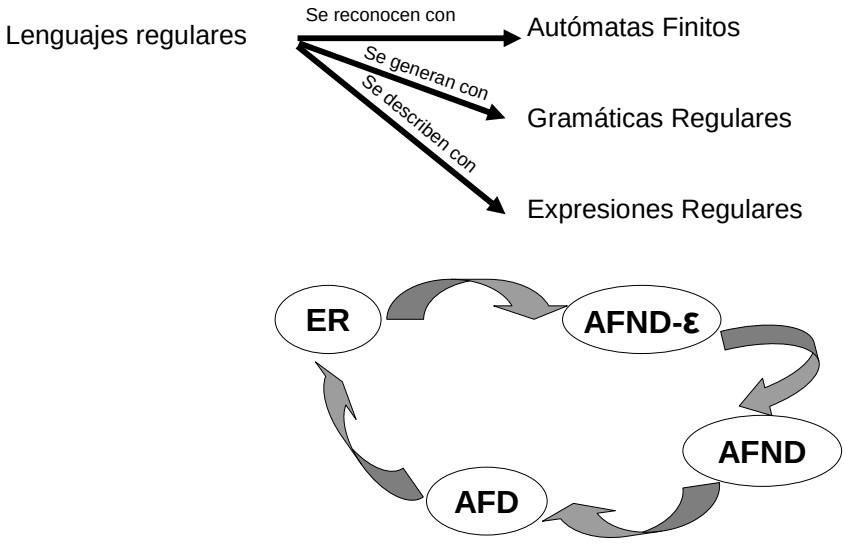
(es decir, r_1 y r_2 describen el mismo conjunto de cadenas)

Leyes algebraicas para expresiones regulares

Sean r, s y t expresiones regulares:

1) $r + \emptyset \equiv \emptyset + r \equiv r$
2) $r \,.\, \varepsilon \equiv \varepsilon \,.\, r \equiv r$
3) $r \,.\, \emptyset \equiv \emptyset \,.\, r \equiv \emptyset$
4) $r + s \equiv s + r$
5) $(r + s) + t \equiv r + (s + t)$
6) $(r \,.\, s) \,.\, t \equiv r \,.\, (s \,.\, t)$
7) $r \,.\, (s + t) \equiv r \,.\, s + r \,.\, t$
8) $(s + t) \,.\, r \equiv s \,.\, r + t \,.\, r$
9) $r + r \equiv r$
10) $\emptyset^* \equiv \varepsilon$
11) $r \,.\, r^* \equiv r^* \,.\, r$
12) $r \,.\, r^* + \varepsilon \equiv r^*$
13) $(r^* \,.\, s^*)^* \equiv (r + s)^*$

Lenguajes Regulares



ER – AFND-ε

Algoritmo para construir AFND- **ε** a partir de ER

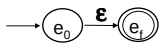
Sea r una ER con n operadores y sin variables como operandos atómicos, existe un AFND M con transiciones ε (AFND-ε) que acepta solamente aquellas cadenas que están en L(r).

M tiene un estado final, no entran arcos al estado inicial y no salen arcos del estado final.

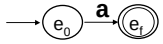
• Si r no tiene operadores, entonces:

1) Para $r = \varnothing$ el AFND-ε es $M = <\{e_0, e_f\}, A, \delta, e_0, \{e_f\}>$

2) Para $r = \varepsilon$ el AFND-ε es $M = <\{e_0, e_f\}, A, \delta, e_0, \{e_f\}>$



3) Para r = a ($a \in A$) el AFND-ε es $M = <\{e_0, e_f\}, A, \delta, e_0, \{e_f\}>$



ER – AFND-ε

Algoritmo para construir AFND- **ε** a partir de ER

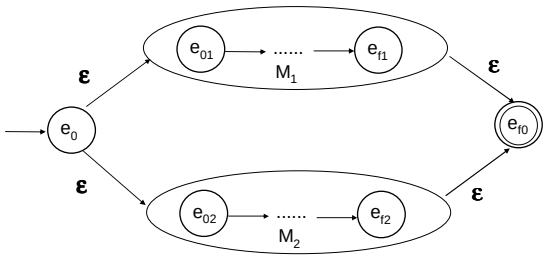
• Si r tiene operadores:

1) $r = r_1 + r_2$

Sean $M_1 = <E_1, A, \delta_1, e_{01}, \{e_{f1}\}>$ y $M_2 = <E_2, A, \delta_2, e_{02}, \{e_{f2}\}>$ $E_1 \cap E_2 = \varnothing$

tal que $L(M_1) = L(r_1)$ y $L(M_2) = L(r_2)$

Nuevo autómata $M = < E_1 \cup E_2 \cup \{e_0, e_{f0}\}, A, \delta, e_0, \{e_{f0}\}>$



$e_0, e_{f0} \notin E_1 \cup E_2$

$L(M) = L(r_1) \cup L(r_2)$

ER – AFND-ε

Algoritmo para construir AFND- **ε** a partir de ER

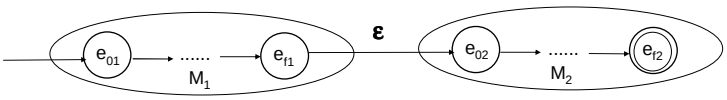
• Si r tiene operadores:

2) $r = r_1 \cdot r_2$

Sean $M_1 = <E_1, A, \delta_1, e_{01}, \{e_{f1}\}>$ y $M_2 = <E_2, A, \delta_2, e_{02}, \{e_{f2}\}>$ $E_1 \cap E_2 = \varnothing$

tal que $L(M_1) = L(r_1)$ y $L(M_2) = L(r_2)$

Nuevo autómata $M = < E_1 \cup E_2, A, \delta, e_{01}, \{e_{f2}\}>$



$L(M) = L(r_1) \cdot L(r_2)$

ER – AFND-ε

Algoritmo para construir AFND- **ε** a partir de ER

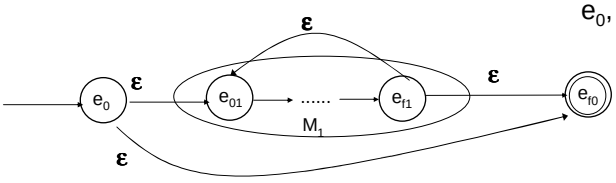
• Si r tiene operadores:

3) $r = r_1^*$

Sea $M_1 = <E_1, A, \delta_1, e_{01}, \{e_{f1}\}>$ tal que $L(M_1) = L(r_1)$

Nuevo autómata $M = < E_1 \cup \{e_0, e_{f0}\}, A, \delta, e_0, \{e_{f0}\}>$

$e_0, e_{f0} \notin E_1 \cup E_2$



$L(M) = L(r_1)^*$

AFND-ε - AFND

Dado un AFND-ε es posible construir un AFND equivalente sin transiciones vacías que reconoce el mismo lenguaje.

• Estados del AFND: *estados importantes* del AFND-ε y estado inicial del AFND-ε.
(estados importantes son los estados a los que llega un arco con un símbolo $\neq \varepsilon$ como rótulo)

• Alfabeto del AFND: alfabeto del AFND-ε.

• Estado inicial del AFND: estado inicial del AFND-ε.

• Función de transición del AFND: se define una transición del estado importante $\mathbf{e_i}$ al estado importante $\mathbf{e_j}$ con el símbolo x, si existe algún estado $\mathbf{e_k}$ tal que:
- se puede llegar desde el estado $\mathbf{e_i}$ al estado $\mathbf{e_k}$ con un camino de 0 ó mas transiciones **ε**; se permite $\mathbf{e_i} = \mathbf{e_k}$;
- en el AFND-ε existe una transición del estado $\mathbf{e_k}$ al estado $\mathbf{e_j}$ rotulada con el símbolo x.

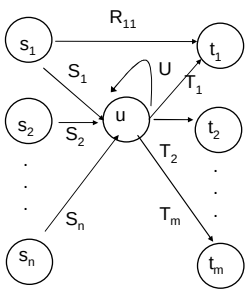
• Estados finales del AFND: estados finales del AFND-ε y todos los estados $\mathbf{e_i}$ del AFND-ε para los cuales existe un camino con transiciones **ε**, en el AFND-ε, a algún estado final del AFND-ε.

AF - ER

- A partir de cualquier AF es posible obtener una ER que describe el lenguaje reconocido por el autómata.

- La conversión consiste en ir eliminando los estados del autómata uno por uno, reemplazando los rótulos sobre los arcos, que inicialmente son símbolos, por expresiones regulares más complicadas.

- Eliminación de un estado



u estado a eliminar
s_i estados predecesores
t_j estados sucesores

Después de eliminar u y todos los arcos que llegan y salen de u, se debe reemplazar el rótulo R_{ij} del arco de s_i a t_j por la expresión regular

$$R_{ij} + S_i U^* T_j$$

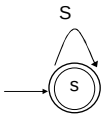
Cuando algún arco no está presente, se puede agregar uno rotulado ∅

AF - ER

Algoritmo para construir la expresión regular a partir del autómata:

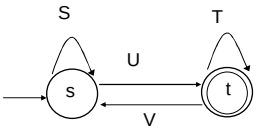
1) Repetir para cada estado final:

• Si el estado final es también inicial, eliminar todos los estados excepto el estado inicial.



$ER = S^*$

•Sino, eliminar los estados del autómata hasta que queden únicamente el estado inicial y el estado final en consideración.



$ER = S^* U (T + V S^* U)^*$

ó

$ER = S^* U (T^* (V S^* U)^*)^*$

2) Realizar la unión de las ER obtenidas para cada estado final del autómata.