GRAMÁTICAS, LENGUAJES Y AUTÓMATAS
2020

Programación Concurrente

FCEFyN

UNC

Bibliografia

- Teoría de autómatas y lenguajes formales, jurado Málaga, Elena, 2008
- Lenguajes, Gramáticas y Autómatas en Procesadores de Lenguaje, Cueva, JM - Izquierdo, R - Juan, AA - Luengo, MC- Ortín, F - Labra, JE, 2003

CONTENIDOS

- Motivación
- Capitulo 1
  - Introducción
- Capitulo 2
  - Definiciones previas
- Capitulo 3
  - Definición formal de gramática
  - Relaciones entre cadenas
  - Sentencias o instrucciones
- Capitulo 4
  - Definición formal de Autómata
  - Algoritmo para obtener la gramática regular de un autómata finito
  - Notación
- Capitulo 5
  - Definición formal de lenguaje
    - Jerarquía de las gramáticas
- Capitulo 6
  - Correspondencia entre gramáticas y lenguajes
- Capitulo 7
  - Expresiones regulares
- Capitulo 8
  - Autómatas, definición formal
  - Máquinas de Moore , Mealy
- Capitulo 9
  - Jerarquía de los autómatas
- Capitulo 10
  - Maquina de Turing
- Capitulo 11
  - Autómatas lineales acotados
- Capitulo 12
  - Autómatas de pila
- Capitulo 13
  - Autómatas finitos

Motivación, *Sistemas Reactivos*

- Los **sistemas reactivos** (RS) son sistemas informáticos que **responden continuamente a su entorno** a una velocidad determinada por éste. La mayoría de los sistemas embebidos de tiempo real, control, supervisión, procesamiento de señales, protocolos de comunicación e interfaces hombre-máquina son sistemas reactivos.
- Esta clase de sistemas **se distingue de los sistemas transformacionales** y también de los i**nteractivos**. Los sistemas transformacionales realizan cálculos, cuyas salidas están disponibles al finalizar la ejecución; cuando terminan, el estado del sistema no es significativo. Mientras que, los sistemas interactivos interactúan continuamente con su entorno, pero a su propio ritmo.
- Los sistemas reactivos **realizan procesos no terminados basados en estados** que están interactuando con el entorno para habilitar, aplicar o prevenir cierto comportamiento en el mismo. Pueden estar sujetos a requisitos rigurosos de tiempo real y, a menudo, realizar varios procesos en paralelo.

Motivación, *Arquitectura dirigida por eventos*

- La arquitectura dirigida por eventos (EDA) **es un patrón de arquitectura de software impulsado por la producción, detección, consumo, y reacción a eventos.**
- Desde una perspectiva formal, lo que es producido, publicado, propagado, detectado o consumido **es un mensaje llamado notificación del evento**, y no el hecho que generó el evento. El término evento es usado para denotar el mensaje de notificación del estímulo.
- Un sistema dirigido por eventos **poseer emisores y consumidores de eventos**. Los consumidores tienen la responsabilidad de llevar a cabo una reacción tan pronto como el evento esté presente. Esta puede filtrar, transformar y reenviar el evento a otro componente o proporcionar una reacción.
- Una EDA facilita un **mayor grado de reacción**, debido a que están diseñados para entornos no predecibles y asíncronos.

Capitulo 1 INTRODUCCIÓN

- El objetivo es introducir los conceptos teóricos necesarios sobre Teoría de Lenguajes Formales, Gramáticas y Autómatas
  - Se presenta la Teoría de Gramáticas y Lenguajes Formales, como una **herramienta matemática**
  - Se desarrollan los conceptos necesarios para la **construcción de Autómatas** para el reconocimiento de lenguajes de programación
- Historia
  - La Teoría de los Lenguajes Formales tiene su origen en un campo aparentemente bastante alejado de la Informática: la Lingüistica.
  - Los lingüistas de la llamada *escuela estructuralista americana* habían elaborado por los años 50 algunas ideas informales acerca de la gramática universal.
  - Se entiende por **gramática universal**, una gramática que caracteriza las propiedades generales de cualquier lenguaje humano
  - El primer trabajo que desarrolló teorías formales sobre gramáticas y lenguajes fue obra de Avram Noam Chomsky (1928-),
  - El concepto de Gramática Formal adquirió gran importancia para la especificación de lenguajes de programación; concretamente, se definió con sus teorías la sintaxis del lenguaje ALGOL 60

Introducción

- La Teoría de Lenguajes Formales es de gran utilidad para el trabajo en campos de la Informática por ejemplo en Informática Teórica, Inteligencia Artificial, Procesamiento de lenguajes naturales (comprensión, generación, y traducción), Reconocimiento del Habla, los Sistemas Reactivos.
- La Teoría de los **Lenguajes y Gramáticas Formales tiene una relación directa con la Teoría de Autómatas**, siendo posible establecer entre ambas una correspondencia denominada en Algebra isomorfismo
- La Teoría de los Autómatas proviene del campo de la **Ingeniería Eléctrica.**
- El científico estadounidense **Claude Elwood Shannon**(1916-2001) publicó varios trabajos, donde mostraba las bases para la aplicación de la Lógica Matemática a los circuitos combinatorios y secuenciales. Dando lugar a la Teoría de Autómatas

Introducción

- Los autómatas son sistemas que **reciben información, la transforman y producen otra información** que se transmite al entorno.
- La Teoría de Autómatas tiene aplicación en campos muy diversos :
  - Lógica de los Circuitos Secuenciales
  - Teoría de Control de Sistemas
  - Teoría de la Comunicación
  - Arquitectura de Ordenadores
  - Redes Conmutadoras y Codificadoras
  - Teoría de los Sistemas Evolutivos y Auto-reproductivos
  - Reconocimiento de patrones
  - Redes Neuronales
  - Reconocimiento y procesado de lenguajes de programación
  - Traducción de lenguajes
  - Teoría de Lenguajes Formales

Capitulo 2 DEFINICIONES PREVIAS

- **Símbolo**
  - Es una entidad abstracta, que no se va a definir, pues se dejará como axioma.
  - Normalmente los símbolos son letras (a, b, c, . . . ,z), dígitos (0, 1, . . ., 9), y otros caracteres (+, -, *, /, ?, . . .).
    - Los símbolos también pueden estar formados por varias letras o caracteres
- **Vocabulario o alfabeto**
  - Es un conjunto finito de símbolos, no vacío.
  - Para definir que un símbolo **a** pertenece a un alfabeto **V** se utiliza la notación a $\boldsymbol{a} \in \boldsymbol{V}$.
  - Los alfabetos se definen por enumeración de los símbolos que contienen.
  - También se puede definir las tablas ASCII y EBCDIC como alfabetos
- **Cadena**
  - Una cadena es una secuencia finita de símbolos de un determinado alfabeto.
  - Longitud de cadena
    - La longitud de una cadena es el número de símbolos que contiene. La notación empleada es la que se indica en los siguientes ejemplos.

$$| abcb | \rightarrow 4$$
$$| a + 2 * b | \rightarrow 5$$

- **Cadena vacía**
  - Existe una cadena denominada cadena vacía, que no tiene símbolos y se denota con λ , entonces su longitud es :

$$|\lambda| \rightarrow 0$$

DEFINICIONES PREVIAS

- **Concatenación de cadenas**
  - Sean $\alpha\ y\ \beta$ dos cadenas cualesquiera, se denomina concatenación de $\alpha\ y\ \beta$ a una nueva cadena $\alpha\beta$ constituida por los símbolos de la cadena α seguidos por los de la cadena β .
  - El elemento neutro de la concatenación es : λ , se cumple:

$$\alpha\lambda = \lambda\alpha = \alpha$$

- **Universo del discurso**
  - El conjunto de todas las cadenas que se pueden formar con los símbolos de un alfabeto $V\ se\ denomina\ universo\ del\ discurso\ de\ V\ y\ se\ representa\ por\ W(V)$
  - Evidentemente W(V) es un conjunto infinito.
  - La cadena vacía pertenece a W(V) (Cierre o Clausura de un Alfabeto).
    - Sea un alfabeto con una sola letra $V = \{ a \}$, entonces el universo del discurso es : $W(V) = \{\lambda, a, aa, aaa, aaaa, \dots\}$ que contiene infinitas cadenas.
    - **Clausura Positiva de un Alfabeto**: la Clausura Positiva de una Alfabeto está formado por el Universo de Discurso del Alfabeto sin incluir la palabra vacía.
    - **Cierre o Clausura de un Alfabeto**: el Cierre o Clausura de un Alfabeto es el Universo de Discurso de dicho Alfabeto, incluyendo la palabra vacía.

DEFINICIONES PREVIAS

- **Lenguaje**
  - Se denomina lenguaje sobre un alfabeto V a un subconjunto del universo del discurso.
    - También se puede definir como un conjunto de palabras de un determinado alfabeto.
    - Alguien puede pensar que los lenguajes se pueden definir por enumeración de las cadenas que pertenecen a dicho lenguaje, pero este método además de ineficiente, es en muchos casos imposible (esto sucede cuando el lenguaje tiene infinitas cadenas).

Los lenguajes se definen por las propiedades que cumplen las cadenas del lenguaje.

Ejemplo: el conjunto de palíndromos (cadenas que se leen igual hacia adelante, que hacia atrás) sobre el alfabeto $\{0,1\}$, este lenguaje tiene infinitas cadenas. Algunas cadenas de este lenguaje son:

$$\lambda\ ,0\ ,1\ ,00\ ,11\ ,010\ ,\ 0110,\ 000000,\ 101101,\ 111111,\ \ldots..$$

- **Lenguaje vacío**
  - el *lenguaje vacío* es un conjunto vacío y que se denota por $\{\emptyset\ \}$.
    - El lenguaje vacío no debe confundirse con un lenguaje que contenga una sola cadena, y que sea la cadena vacía, es decir $\{\ \lambda\}$, ya que el número de elementos (cardinalidad) de estos dos conjuntos es diferente: $Cardinal\ (\{\emptyset\})\ =\ 0, Cardinal\ (\{\lambda\})\ =\ 1$

DEFINICIONES PREVIAS

- **Gramática**

  La gramática es un ente formal para especificar, de una manera finita, el conjunto de cadenas de símbolos que constituyen un lenguaje.

- **Autómata**

  Un autómata es una construcción lógica que recibe una entrada y produce una salida en función de todo lo recibido hasta ese instante.

  - En el caso de los *Procesadores de Lenguaje* un autómata es una construcción lógica que recibe como entrada una cadena de símbolos y produce una salida indicando si dicha cadena pertenece o no a un determinado lenguaje.

CAPÍTULO 3: DEFINICIÓN FORMAL DE GRAMÁTICA

- Una gramática es una **cuádrupla :** $\boldsymbol{G = (VT, VN, P, S)}$
  - donde :
    - VT = {conjunto finito de símbolos terminales} (estados)
    - VN = {conjunto finito de símbolos no terminales} (eventos)
    - S es el *símbolo inicial* y pertenece a VN.
    - P = {conjunto de producciones o de reglas de derivación}
    - Todas las cadenas del lenguaje definido por la gramática están formados con símbolos del ***vocabulario terminal*** **VT**.
      - El vocabulario terminal se define por enumeración de los símbolos terminales.
    - El **vocabulario no terminal VN** es el conjunto de símbolos introducidos como elementos auxiliares para la definición de la gramática, y que no figuran en las cadenas del lenguaje.
      - El vocabulario no terminal se define por enumeración de los símbolos no terminales.
      - La intersección entre el VT y VN es el conjunto vacío : $\{VN\} \cap \{VT\} = \{\emptyset\}$
      - La unión entre el VT y VN es el vocabulario : $\{VN\} \cup \{VT\} = \{V\}$
        - En ocasiones es importante distinguir si un determinado vocabulario incluye o no la cadena vacía
          $V^+ = V - \lambda \; y \; V^* = V + \lambda$
    - El ***símbolo inicial*** **S** es un símbolo no terminal a partir del cual se aplican las reglas de la gramática para obtener las distintas cadenas del lenguaje
    - Las **producciones P** son las reglas que se aplican desde el símbolo inicial para obtener las cadenas del lenguaje.
      - El conjunto de producciones P se define por medio de la enumeración de las distintas producciones, en forma de reglas o por medio de un metalenguaje

Capitulo 4: Definición formal de autómata

- **Un autómata es una quíntupla** $A = (E, S, Q, f, g)$ donde :
  - E = {conjunto de entradas o vocabulário de entrada}
    - E es un conjunto finito, y sus elementos se llaman entradas o símbolos de entrada.
  - S = {conjunto de salidas o vocabulário de salida}
    - S es un conjunto finito, y sus elementos se llaman salidas o símbolos de salida.
  - Q = {conjunto de estados}
    - Q es el conjunto de estados posibles, puede ser finito o infinito.
  - f :$ExQ \rightarrow Q$
    - es la *función de transición* o función del estado siguiente, y para un par del conjunto
    - ExQ devuelve un estado perteneciente al conjunto Q.

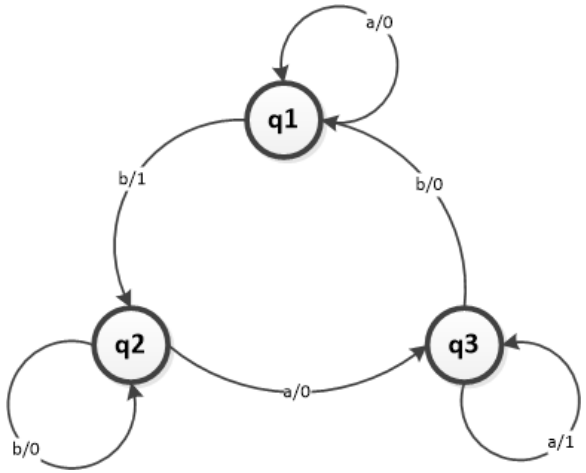
      ExQ es el conjunto producto cartesiano de E por Q.
  - g : $ExQ \rightarrow S$
    - es la *función de salida*, y para un par del conjunto E Q, devuelve un símbolo de salida del conjunto S.

- **Representación de autómatas**
  - Los autómatas se pueden representar mediante :
    - Tabla de transiciones
    - Diagrama de Moore

Tabla de transiciones

- Las funciones f y g pueden representarse mediante una tabla, con tantas filas como estados y tantas columnas como entradas.
- Así por ejemplo se puede representar el autómata
- A = (E, S, Q, f, g) donde
  - E = {*a,b*}, (entradas)
  - S = {*0,1*}, (salidas)
  - Q = {q1, q2, q3}, (estados)
  - f y g se pueden representar

    por las siguientes relaciones:



f	*a*	*b*
q_1	q_1	q_2
q_2	q_3	q_2
q_3	q_3	q_1

g	*a*	*b*
q_1	*0*	*1*
q_2	*0*	*0*
q_3	*1*	*0*

f / g	*a*	*b*
q_1	$q_1/0$	$q_2/1$
q_2	$q_3/0$	$q_2/0$
q_3	$q_3/1$	$q_1/0$

Así se tiene que $f(a, q_1) = q_1$; $g(a, q_1) = 0$; o también $f(a, q_2) = q_3$; y $g(a, q_3) = 1$

Notación

- **Vocabulario terminal**
  - Los elementos del vocabulario terminal se representan por :
    - - letras minúsculas de comienzo del abecedario : a, b, c, . . . , g.
    - - operadores tales como : + , - , * , / , . . .
    - - caracteres especiales : # , @ , (,) , . , ; , . . .
    - - los dígitos : 0, 1, . . . , 9
    - las palabras reservadas de lenguajes de programación con letras minúsculas y en negrita : if, then, else, . . .
- **Vocabulario no terminal**
  - Los elementos del vocabulario no terminal se representan por :
    - - letras mayúsculas de comienzo del abecedario : A, B, . . . , G.
    - La única excepción suele ser el símbolo inicial que se representa con **S**.
    - - nombres en minúscula, pero encerrados entre paréntesis angulares : <expresión>, <operador>, .
- **Vocabulario**
  - Los elementos indiferenciados del vocabulario terminal y no terminal se denotan con : las letras mayúsculas del final del abecedario : U, V, W, X, Y, Z.
- **Cadenas terminales**
  - Las cadenas compuestas totalmente por símbolos terminales se representan como : las letras minúsculas del final del abecedario : t, u, v, x, y, z.
- **Cadenas**
- Las cadenas que contienen símbolos terminales y no terminales indiferenciados se representan por : letras minúsculas griegas : $\alpha,\ \beta,\ \gamma,\ \delta,\ \varepsilon, \ldots$

Algoritmo para obtener la gramática regular desde el autómata finito

- Algoritmo para obtener una gramática regular que genera un lenguaje regular dado a partir del autómata finito que reconoce ese lenguaje
- Una gramática formal es una cuadrupla

$$G = (N, T, P, S)$$

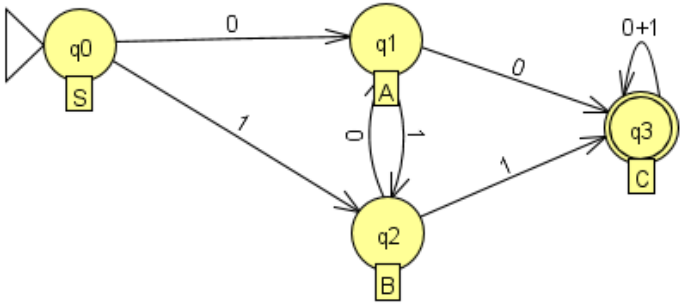
- Donde
  - N es un conjunto finito de símbolos no terminales
  - T es un conjunto finito de símbolos terminales
  - P es un conjunto finito de producciones
  - S es el símbolo distinguido o axioma

Algoritmo para obtener la gramática regular desde el autómata finito

- Ejemplo
  - $L4 = \{ x/x \in \{0,1\}^* \; y \; x \; contiene \; la \; subcadena \; 00 \; ó \; x \; contiene \; la \; subcadena \; 11\}$
  - L4 = < {q0, q1, q2, q3}, {0, 1}, δ, p0, {q3}> ; δ es definido por el diagrama de transiciones
  - Como al estado inicial no entran arcos, se asocia únicamente el símbolo distinguido S.
- La gramática correspondiente a este lenguaje es
- G = ({A, B, C}, {0, 1}, P, S), siendo P el siguiente conjunto:

$S \rightarrow 0A; \;\; \delta(q0, 0) = q1 \; y$
$S \; y \; A \; estan \; asociados \; a \; p0 \; y \; p1 \; respectivamente$

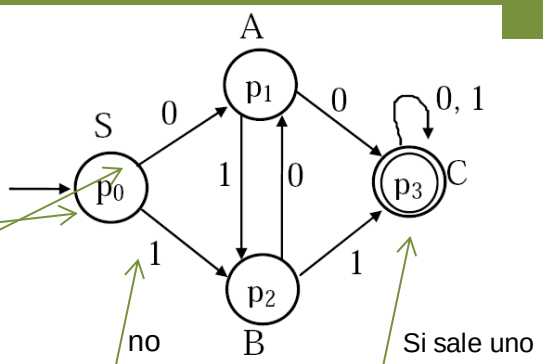
$S \rightarrow 1B; \;\; \delta(q0, 1) = q2 \; y$
$S \; y \; B \; estan \; asociados \; a \; p0 \; y \; p2 \; respectivamente$



LHS		RHS
C	→	λ
B	→	1C
C	→	0+1C
S	→	1B
B	→	0A
A	→	1B
S	→	0A
A	→	0C

Jflat

Algoritmo para obtener la gramática regular desde el autómata finito



- 1) Asociar al **estado inicial** el símbolo distinguido S.
- 2) Asociar **a cada estado** del autómata (menos el estado inicial) un símbolo **no terminal** {A,B,C}.

  Si **al** estado **inicial** llega algún arco asociar también un símbolo terminal (además del símbolo distinguido).

  No asociar símbolo no terminal a aquellos estados finales de los que **no salen arcos**.
- 3) Para **cada transición** definida δ (ei , a) = ej , **agregar un elemento al conjunto de producciones**, la producción A → 1B, siendo A y B los símbolos no terminales asociados a ei y ej respectivamente.

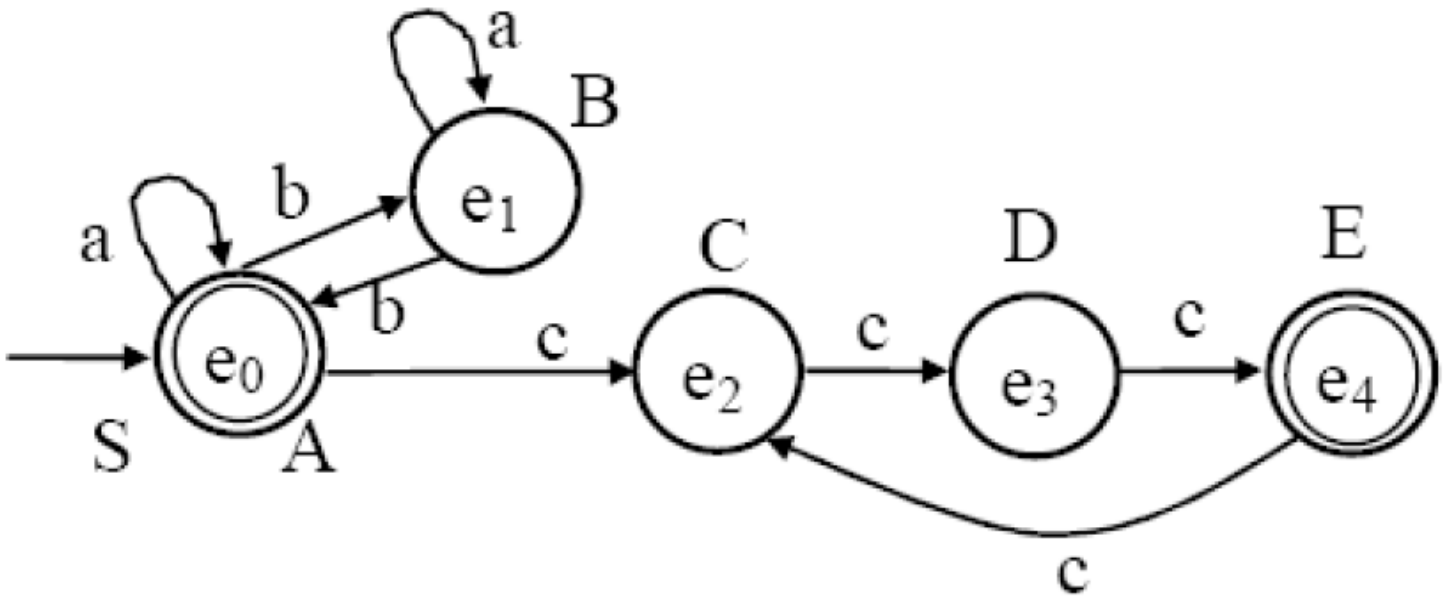
  Si ej es un estado final, agregar también la producción A → a.

  Si ej es **el estado inicial** (tiene dos símbolos asociados, el distinguido y un no terminal), **utilizar el símbolo no terminal** (de esta manera se evita que el símbolo distinguido aparezca a la derecha de una producción).
- 4) Si el estado inicial es también final agregar la producción S → ε.

$$G = (N, T, P, S)$$

Ejercicio

- $L3 = \{xc^{3m} | \, x \, \{a, b\}^*$ y la cantidad de b's es par y $m \geq 0\}$

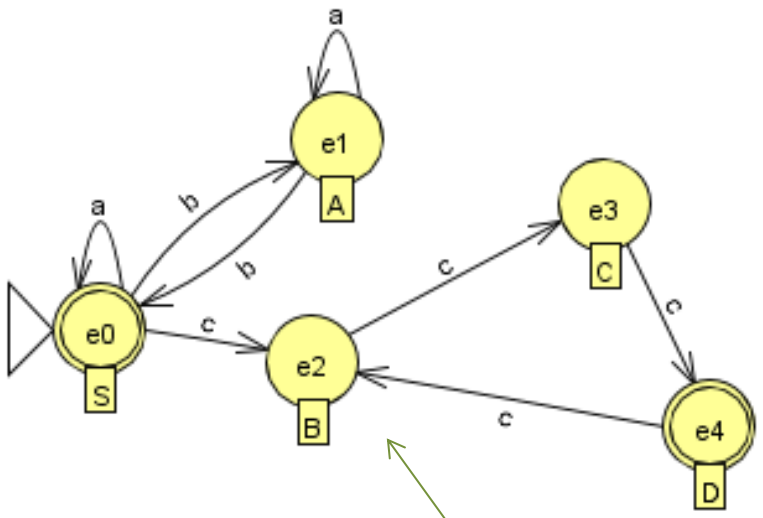


Solución

$L_3 = \{xc^{3m} / x \in \{a, b\}^*$ y la cantidad de b's es par y $m \geq 0\}$, siendo $L_3 = L(M_{3D})$

$M_{3D} = < \{e_0, e_1, e_2, e_3, e_4\}, \{a, b, c\}, \delta_{3D}, e_0, \{e_0, e_4\}>$

δ_{3D} está definida por el siguiente diagrama de transición de estados



LHS		RHS
A	→	aA
C	→	cD
S	→	aS
S	→	cB
S	→	λ
D	→	λ
D	→	cB
B	→	cC
A	→	bS
S	→	bA

Jflat

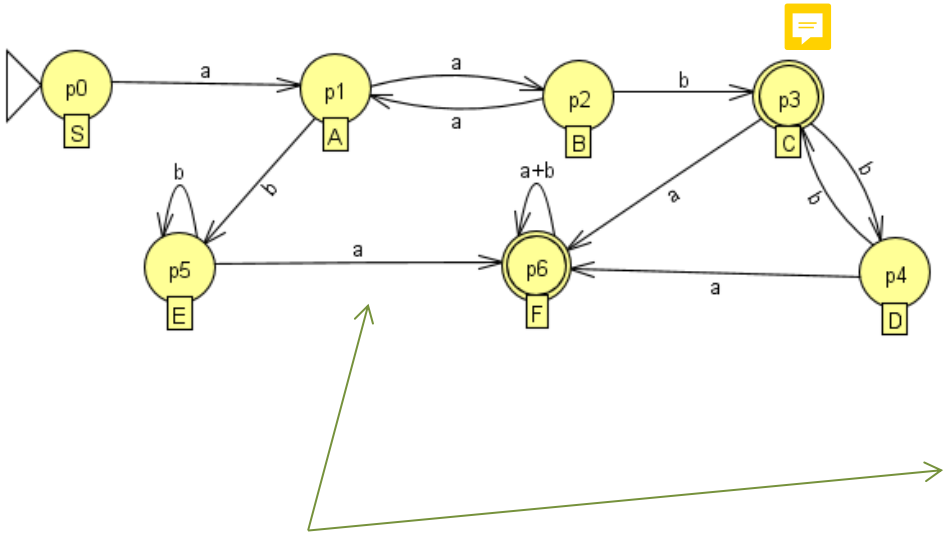
Algoritmo para obtener la gramática regular desde el autómata finito

- Derivación de la gramática correspondiente al lenguaje

$L_7 = \{ a^{2n}b^{2k+1} / n \geq 1 \text{ y } k \geq 0\} \cup \{ax / x \in \{a, b\}^* \text{ y } x \text{ contiene la subcadena ba}\}$

$L_7 = L(M_{7Dmin})$, $M_{7Dmin} = < \{p_0, p_1, p_2, p_3, p_4, p_5, p_6\}, \{a, b\}, \delta, p_0, \{p_3, p_6\}>$

δ está definida por el siguiente diagrama de transición de estados



Jflat

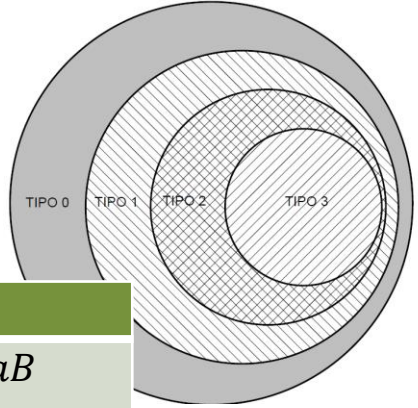
LHS		RHS
C	→	aF
E	→	bE
A	→	bE
C	→	λ
F	→	λ
E	→	aF
D	→	aF
D	→	bC
C	→	bD
S	→	aA
A	→	aB
B	→	aA
F	→	a+bF
B	→	bC

CAPÍTULO 5: Definición formal de los Lenguajes

JERARQUÍA DE LAS GRAMÁTICAS

- *Chomsky* definió cuatro tipos distintos de gramáticas en función de la forma de las reglas de derivación P (*Chomsky, 1959*).
- La clasificación comienza con un tipo de gramáticas que pretende ser universal, aplicando restricciones a sus reglas de derivación se van obteniendo los otros tres tipos de gramáticas.
- Esta clasificación es jerárquica, es decir cada tipo de gramáticas engloba a todos los tipos siguientes.
  - **Gramáticas de tipo 0**
    - También llamadas *gramáticas no restringidas* o *gramáticas con estructura de frase*.
  - **Gramáticas de tipo 1**
    - También llamadas *gramáticas sensibles al contexto*
  - **Gramáticas de tipo 2**
    - También se denominan *gramáticas de contexto libre* o *libres de contexto*
  - **Gramáticas de tipo 3**
    - También denominadas *regulares* o *gramáticas lineales a la derecha* comienzan sus reglas de producción por un símbolo terminal, que puede ser seguido o no por un símbolo no terminal

Jerarquía entre las gramáticas

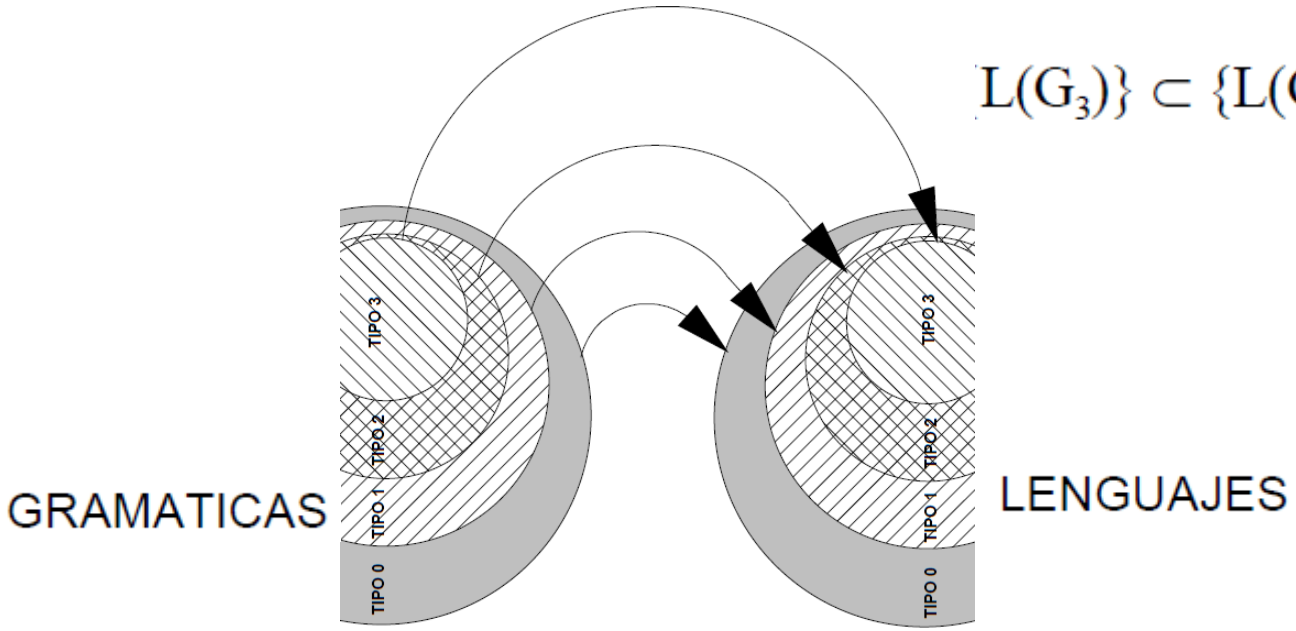


Gramáticas tipo	0	1	2	3
reglas derivación	$\alpha \to \beta$	$\alpha A \beta \to \alpha \gamma \beta$	$A \to \alpha$	$A \to aB$ $A \to a$
siendo	$\alpha \in (T \cup N)^+$, $\beta \in (T \cup N)^*$	$A \in N$ $\gamma \in (T \cup N)^+$, $\alpha, \beta \in (T \cup N)^*$	$A \in N$ $\alpha \in (T \cup N)^+$,	$A, B \in N$ $a \in T$
restricción	$\lambda \to \beta$	(1)	Cada regla es una par ordenado (A, α)	(4)
Ejemplo	$\{0^i 1^{i+k} 2^k 3^{n+1} \mid i, k, n \geq 0\}$ (2)	$\{a^n b^n c^n \mid n > 0\}$ (3)	$\{a^n b^n : n \geq 1\}$	$aa^* bb^*$
(1)	Se puede remplazar A por γ siempre que estén en el contexto $\alpha \ldots \beta$ la longitud de la parte derecha de la producción es mayor o igual a la de la parte izquierda.			
(2)	http://www.exa.unicen.edu.ar/catedras/ccomp1/Apunte5.pdf			
(3)	http://www.exa.unicen.edu.ar/catedras/ccomp1/ApunteGramSensibles.pdf			
(4)	La regla de producción comienzan por un símbolo terminal, que puede ser seguido o no por un símbolo no terminal			

CAPÍTULO 6: CORRESPONDENCIA ENTRE GRAMÁTICAS Y LENGUAJES

- Se denomina *lenguaje de tipo 0* al generado por una gramática de tipo 0.
- De la misma forma, se denominan *lenguajes de tipo 1, tipo 2, y tipo 3*, a los generados por las gramáticas de tipo 1, tipo 2, y tipo 3, respectivamente.
- Si los lenguajes generados por los distintos tipos de gramáticas se relacionan entre sí con respecto a la relación de inclusión se obtiene :

$$L(G_3)\} \subset \{L(G_2)\} \subset \{L(G_1)\} \subset \{L(G_0)\}$$



CAPÍTULO 7: EXPRESIONES REGULARES

- Anteriormente se estudiaron los lenguajes formales, y se vio como en algunos ejemplos no era fácil representar los lenguajes de una manera condensada.
- En este capítulo se va a presentar una herramienta, las expresiones regulares, para describir lenguajes de tipo 3 o lenguajes regulares.
- Las expresiones regulares se introducen para describir los lenguajes regulares, entonces las expresiones regulares serán metalenguajes.
- Es decir *las expresiones regulares son un metalenguaje para describir los lenguajes regulares.*

Operaciones con los lenguajes regulares

- a) **Unión o alternativa** : Sean dos lenguajes definidos sobre un mismo alfabeto, se denomina unión de los dos lenguajes al conjunto formado por las cadenas que pertenezcan indistintamente a uno u otro de los dos lenguajes. Formalmente se puede expresar :

$$L_1 \cup L_2 = \{x / x \in L_1 \vee x \in L_2\}$$

- b) **Concatenación** : Sean dos lenguajes definidos sobre el mismo alfabeto, se denomina concatenación de los dos lenguajes al conjunto de todas las cadenas formadas concatenando una palabra del primer lenguaje con otra del segundo. Formalmente se puede expresar :

$$L_1 L_2 = \{x_1 x_2 / x_1 \in L_1 \wedge x_2 \in L_2\}$$

- c) **Potencia de un lenguaje** : Esta no es una nueva operación, sino un caso particular de la anterior. Se denomina potencia i-ésima de un lenguaje a la operación que consiste en concatenarlo consigo mismo i-veces. En el caso de i=0, el resultado es el conjunto vacio.

- d) **Cierreu operación estrella** : La operación cierre de un lenguajeLes otro lenguaje L* obtenido uniendo el lenguaje L con todas sus potencias posibles, incluso L0. Formalmente se puede expresar como :

$$L^* = \{\varnothing\} \cup \{L\} \cup \{LL\} \cup \{LLL\} \cdots = \bigcup_{n=0}^{\infty} L^n$$

- e) **Cierre positivo** : La operación cierre positivo de un lenguaje L es otro lenguaje L+ obtenido uniendo el lenguaje L con todas sus potencias posibles, excepto L0. Formalmente se puede expresar como :

$$L^+ = \{L\} \cup \{LL\} \cup \{LLL\} \cdots = \bigcup_{n=1}^{\infty} L^n$$

Operaciones con las expresiones regulares

- Si α es una expresión regular, entonces $\{\alpha\}$ es el conjunto descrito por la expresión regular α
- También se puede decir que α denota el lenguaje de la cadena α.
- Las expresiones regulares describen los lenguajes regulares, luego sus operaciones corresponderán a las indicadas para los lenguajes regulares.

Operaciones con las expresiones regulares

Nota: $\cup \equiv |$

- a) **Unión o alternativa** : Si α y β son expresiones regulares, $\alpha \mid \beta$ es una expresión regular tal que : $\{\alpha \mid \beta\} = \{\alpha\} \cup \{\beta\}$ es decir puede aparecer o indistintamente.

- b) **Concatenación** : Si α y β son expresiones regulares, $\alpha\,\beta$ es una expresión regular tal que $\{\alpha\,\beta\} = \{\alpha\}\,\{\beta\}$

- c) **Cierre u operación estrella** : Si α es una expresión regular, α^* entonces es una expresión regular que denota $\{\alpha\}^*$.

- d) **Cierre positivo** : Si α es una expresión regular, entonces es una α^+ expresión regular que denota $\{\alpha\}^+$

- **Precedencia de las operaciones**
- Se permite el uso de paréntesis para indicar la precedencia de las operaciones, pero cuando no se utilizan paréntesis para evaluar una expresión regular, hay que tener en cuenta el siguiente orden de precedencia :
  - **1ª.- Uso de paréntesis**
  - **2ª.- Operación cierre y cierre positivo**
  - **3ª.- Operación concatenación**
  - **4ª.- Alternativa**

Expreciones regulares

- **Teorema**
  - Dos expresiones regulares son iguales, si designan al mismo conjunto regular.
- **Propiedades**

A partir del teorema anterior se pueden enunciar las siguientes propiedades :

a) **Asociatividad de la operación concatenación**

$$\alpha(\beta\gamma) = (\alpha\beta)\gamma$$

b) **Distributividad de la operación alternativa respecto de la concatenación**

$$\alpha\beta \mid \alpha\gamma \;=\; \alpha(\beta \mid \gamma)$$

c) λ es el **elemento neutro de la concatenación**, es decir

$$\alpha\lambda \;=\; \lambda\alpha \;=\; \alpha$$

d) **Propiedades de la operación cierre**

d.1) $(\alpha \mid \beta)^* \;=\; (\alpha^* \mid \beta^*)^* \;=\; (\alpha^*\beta^*)^*$

d.2) $(\alpha \mid \lambda)^* \;=\; (\alpha^* \mid \lambda) \;=\; \alpha^*$

d.3) $\alpha\alpha^* \mid \lambda \;=\; \alpha^*$

d.4) $\lambda^* \;=\; \lambda$