Teoría de la Computación y Lenguajes Formales

Propiedades de los Lenguajes Libres de Contexto (LLC)

Prof. Hilda Y. Contreras
Departamento de Computación
hyelitza@ula.ve
hildac.teoriadelacomputacion@gmail.com

UNIVERSIDAD DE LOS ANDES

Lenguaje Regular y Lenguaje Libre de Contexto

- Jerarquía de Chomsky (Tipo 3 y Tipo 2)

Tipo	Lenguaje	Máquina	Gramática
0	Recursivamente enumerable	Máquina de Turing	Sin restricciones
1	Dependiente del Contexto	Autómata linealmente acotado	*Gramática dependiente del contexto* $\alpha A \beta \rightarrow \alpha \gamma \beta$
2	**Independiente del Contexto**	**Autómata de Pila**	***Gramática libre de contexto*** $A \rightarrow \gamma$
3	**Lenguaje Regular**	**Autómata finito**	***Gramática Regular*** $A \rightarrow aB$ $A \rightarrow a$

Contenido

- Lema del Bombeo para los LLC
- Propiedades cerradas de los LLC
- Algoritmos de decisión de los LLC

Lemas del Bombeo

Antecedentes para LLC:

– Forma Normal de Chomsky (1959)

– Debido a Bar-Hillel, Perles & Shamir (1961)

– El lema de bombeo para los LRs es una simplificación del lema de bombeo para LLC

Lemas del Bombeo

Demostrar que un Lenguaje L **NO es LLC**

Importancia: identificar el tipo de lenguaje para poder usar las herramientas adecuadas para procesarlo.

Para demostrar que un Lenguaje L si es LLC ➔ Obtener un AP o una GLC

Lema del Bombeo

Sea L un lenguaje libre de contexto sobre Σ. Existe un número natural **m** (dependiente del lenguaje L) tal que para todo w en **L** si |w| ≥ **m**, existen **u**,**v**,**x**,**y**,**z** en Σ* tales que w = **uvxyz** y donde:

1. |**vxy**| ≤ **m**
2. |**vy**| ≥ 1
3. Para todo **i** ≥ 0, $\mathbf{uv^ixy^iz}$ en L

Condición **necesaria** para que un lenguaje sea libre de contexto: todos LLC tiene esta propiedad.

Lema del Bombeo

Si una cadena de L es suficientemente larga, siempre hay 2 subcadenas cortas (v, y) lo suficientemente cerca que pueden ser bombeadas el mismo número de veces.

$S \rightarrow uAz$

$A \rightarrow vAy \mid x$

S => uAz => uvAyz => uvxyz en L

S => uAz => uvAyz => uvvAyyz => uvvvAyyyz => … => uv^iAy^iz => uv^ixy^iz en L para $i \geq 0$

Lema del Bombeo

Método de uso:

1. Tomar **m** como el valor de la constante del lema de bombeo
2. Escoger una palabra w en L, tal que |w| ≥ **m**
3. Considerar todas las posibles factorizaciones de w (uvxyz según #2 y #1 del lema)
4. Mostrar que, para todas las factorizaciones posibles, puede encontrarse un valor de **i** tal que $\mathbf{uv^ixy^iz}$ no esta L (contradicción #3)

Lema del Bombeo

Por ejemplo: $L = \{ a^i b^i c^i \mid i \geq 0 \}$

1. Escoger una palabra **w** tal que **|w| ≥ m**
   $w = a^m b^m c^m$
2. Considerar todas las posibles factorizaciones de **w**
3. Mostrar que, para todas las factorizaciones posibles, puede encontrarse un valor de **i** tal que $\mathbf{uv^i xy^i z}$ no esta L

Lema del Bombeo

Por ejemplo: $L = \{ a^i b^i c^i \mid i \geq 0 \}$, $w = a^m b^m c^m$

- Escogiendo u, v, x, y, z según el lema, v e y pueden constar o sólo de as, bs o sólo de cs (si uno de ellos tuviera as, bs y bs y cs, la palabra uv^2xyx^2z tendría más de dos cambios).
- Si v e y constan sólo de as, bs o sólo de cs entonces cada uno de ellos está enteramente contenido en uno de los tres grupos de símbolos iguales que forman w.
- Pero entonces en uv^2xy^2z a lo sumo dos de los grupos de símbolos iguales que forman w habrán crecido, pero debe haber otro que se mantiene en k símbolos, por lo que uv^2xy^2z no está en L.

Lema del Bombeo

Por ejemplo: $L = \{ a^i b^i c^i \mid i \geq 0 \}$, $w = a^m b^m c^m$

Caso 1: vxy en contenido en las a's

Caso 2: vxy tiene a's & b's

Caso 3: vxy está en el bloque de b's

Caso 4: vxy tiene b's & c's

Caso 5: vxy está en el bloque de c's

Usar JFLAP para ver demostración:

http://www.jflap.com/

Propiedades de clausura

Propiedades cerradas: operaciones aplicadas a un conjunto cuyo resultado pertenece al mismo conjunto

Importancia: componer varios lenguajes de un tipo y obtener otro lenguaje

Propiedades de clausura

$L = \{ a^i b^j c^k \mid i < j < k \}$

Pueden verse dos lenguajes
$L_1 = \{ a^i b^j c^k \mid i < j \}$ es LLC
$L_2 = \{ a^i b^j c^k \mid j < k \}$ es LLC

$L = L_1 \cap L_2$ no es LLC porque la intersección no es cerrada

Propiedades de clausura

Propiedad	LR	LLC
U (unión)	S	S
∩ (intersección)	S	N
complemento	S	N
concatenación	S	S
∩ LR	S	S

Propiedades de clausura

Propiedad	**LR**	**LLC**
Kleene-clausura	S	S
Reflejo	S	S
Morfismo	S	S
Morfismo^{-1}	S	S
Diferencia	S	N

Propiedades de clausura

Unión, concatenación y clausura de Kleene:

Si G_1 y G_2 son GLC, $G_i = (V_i, T_i, P_i, S_i)$

- Unión: $L_1 \cup L_2 = L(G)$

$G = (V_1 \cup V_2, T_1 \cup T_2, P_1 \cup P_2 \cup \{S \rightarrow S_1 \mid S_2\}, S)$

- Concatenación: $L_1 L_2 = L(G)$

$G = (V_1 \cup V_2, T_1 \cup T_2, P_1 \cup P_2 \cup \{S \rightarrow S_1 S_2\}, S)$

- Clausura: $L_1^* = L(G)$

$G = (V_1, T_1, P_1 \cup \{S \rightarrow S_1 S \mid \lambda\}, S)$

Algoritmos de decisión

Problemas	LR	LLC
Equidad	D	N
Inclusión	D	N
Membresía	D	D
Vacuidad	D	D
Finitud	D	D

Algoritmos de decisión

Vacuidad: L_1 es vacio?. Dada G=(V,T,P,S)

L_1= L(G) es vacío: si S es inútil (S es derivable)

Finitud: L_1 es finito?. Dada G=(V,T,P,S)

L_1= L(G) es finito: si G es recursiva. Parte de una G simplificada, determinar a través de un digrafo las secuencias de derivaciones de G.

Si P contiene A → DB | C, D→ 1A, entonces el digrafo tiene los arcos {(A,D), (A,B), (A,C), (D,A)}. Si hay ciclos en el digrafo entonces es no es finito

Algoritmos de decisión

Membresía: w en Σ* esta en L_1?. Dada G=(V,T,P,S), si w es generado a partir de G

Proceso de Análisis o reconocimiento sintáctico

Se puede realizar con Algoritmos de parsing. Por ejemplo:

- Algoritmo de Cocke-Younger-Kasami (CYK)
- Algoritmo de Graham-Harrison-Ruzzo (GHR)
- Algoritmo de Earley

CYK

Algoritmo de Cocke-Younger-Kasami (CYK)

- La gramática G es una GLC en FNC.
- Las subcadenas de w, |w| = n, se identifican mediante su posición inicial y su longitud. p.e.: w_{ij} es la subcadena de w que comienza en la posición i y tiene una longitud j.
- El algoritmo construye conjunto de no terminales N_{ij} que generan las subcadenas w_{ij} de w. Si S esta en N_{1n}, entonces w esta en L(G).

CYK

Algoritmo de Cocke-Younger-Kasami (CYK)

- Algoritmo basado en programación dinámica conocido como algoritmo de “relleno de tabla” o “tabulación”.
- Teorema: El algoritmo CYK calcula correctamente N_{ij} para todo i,j, por lo tanto, w está en L(G) si y solo si S esta en N_{1n}. El tiempo de ejecución del algoritmo es O(n3)

CYK

Algoritmo: Entrada $w = w_1 w_2 \ldots w_n$, w en T* y G = (V,T,P,S)

Para i = 1..n hacer

1. $N_{i1} = \{ A \mid A \text{ en } V \text{ y } A \rightarrow w_{i1} \}$
2. Para j = 2..n hacer

   Parai = 1..(n-j+1) hacer

   a. Inicializa $N_{ij} = \varphi$

   b. Para K = 1..(j-1) añadir a N_{ij} todos los no terminales A para los que $A \rightarrow BC$, con B en N_{ik} y C en $N_{i+k,j-k}$
3. Si S en N_{1n} entonces w en L(G)

CYK

G = ({S,A,B,C},{0,1},P,S) en FNC

S → AB | BC

A → BA | 0

B → CC | 1

C → AB | 0

Determinar si:

¿W = 0010

esta en L(G)?

4				
3				
2				
1	N_{11}={A,C}	N_{21}={A,C}	N_{31}={B}	N_{41}={A,C}
	$w_1 = 0$	$w_2 = 0$	$w_3 = 1$	$w_4 = 0$

CYK

G = ({S,A,B,C},{0,1},P,S) en FNC

S → AB | BC

A → BA | 0

B → CC | 1

C → AB | 0

Determinar si:

¿W = 0010

esta en L(G)?

4				
3				
2	$N_{12}=\{B\}$	$N_{22}=\{S,C\}$	$N_{32}=\{A,S\}$	
1	$N_{11}=\{A,C\}$	$N_{21}=\{A,C\}$	$N_{31}=\{B\}$	$N_{41}=\{A,C\}$
	$a_1 = 0$	$a_2 = 0$	$a_3 = 1$	$a_4 = 0$

CYK

G = ({S,A,B,C},{0,1},P,S) en FNC

S → AB | BC

A → BA | 0

B → CC | 1

C → AB | 0

Determinar si:

¿W = 0010

esta en L(G)?

4				
3	$N_{13}=\{B\}$	$N_{23}=\{B\}$		
2	$N_{12}=\{B\}$	$N_{22}=\{S,C\}$	$N_{32}=\{A,S\}$	
1	$N_{11}=\{A,C\}$	$N_{21}=\{A,C\}$	$N_{31}=\{B\}$	$N_{41}=\{A,C\}$
	$a_1 = 0$	$a_2 = 0$	$a_3 = 1$	$a_4 = 0$

CYK

G = ({S,A,B,C},{0,1},P,S) en FNC

S → AB | BC

A → BA | 0

B → CC | 1

C → AB | 0

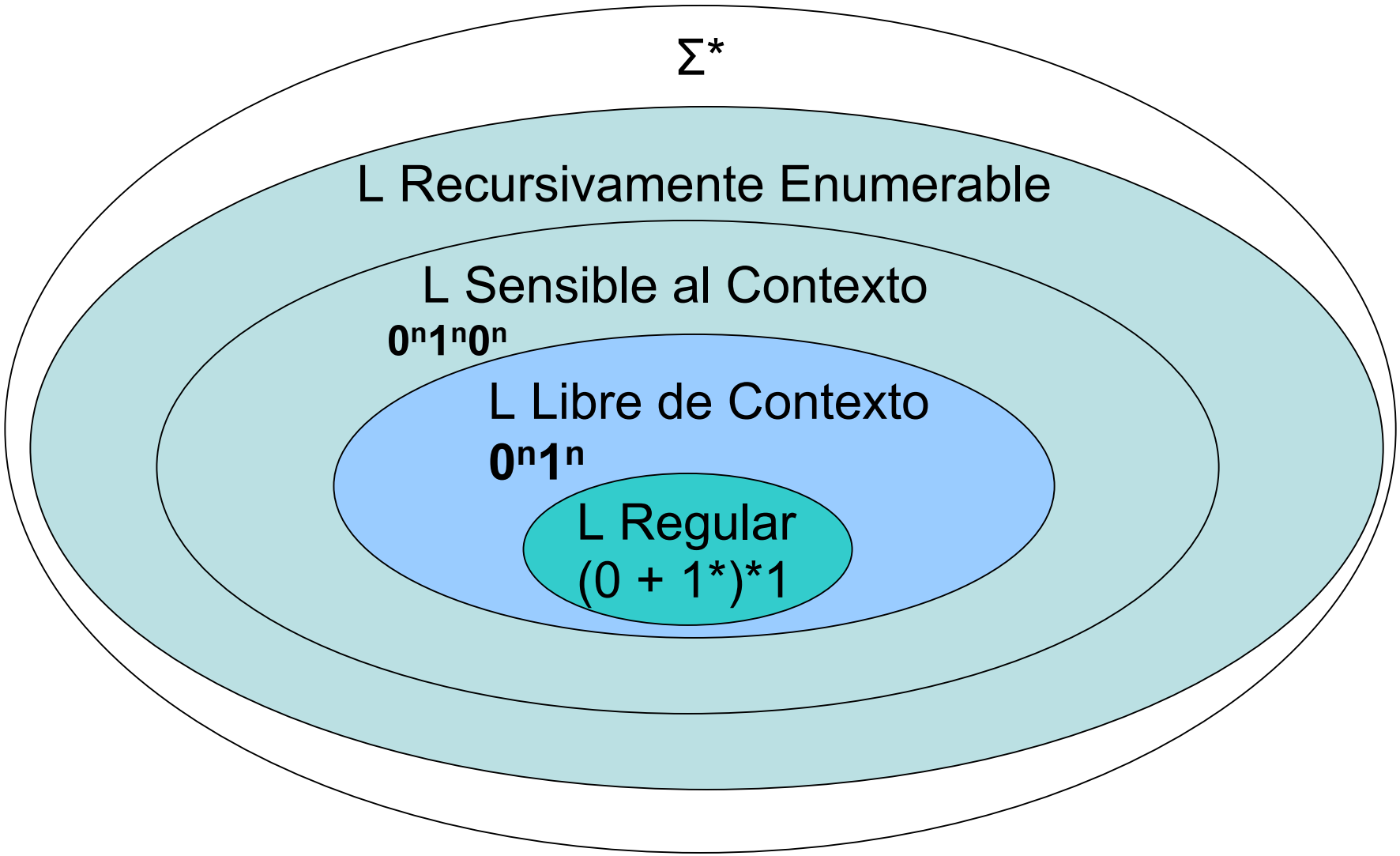
Determinar si:

¿W = 0010

esta en L(G)?

4	N_{14}={S,C,A}			
3	N_{13}={B}	N_{23}={B}		
2	N_{12}={B}	N_{22}={S,C}	N_{32}={A,S}	
1	N_{11}={A,C}	N_{21}={A,C}	N_{31}={B}	N_{41}={A,C}
	a_1 = 0	a_2 = 0	a_3 = 1	a_4 = 0

Jerarquía de los Lenguajes



Tipo de Lenguajes

Ejemplos de lenguajes:

- **Lenguaje de las expresiones regulares**

Con Σ = {a,b} → L = {λ,Φ,a, b, a*, b*, a+b+ λ, (a+b)*,(b+ λ), a*bb(ab+b)*b,…}

Análogo a: expresiones algebraicas (operaciones binarias de + y .) y 0^n1^n

- **Lenguajes de programación y Lenguajes de marcado**: HTML, XML, OWL, etc.

Con Σ caracteres gráficos → Lenguaje L estructurado con expresiones algebraicas y control de abrir y cerrar (llaves, paréntesis, condicionales, etiquetas, nodos).

Análogo a 0^n1^n

Lenguaje Libre de Contexto

GLC para el siguiente lenguaje:

$\mathbf{L_1 = \{ 0^n 1^n \mid n \geq 0 \}}$ y G = ({S},{0,1},P,S)

P = { S → λ , S → 0S1 } ó { S → λ | 0S1 }

Dado el homomorfismo: h(0) = (y h(1) =)

P = { S → λ | (S) }

Si se quiere validar las secuencias de paréntesis correctamente anidados p.e: "(()())(())"

G = ({S},{(,)},P,S) con P = { S → λ | (S) | SS }

Tipo de Lenguajes

Ejemplos de lenguajes:

- **Lenguas humanas**

Niveles de procesamiento lingüístico: léxico, sintáctico, semántico, contextual, pragmático.

- **Lenguajes biológicos**: ADN, ARN

Debe preguntarse si hay estructuras 0^n1^n, en caso contrario es regular. PROSITE asume que ADN es regular

- **Códigos**: Morse, ASCII

Si el código es una función homomórfica de un alfabeto a un lenguaje es regular. La letra “a” es “00001010”

Tipo de Lenguajes

Ejemplos de lenguajes:

- Lenguaje de las Gramáticas Libres de Contexto
- Lenguaje de la Música
- Lenguaje Latex
- Lenguaje de un protocolo de comunicación
- Etc.

Si no conoce el tipo de lenguaje

- Puede estar sub utilizando el modelo ➔ más recursos y costo cuando no lo necesita. “Solo necesita sumar y usa un computador en lugar de una calculadora”.
- El modelo esta limitado para procesar el lenguaje ➔ no se resuelve el problema. “Usa una calculadora para realizar transformaciones no algebraicas”.