>>> IF013 - Fundamentos Teóricos de Informática
>>> Licenciatura de Sistemas - UNPSJB - Sede Trelew

Name: Celia Cintas[†], Pablo Navarro[‡], Samuel Almonacid[§]
Date: August 23, 2017

[†]cintas@cenpat-conicet.gob.ar, cintas.celia@gmail.com, @RTFMCelia
[‡]pnavarro@cenpat-conicet.gob.ar, pablo1n7@gmail.com
[§]almonacid@cenpat-conicet.gob.ar, almonacid.samuel.tw@gmail.com

>>> Unidad 2

1. Gramáticas libres de contexto. Clasificación de Chomsky.
2. Relación entre gramáticas regulares y libres de contexto.
3. Árboles de derivación. Ambigüedad. Formas normales de Chomsky, Greibach y Backus-Naur.
4. Autómata a pila. No determinismo. Relación entre los autómatas a pila y las gramáticas libres de contexto.
5. Propiedades de los lenguajes libres de contexto.
6. Sustituciones y Homomorfismos.
7. Usos y aplicaciones de los lenguajes libres de contexto.
8. Sistemas de Lindenmayer. Sistemas-DOL. Sistemas-L Estocásticos.

Definición

Sean Σ_1 y Σ_2 dos alfabetos finitos, llamaremos sustitución a un mapeo $f_s : \Sigma_1 \to P(\Sigma_2^*)$. Una sustitución f_s es una función que asocia un lenguaje a cada símbolo de Σ. El mapeo f_s se extiende a cadenas de Σ_1^* como sigue:

$$f_c(\alpha) = \begin{cases} f_c(\lambda) = \lambda \\ f_c(x \bullet a) = f_c(x) \bullet f_s(a) \end{cases}$$

y a lenguajes definiendo $f_l(L)$ mediante: $f_l(L) = \cup f_c(\alpha)$, para toda cadena $\alpha \in L$.

>>> Propiedades de los Lenguajes Libres de Contexto (Cont.)

Definición

Una familia de lenguajes de tipo i se dice cerrada bajo sustitución si siempre que un lenguaje L es de tipo i y se aplica una sustitución f_l tal que $f_s(a)$ es de tipo i para cada $a \in \Sigma$, entonces $f_l(L)$ también es de tipo i.

Teorema

La clase de lenguajes Libres de Contexto es cerrada bajo sustitución.

>>> Propiedades de los Lenguajes Libres de Contexto (Cont.)

Los homomorfismos sobre cadenas, donde la función asocia a cada símbolo del alfabeto una palabra. Mas formalmente: $h|h(a) = \omega$ para cada a perteneciente a Σ_1 y alguna palabra ω perteneciente a Σ_2^*. $h : \Sigma_1 \to \Sigma_2^*$.

Corolario

La clase de lenguajes libres del contexto es cerrada bajo homomorfismo.

Ejemplo: Sea L el lenguaje de las cadenas de corchetes balanceados. Un homomorfismo para L_1 de las cadenas balanceadas de *begin, end* de Pascal es LC. Sea el homomorfismo $h : \{[,]\} \to \{begin, end\}$ tal que:

$$h(\alpha) = \begin{cases} h([) = begin \\ h(]) = end \end{cases}$$

Teorema

Los lenguajes libres del contexto son cerrados bajo unión, concatenación, estrella de Kleene e intersección con un lenguaje regular.

Teorema

Si L_1 y L_2 son dos lenguajes libres de contexto, entonces $L_1 \cap L_2$ no siempre es un LLC.

Demostración

$L_1 = \{a^n b^m c^j | n, m, j \geq 0 \; y \; n = m\}$

$L_2 = \{a^n b^m c^j | n, m, j \geq 0 \; y \; m = j\}$

Podemos encontrar G_1 y G_2 tal que $L_1 = L(G_1)$ y $L_2 = L(G_2)$.

Pero:

$L_1 \cap L_2 = \{a^n b^m c^j | n, m, j \geq 0 \; y \; n = m = l\}$

Este lenguaje no es LC, el cual puede chequearse con el lema de bombeo.

Teorema

La clase de Lenguajes Libres del Contexto no es cerrada bajo complemento.

Teorema

Si L_1 y L_2 son dos lenguajes libres de contexto, entonces $L_1 \cup L_2$ lo es también.

Demostración

Como L_1 y L_2 son LLC, existen dos GLC: $G_1 = (V_{N_1}, \Sigma, P_1, S_1)$ y $G_2 = (V_{N_2}, \Sigma, P_2, S_2)$. Tal que $L_1 = L(G_1)$ y $L_2 = L(G_2)$. Asumimos que $V_{N_1} \cap V_{N_2} = \emptyset$.
Entonces G:
$G = (V_{N_1} \cup V_{N_2} \cup \{S\}, \Sigma, P_1 \cup P_2 \cup \{S \to S_1, S \to S_2\}, S)$
Puede demostrarse fácilmente que $\forall \alpha \in \Sigma^*, \alpha \in L(G)$ si y solo si $\alpha \in L(G_1) \cup L(G_2)$.

Teorema

Si L_1 y L_2 son dos lenguajes libres de contexto, entonces $L_1 L_2$ lo es también.

Demostración

Como L_1 y L_2 son LLC, existen dos GLC: $G_1 = (V_{N_1}, \Sigma, P_1, S_1)$ y $G_2 = (V_{N_2}, \Sigma, P_2, S_2)$. Tal que $L_1 = L(G_1)$ y $L_2 = L(G_2)$. Asumimos que $V_{N_1} \cap V_{N_2} = \emptyset$.

Entonces G:

$G = (V_{N_1} \cup V_{N_2} \cup \{S\}, \Sigma, P_1 \cup P_2 \cup \{S \to S_1 S_2\}, S)$

Puede demostrarse fácilmente que $\forall \alpha \in \Sigma^*, \alpha \in L(G)$ si y solo si $\alpha \in L(G_1)L(G_2)$.

Teorema

Si L_1 es un lenguaje libre de contexto, entonces L_1^+ también lo es.

Demostración

Como L_1 es LLC, existe una GLC: $G_1 = (V_{N_1}, \Sigma, P_1, S_1)$. Tal que $L_1 = L(G_1)$.

Entonces G:

$G = (V_{N_1} \cup \{S\}, \Sigma, P_1 \cup \{S \to S_1 S, S \to S_1\}, S)$

Puede demostrarse fácilmente que $\forall \alpha \in \Sigma^*, \alpha \in L(G)$ si y solo si $\alpha \in L(G_1)^+$.

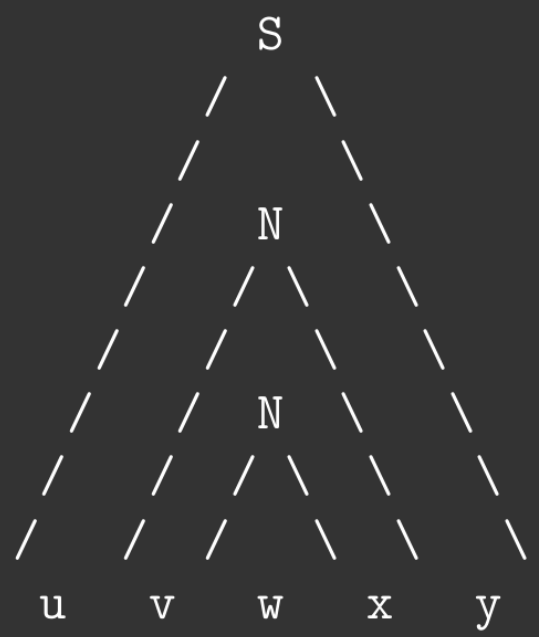
Teorema

("Pumping" para LLC) Sea L un lenguaje libre de contexto infinito. Entonces existe una constante $k, k \in \mathbb{Z}^+$, dependiente del lenguaje L, tal que para toda cadena $z \in L$, si $long(z) > k$ entonces z puede ser escrita de la forma $z = uvwxy$, de forma que $vx \neq \lambda$, $long(vwx) \leq k$ y $uv^n wx^n y \in L$ para todo $n > 0$.

>>> Pumping Lemma para LLC (cont.)

Demostración

Sea una gramática $G = (V_n, V_t, S, P)$ tal que $|V_n| = m$ y $p = max\{long(\alpha) | A \to \alpha \in P\}$. Si $z \in L(G)$ y $|z| > k = p^m$ existe un árbol de derivación de z tal que contiene un camino en el cual algún símbolo no terminal aparece mas de una vez.



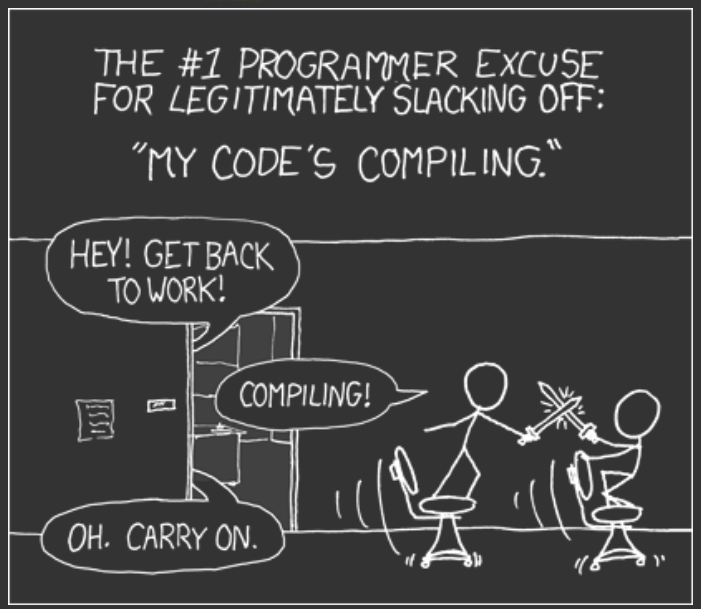
De acuerdo a la longitud de la cadena z será la cantidad de usos del no terminal N y en consecuencia la cantidad de repeticiones de v e x conduciendo a la forma esperada: $uvwxy$.

>>> Pumping Lemma para LLC (cont.)

Es $L = \{a^n b^n c^n | n \geq 0\}$ libre de contexto?
Sea $z = a^k b^k c^k$ una cadena, por el teorema de pumping, existe una $z = uvwxy$, con $vx \neq \lambda$, $long(vwx) \leq k$ tal que $uv^n wx^n y \in L$ para todo $n \geq 0$. Los casos posibles son:

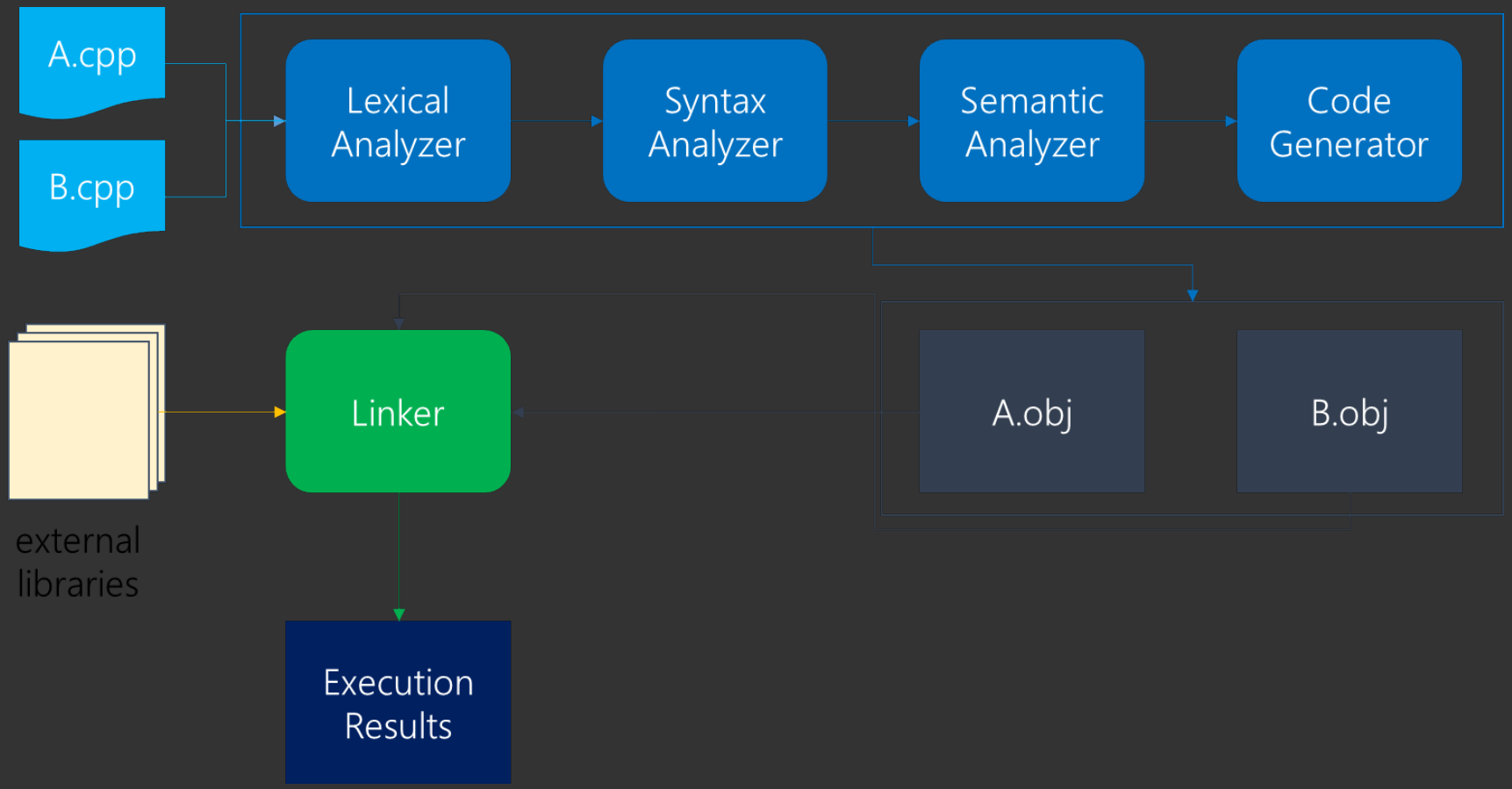
1. v o x contienen dos tipos de símbolos, por ejemplo $a^p b^q$, aplicando potencia n se pierde la estructura.
2. v y x contienen un sólo tipo de símbolos. Consideremos: $v = a^p$ y $x = a^q, v = b^p$ y $x = b^q$ o $v = c^p$ y $x = c^q$. Entonces, en $uv^n wx^n y$ quedará un número mayor de un cierto tipo de símbolos que de los otros dos. Si $v = a^p$ y $x = b^q$ o $v = b^p$ y $x = c^q$ quedará un número mayor de dos símbolos que del restante.

>>> Compiladores



http://talks.florian-rappl.de/Compiler/

>>> Compiladores (Cont.)



http://talks.florian-rappl.de/Compiler/

LL derivation

rule	read	↓stack	remaining
		S	1+2*3
$S \to E$		E	1+2*3
$E \to TP$		TP	1+2*3
$T \to FM$		FMP	1+2*3
$F \to N$		NMP	1+2*3
$N \to 1$		$1MP$	1+2*3
read	1	MP	+2*3
$M \to \epsilon$	1	P	+2*3
$P \to +E$	1	$+E$	+2*3
read	1+	E	2*3
$E \to TP$	1+	TP	2*3
$T \to FM$	1+	FMP	2*3
$F \to N$	1+	NMP	2*3
$N \to 2$	1+	$2MP$	2*3
read	1+2	MP	*3
$M \to *T$	1+2	$*TP$	*3
read	1+2*	TP	3
$T \to FM$	1+2*	FMP	3
$F \to N$	1+2*	NMP	3
$N \to 3$	1+2*	$3MP$	3
read	1+2*3	MP	
$M \to \epsilon$	1+2*3	P	
$P \to \epsilon$	1+2*3		

LR derivation

rule	read	stack↓	remaining
			1+2*3
shift	1	1	+2*3
$N \to 1$	1	N	+2*3
$F \to N$	1	F	+2*3
$M \to \epsilon$	1	FM	+2*3
$T \to FM$	1	T	+2*3
shift	1+	$T+$	2*3
shift	1+2	$T+2$	*3
$N \to 2$	1+2	$T+N$	*3
$F \to N$	1+2	$T+F$	*3
shift	1+2*	$T+F*$	3
shift	1+2*3	$T+F*3$	
$N \to 3$	1+2*3	$T+F*N$	
$F \to N$	1+2*3	$T+F*F$	
$M \to \epsilon$	1+2*3	$T+F*FM$	
$T \to FM$	1+2*3	$T+F*T$	
$M \to *T$	1+2*3	$T+FM$	
$T \to FM$	1+2*3	$T+T$	
$P \to \epsilon$	1+2*3	$T+TP$	
$E \to TP$	1+2*3	$T+E$	
$P \to +E$	1+2*3	TP	
$E \to TP$	1+2*3	E	
$S \to E$	1+2*3	S	

Definición

Los LLC que se pueden reconocer con la construcción descrita arriba se llaman LL(1). Si se pueden reconocer mediante mirar de antemano los k caracteres de la entrada se llaman $LL(k)$.

Este tipo de parsing se llama "top-down" porque se puede visualizar como generando el árbol de derivación desde arriba hacia abajo, pues decidimos qué producción utilizar (es decir la raíz del árbol) antes de ver la cadena que se derivará.

Qué autómatas usarían para realizar Parsing LL(k)?

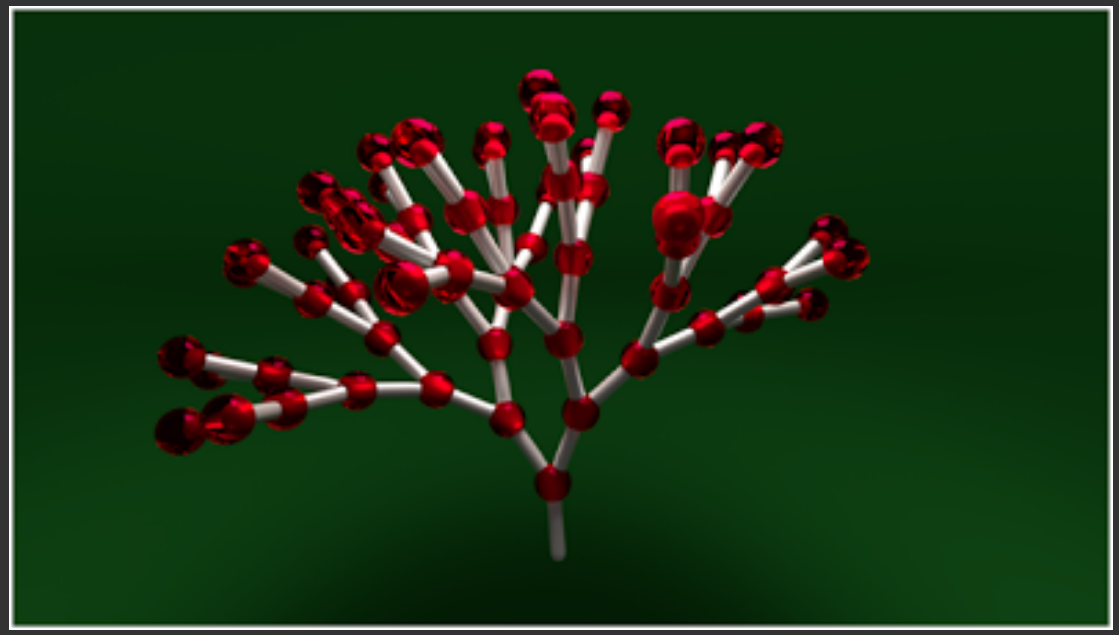
El parsing $LR(k)$ es más flexible. La idea esta vez es construir el árbol de parsing de abajo hacia arriba. La idea de este parsing es que la pila contiene lo que el parser ha visto de la entrada, no lo que espera ver.

Qué autómatas usarían para realizar Parsing LR(k)?

Definición

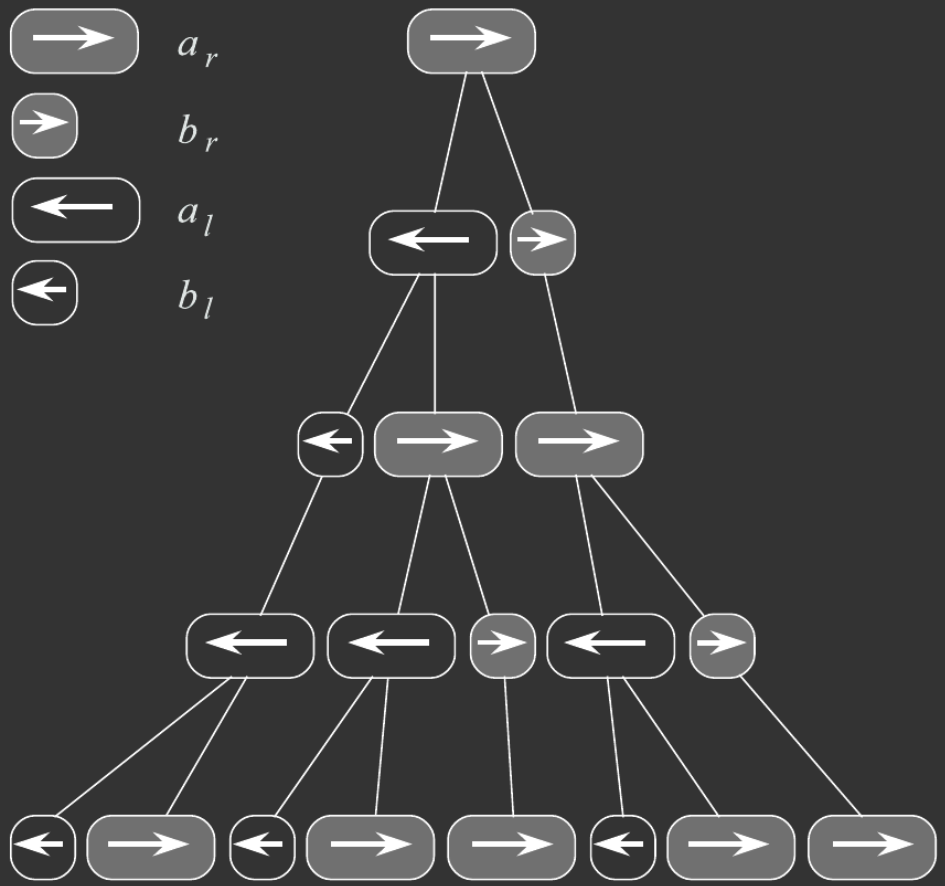
Sea un Sistema-L definido cómo: $G = \{V, S, \omega, P\}$, donde:

* V Conjunto de Símbolos No terminales.
* S Conjunto de Símbolos Terminales.
* ω cadena compuesta por elementos de V que define el axioma.
* P Conjunto de producciones.



http://michelanders.blogspot.com.ar/p/creating-blender-26-python-add-on.html#exercises

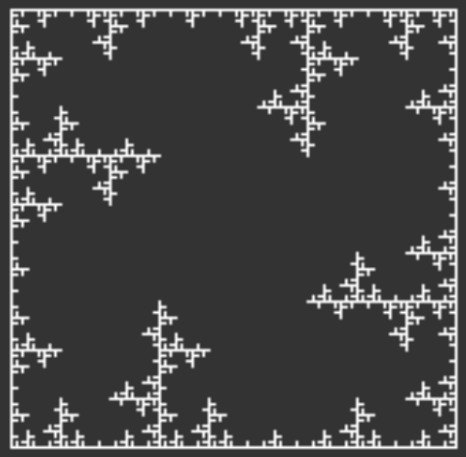
Sistemas-DOL



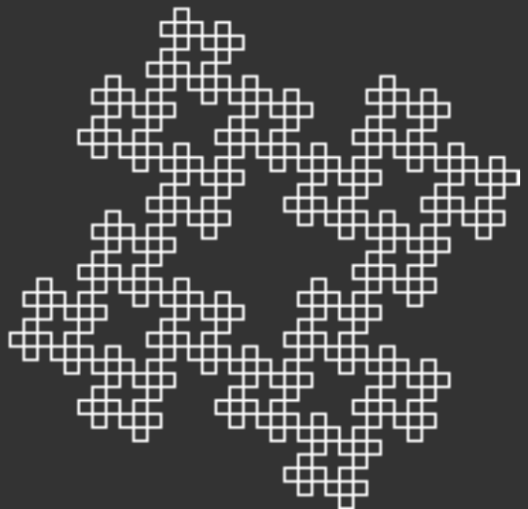
$$\omega : a_r$$
$$p_1 : a_r \rightarrow a_l b_r$$
$$p_2 : a_l \rightarrow b_l a_r$$
$$p_3 : b_r \rightarrow a_r$$
$$p_4 : b_l \rightarrow a_l$$

>>> Gramáticas de Lindenmayer (Cont.)

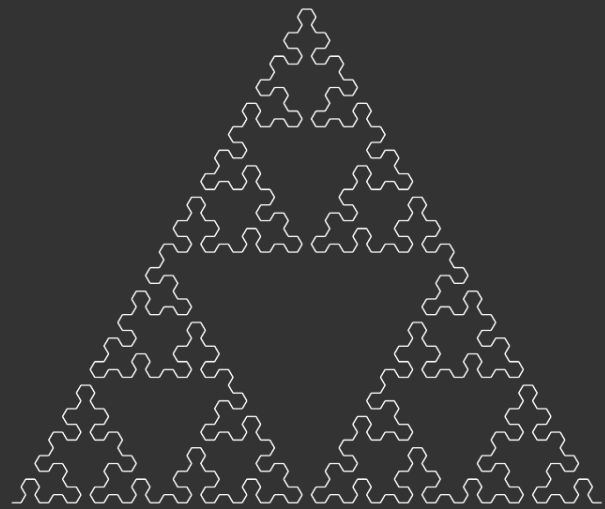
Sistemas-DOL



$n = 4,\ \delta = 90^\circ$
F-F-F-F
$F \rightarrow FF\text{-}F\text{--}F\text{-}F$



$n = 4,\ \delta = 90^\circ$
F-F-F-F
$F \rightarrow F\text{-}F+F\text{-}F\text{-}F$



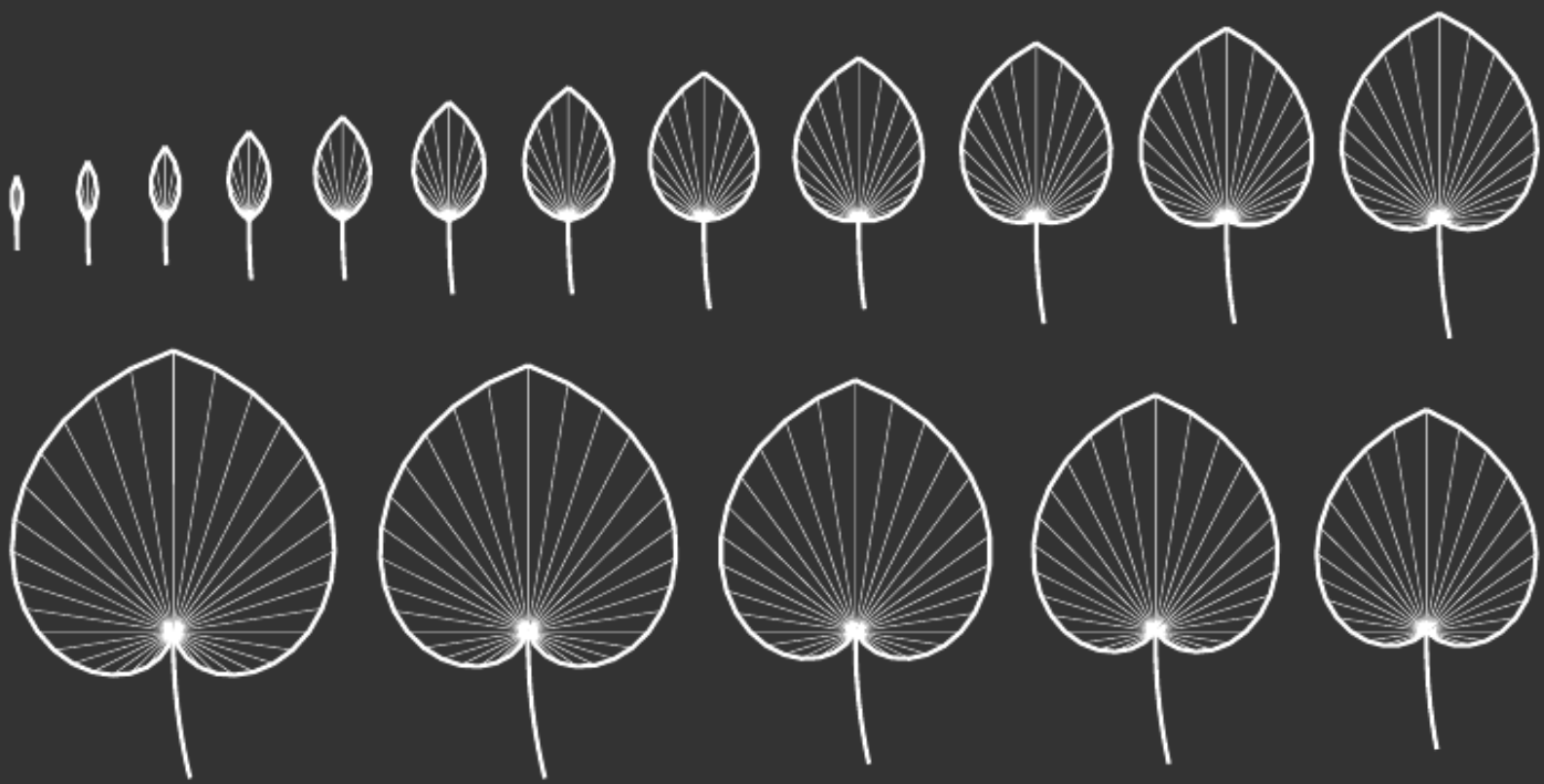
$n=6,\ \delta=60^\circ$
F_r
$F_l \rightarrow F_r+F_l+F_r$
$F_r \rightarrow F_l-F_r-F_l$

Sistemas-L Estocásticos

$$\omega : F$$
$$p_1 : F \xrightarrow{.33} F[+F]F[-F]F$$
$$p_2 : F \xrightarrow{.33} F[+F]F$$
$$p_3 : F \xrightarrow{.34} F[-F]F$$

>>> Gramáticas de Lindenmayer (Cont.)

Simulación de crecimiento con Sistemas-L



$$\omega : [A][B]$$
$$p_1 : A \rightarrow [+A\{.].C.\}$$
$$p_2 : B \rightarrow [-B\{.].C.\}$$
$$p_3 : C \rightarrow GC$$

>>> Demo Time!

Sistemas-L en Python - Ver ipynb

>>> Gracias!

Bibliografía

1. The Algorithmic Beauty of Plants, Przemyslaw Prusinkiewicz, Aristid Lindenmayer 2004.
2. Introduction to Automata Theory, Languages, and Computation - Hopcroft et. al 2007 (3er ed.)
3. Teoría de la Computación - Gonzalo Navarro 2011.
4. Fundamentos de Cs. de la Computación - Juan Carlos Augusto 1995.