Compiladores e Intérpretes

San José

Grupo: 40

Apuntes del 28/04/2017

Profesor: Dr. Francisco Torres Rojas

Apuntador: Dylan Rodríguez Barboza

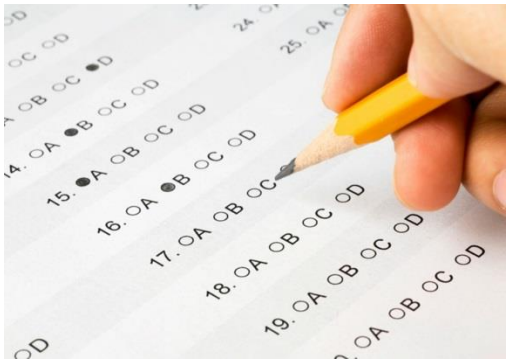
2015057714

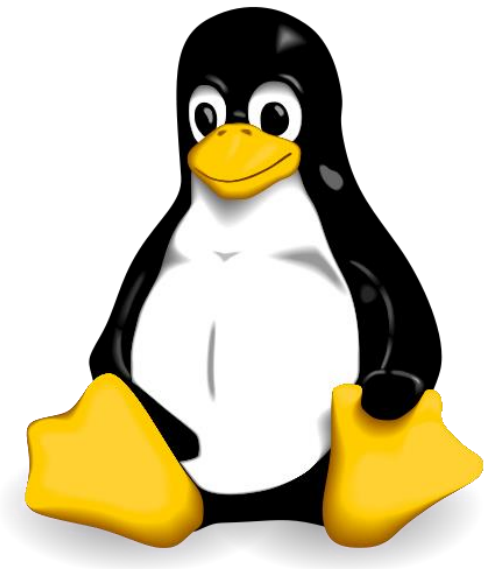
I Semestre, 2017

Contenido

Recordatorios

- ✓ Hay un nuevo desglose basado en las correcciones que se hicieron para el examen.
- ✓ Habrá quiz el miércoles 3 de mayo.
- ✓ El proyecto 2 se entrega y revisa el viernes 5 de mayo.





Repaso de la clase pasada: Análisis Sintáctico

Análisis sintáctico

- ✓ Se revisa que la sintaxis de un programa está correcta.
- ✓ Se le conoce en el bajo mundo como **parser**.
- ✓ Toma secuencias de tokens y verifica que correspondan a construcciones válidas.
- ✓ Requiere de una gramática.
- ✓ Hay muchos algoritmos de parsing.

Tipos de parsing

- ✓ Top down parsing.
- ✓ Bottom up parsing.

Errores de sintaxis

- ✓ El manejo de errores es muy difícil en el análisis sintáctico.
- ✓ En el análisis léxico es relativamente fácil.
- ✓ El parser debe hacer muchas cosas frente a un error, a esto lo llamaremos **triple R**.

Triple R

Reportar

- ✓ Es encontrar que hay un error, es automático con los algoritmos de parsing.
- ✓ Se sabe que la hilera está mala.
- ✓ Localización del error: posición exacta del token es difícil. Puede ser la ausencia de un token.
- ✓ Mensaje de error: en el peor de los casos, solo se imprimirá un error que diga “error de sintaxis”. La reducción apropiada es difícil.

Recuperar

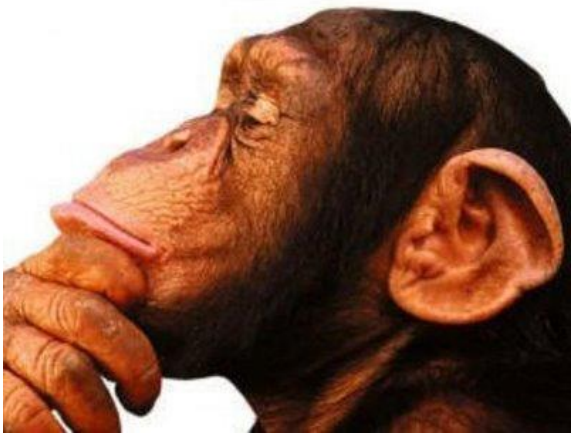
- ✓ Con un único error, ya toda la hilera está mal. Todo el programa está mal.
- ✓ Es impráctico que se reporte solo el primer error.
- ✓ El parser debe recuperarse para seguir analizando el resto del programa.
- ✓ Avanzar hasta “un punto seguro” y continuar el proceso del parsing.
- ✓ ¿Cuáles son los problemas? Saltar demasiado, cascada de errores.

Reparar

- ✓ “Si el parser “sabe” que tiene malo el programa ¿Por qué no lo arregla?” –Ariel
- ✓ A veces se necesita como parte de la recuperación, por ejemplo, que el compilador declare una variable que necesite, se declara (tabla de símbolos), para poder seguir compilando, es una reparación que no pretende que el programa corra, pero es solo para poder terminar el proceso.
- ✓ Los warning pueden ser resultado de una reparación, por ejemplo, rutinas de conversión hechas por el compilador para seguir compilando.
- ✓ ¿Realmente qué es eso?
- ✓ Muchas explicaciones posibles de un error, implican muchas reparaciones posibles.

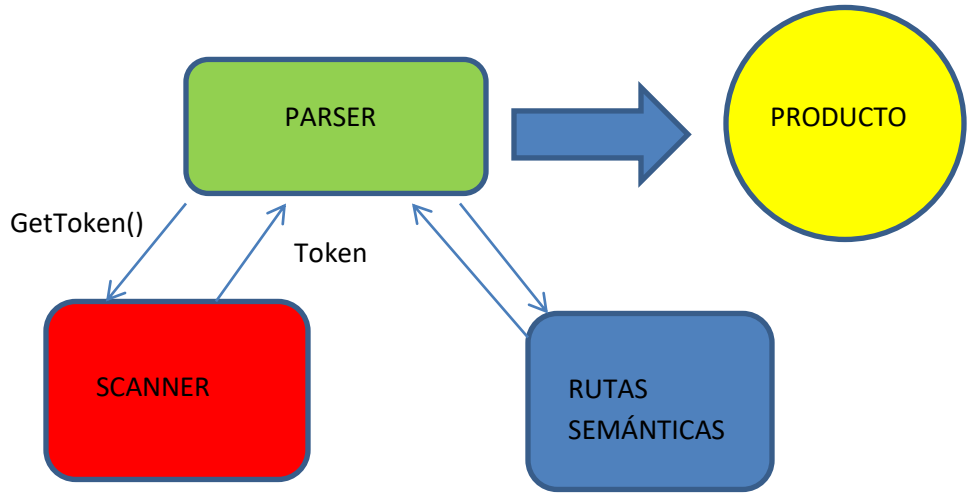
Momento para pensar, con el profe

- ✓ “Los novatos piensan: ¿Por qué si me dice que falta ‘;’, no lo pone solamente y ya?”
- ✓ No saben con certeza absoluta en dónde está el error.
- ✓ “¿Realmente quieren que el compilador arregle su programa?”
- ✓ Se hace un tipo de reparación para seguir compilando.
- ✓ ¿Por qué no lo repara? Hay muchas interpretaciones posibles para un solo error.

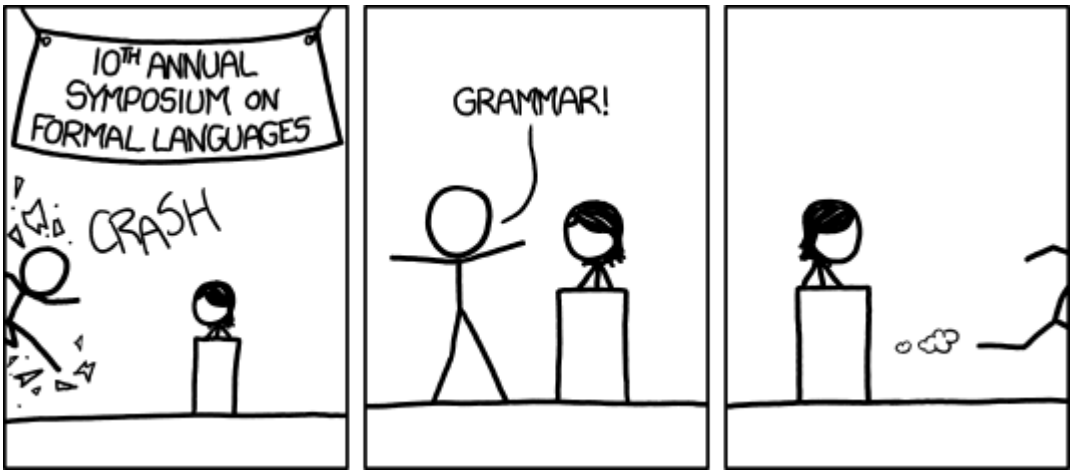


Traducción dirigía por sintaxis

- ✓ Organización más usual en compiladores.
- ✓ El parser dirige todo el proceso: invocar al scanner cada vez que necesita a un token. Invocar rutinas semánticas en los puntos apropiados.
- ✓ Su trabajo podría terminar al generar código intermedio o, inclusive, podría generar lenguaje máquina.
- ✓ El parser es el cliente del scanner (getToken()).
- ✓ El parser puede invocar rutinas semánticas en las partes apropiadas.
- ✓ En bison, por ejemplo, las rutinas se llaman dentro de “{}”.
- ✓ El proyecto 2 tiene relación con esto.



Gramáticas Libres de Contexto

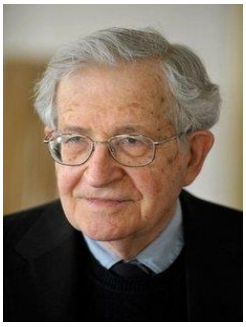


Avram Noam Chomsky

- ✓ Lingüista, filósofo, científico social, U.S.A. Nacido en 1928.
- ✓ "Padre de la lingüística".
- ✓ Profesor emérito M.I.T.
- ✓ Unas de las figuras culturales más importantes de tiempos recientes.
- ✓ De los autores más citados.
- ✓ Posiciones políticas antibélicas.

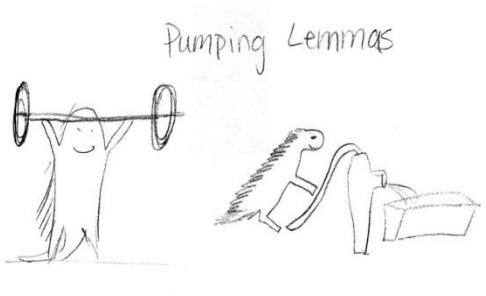
"Súper peleón. Debe de tener el hígado en pedazos con Trump como presidente." – Torres

“Profe, se parece a Chomsky”



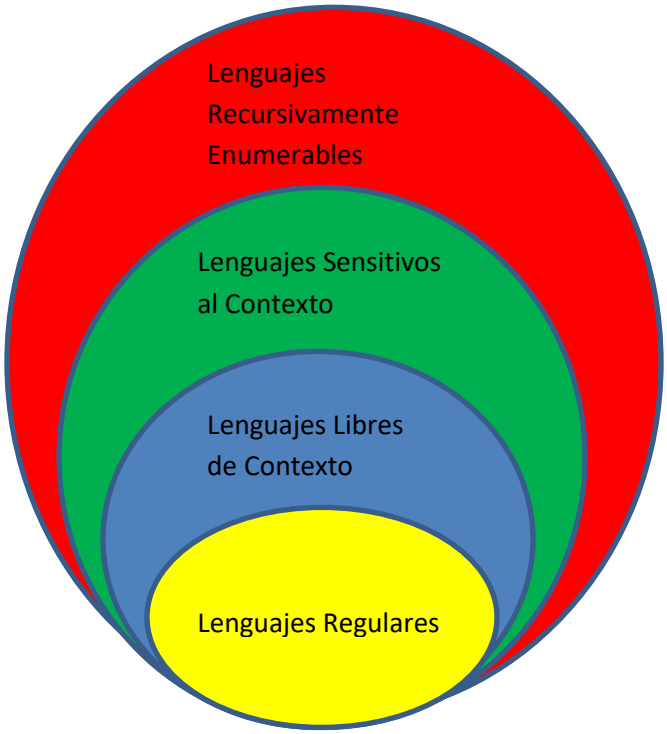
Lenguajes no regulares

- ✓ El pumping lemma nos permitió demostrar que algunos lenguajes no son regulares.
- ✓ Sea L el lenguaje sobre ∑={a,b} definido como $\{a^n b^n | n \geq 0\}$.
- ✓ ¿Podrán ser generados y reconocidos por algún mecanismo?
- ✓ ¿Cómo se relacionan entre sí los lenguajes?
- ✓ Por medio de la **Jerarquía de Chomsky**.



Jerarquía de Chomsky

- ✓ Todos los lenguajes regulares son libres de contexto, no todos los lenguajes libres de contexto son lenguajes regulares.



Gramática libre de contexto

Esto es una gramática libre de contexto:

$$E^1 \rightarrow E+E^2$$

$$E \rightarrow E*E$$

$$E^3 \rightarrow a$$

$$E \rightarrow b$$

1: Es el símbolo inicial.

2: Terminales, pueden ir en la parte izquierda de cada regla.

3: No terminales, pueden ir a la izquierda y derecha de cada regla.

A todos los elementos se les llama reglas.

Hileras derivables de abba: {abba}.

Definición formal

Una gramática libre de contexto (CFG) es un cuarteto G=(V, ∑,P,S), donde:

- ✓ V es un conjunto finito de no terminales.
- ✓ ∑ es un conjunto finito de terminales.
- ✓ P es un conjunto finito de reglas.
- ✓ S∈V es el símbolo inicial.

V∩∑ = ∅ (terminales y no terminales son conjuntos disjuntos).

P es un subconjunto que incluye Vx(V∪ ∑)*, las cuales son hileras con terminales y no terminales de cualquier longitud. Por ejemplo, tenemos un par (No terminal, w), el cual podría resultar en (E, a+E), dando como resultado: **E -> a+E**.

Las reglas tienen un único no terminal a la izquierda, una hilera de terminales y no terminales a la derecha. A la derecha podría estar ε.

Derivación

- ✓ Sean A, B y C hileras tomadas de (VU $\sum$)*.
- ✓ Si se tiene una hilera w de la forma ATB y hay una regla en la gramática de la forma T -> C, entonces w se puede reescribir como ACB .
- ✓ Decimos que w deriva de la hilera ACB y se denota como w⇒ ACB.
- ✓ Se puede derivar varias veces (mientras haya no terminales).
- ✓ Para aplicar una regla solo interesa la presencia del no terminal que se va a expandir. El contexto que lo rodea no importa.
- ✓ **Gramática libre de contexto.**

Notación

- ✓ Un paso de derivación: w⇒ v.
- ✓ Uno o más pasos de derivación: w⇒$^+$v.
- ✓ Cero o más pasos de derivación: w⇒*v.

Derivación de w

Sea G=(V, ∑,P,S) una CFG y sea w ∈(VU ∑)*.

El conjunto de hileras derivables de w se define recursivamente como:

- ✓ w es derivable de w.
- ✓ Si v=ATB es derivable desde w y T->C ∈ P entonces ACB es derivable desde w.
- ✓ Todas y únicamente las hileras construidas aplicando el paso 2 son derivables desde w.

Hileras derivables desde w={v|w⇒*v}.

Ejemplo

Sea G definida como:

E -> E + E

E -> E * E

E -> a

E –> b

- ✓ Hileras derivables de abba: {abba}.
- ✓ Hileras derivables de *EaE: {*EaE, *aaa, *E+EaE, *E+E+EaE*E,…..}.

Definiciones

Sea G=(V, ∑,P,S) una CFG:

- ✓ Una hilera $w \in (V \cup \sum)^*$ es una forma sentencial de G, si y solo sí $S \Rightarrow^* w$.
- ✓ Una hilera $w \in \sum^*$ es una sentencia de G, sí y solo sí $S \Rightarrow^* w$.
- ✓ El lenguaje G, denotado como L(G) es el conjunto $\{v | v \in \sum^* \text{ y } S \Rightarrow^* v\}$.
- ✓ El lenguaje G es el conjunto de todas las sentencias de G.
- ✓ Lenguaje: conjunto de sentencias.

Ejemplo

Sea G definida como:

E -> E + E

E -> E * E

E -> a

E –> b

Formas sentenciales de G: {E, E+E, E+E*E, E*E, a, a+a, b,….}

Sentencias de G:{a, b, a+a, a+b, a*a, b+a*b, b+b+b+b, ……}

Más definiciones

- ✓ Una derivación es leftmost (más izquierda), si siempre se reemplaza el no terminal que esté más a la izquierda en la hilera actual.
- ✓ Una derivación es rightmost (más derecha), si siempre se reemplaza el no terminal que esté más a la derecha en la hilera actual.

Ejemplo de derivación leftmost

Sea G definido como:

E -> E + E

E -> E * E

E -> a

E –> b

E

⇒**E***E

⇒**E**+E*E

⇒b+**E***E

⇒b+**E***E*E

⇒b+a***E***E

⇒b+a*a***E**

⇒b+a*a*b

Ejemplo de derivación rightmost

E -> E + E

E -> E * E

E -> a

E –> b

E

⇒E*__E__

⇒E+E*__E__

⇒E+E*E+__E__

⇒E+E*__E__+b

⇒E+E*E+__E__+b

⇒E+E*__E__+b+b

⇒E+__E__*a+b+b

⇒__E__+a*a+b+b

⇒b+a*a+b+b

Ejemplos de ~~GFG (Gontext Free Grammar)~~ CFG (Context Free Grammar)

Definición

Un conjunto de hileras es un lenguaje libre de contexto si es completamente generado por una gramática libre de contexto G=(V, ∑,P,S), además G no genera hileras que no pertenezcan al lenguaje.

Ejemplo 1

Presente una CFG que genere el lenguaje de hileras de 1 o más “a”

Solución:

A -> aA

A –> a

Ejemplo 2

Presente una CFG que genere el lenguaje de hileras de 0 o más “a”

Solución:

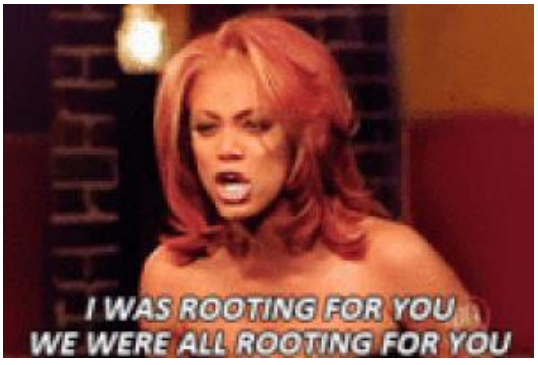
A -> aA

A -> ε

Ejemplo 3

Presenta una CFA que genere el lenguajes de hileras sobre ∑={0,1} que terminen en 1

En este ejercicio hubo muchos intentos, pero casi todos fallidos..



Solución:

S->A1

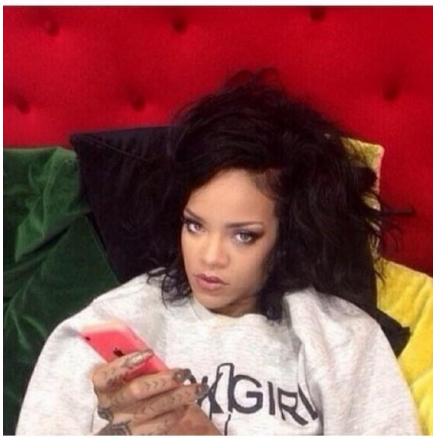
A -> 0A

A -> 1A

A-> ε

La última regla, según el profe se puede entender como: “Alajuela épsilon”.

Y todos así:



Ejemplo 4

Presente una CFG que genere el lenguaje de hileras sobre ∑={0,1} que empiecen con 0 y terminen con 1

Solución:

S -> 0E1

E -> 0E

E -> 1E

E -> ε

Ejemplo 5

Presente una CFA que genere el lenguaje $\{a^n b^m | a>0, m>0\}$

Solución:

S -> AB

A -> aA|a

B-> bB|b

Ejemplo 6

Presente una CFA que genere el lenguaje $\{a^n b^m | a \geq 0, m \geq 0\}$

Solución:

S -> AB

A -> aA| ε

B-> bB| ε

Ejemplo 7

Sea G definida como:

S -> aSa

S -> aBa

B -> bB

B -> b

¿Cuál es el lenguaje de G?

Solución:

$\{a^n b^m a^m, n>0, m>0\}$

Ejemplo 8

Presente una CFG que genere el lenguaje $\{a^n b^m c^m d^{2n} \mid n \geq 0,\ m>0\}$

De este ejercicio dependía que todos fuéramos a almorzar, resuelto por **Ariana.**

Solución:

S -> aSbb

S ->B

B -> bBc

B -> bc

Preguntar frecuentes con el profe

- ✓ ¿Cómo se sabe cuál regla tomar? –Se toma la correcta.
- ✓ ¿Un no terminal puede ser una forma sentencial? –Sí.
- ✓ ¿Cómo definir reglas? –Con mucho cuidado.
- ✓ ¿Qué propiedad tiene el símbolo inicial? –Que es el inicial.

Esto fue todo lo referente a la clase del 28 de abril del 2017.