



Derivaciones y Análisis sintáctico

- Derivación izquierda [**derecha**]: el próximo no terminal a expandir es el que se encuentre más a la izquierda [**derecha**]
- Un parser top-down parte del axioma y despliega el AAS en concordancia con una derivación izquierda
- Un parser bottom-up va reduciendo desde las hojas hacia el axioma, en concordancia con el orden inverso de una derivación derecha



Gramáticas Ambiguas

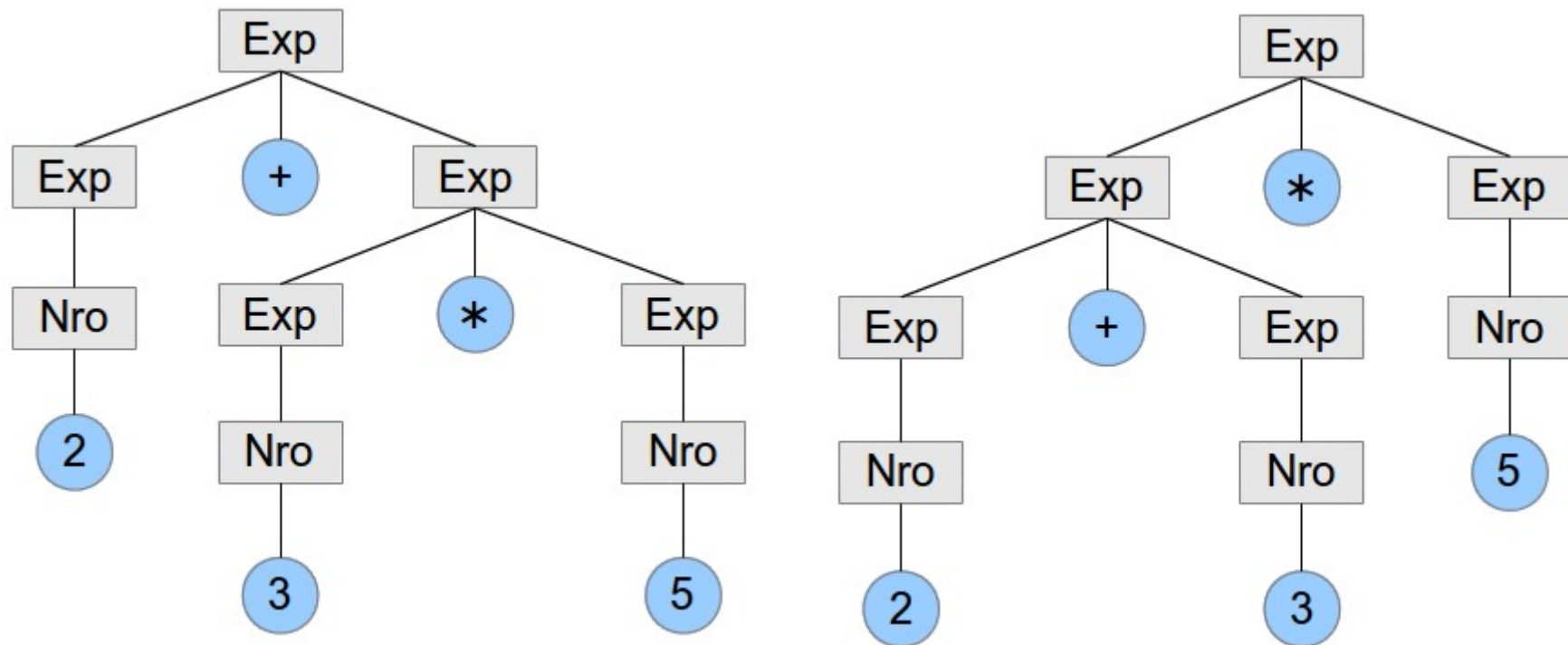
- Una gramática es ambigua si hay dos posibles derivaciones del mismo tipo (izquierda o derecha) para la misma cadena
- También podemos reconocerlas por poder armar dos AAS para la misma cadena
- Sea la gramática

Regla Nro	Regla
1	$\text{Exp} \rightarrow \text{Exp} + \text{Exp}$
2	$\text{Exp} \rightarrow \text{Exp} * \text{Exp}$
3	$\text{Expr} \rightarrow \text{Nro}$
4	$\text{Nro} \rightarrow 0 \mid 1 \mid 2 \mid 3 \mid 4 \mid 5$



Ejemplo Ambiguas

Cadena: $2 + 3 * 5$





Determinación de Ambigüedad

- Determinar si una GIC es ambigua es un problema indecidible, es decir, no existe algoritmo que tomando cualquier GIC conteste por sí o no.
- Sin embargo si es posible determinarlo para algunas clases de gramáticas
- Para demostrar que es ambigua basta con encontrar una cadena que admita dos árboles de derivación diferentes.
- Demostrar que es no ambigua se complica más, hay que demostrar que en ningún caso voy a poder armar dos árboles de derivación diferentes
- Es sabido que las gramáticas $LL(k)$ y $LR(k)$ que veremos a continuación no son ambiguas



LL

- Un análisis sintáctico LL (**L**eft to right, **L**eftmost derivation) significa que se hace recorriendo los tokens de izquierda a derecha y derivando a izquierda
 - Es lo que se usa en el ASDR (top-down)
 - Implica desplegar el AAS desde la raíz, primero en profundidad, en pre orden
- Una gramática LL es una GIC que **admite un análisis sintáctico LL**
- Se dice que es LL(**k**) si puedo decidir que producción usar conociendo anticipadamente los próximos **k** símbolos de preanálisis (tokens)
- Un parser LL(**k**) es conocido también como un parser predictivo (elimina la necesidad de backtracking gracias a los **k** símbolos de preanálisis)



LR

- El caso LR (**L**eft to right, reversed **R**ightmost derivation) se usa en parsers bottom-up
- Invierte una derivación a derecha, es decir, reduce primero la última derivación y va armando el AAS hacía su axioma.
- Lo habitual es implementarlos usando una herramienta tipo bison en lugar de programarlos a mano
- Al igual que con los LL se usan símbolos de preanálisis, teniendo gramáticas LR(k)
- Reconocen un grupo más amplio de lenguajes que las gramáticas LL
- Permite una mejor (más temprana) detección de errores que el análisis sintáctico LL



LL(1)

- Un caso de interés es el LL(1) ya que permite un algoritmo muy eficiente que decide que producción usar con un único **símbolo de preanálisis**
- Obviamente no todas las GIC son LL(1)
- Ejemplos
 - $S \rightarrow aR \mid b$ es LL(1)
 - $S \rightarrow aR \mid a$ NO es LL(1)
 - $S \rightarrow abQ \mid acM$ NO es LL(1) , es LL(2)



Recursividad a Izquierda

- Supongamos la producción: $\text{expr} \rightarrow \text{expr} + \text{term} \mid \text{term}$
- Como vimos los no terminales llaman a un PAS
- Si expr , basado en el ProximoToken decide llamar a la primera producción, llamaría a expr que ... termina generando un loop infinito, porque **solo Match avanza el token leído.**
- Eliminación
 - Tengo $A \rightarrow A\alpha \mid \beta$
 - En definitiva genera una β seguida de α^* , entonces puedo reemplazar por
 - $A \rightarrow \beta R$
 $R \rightarrow \alpha R \mid \varepsilon$
 - donde R es un nuevo terminal que agrega una producción recursiva a derecha



Factorizar a Izquierda

- Hay casos en que no puedo usar LL(1) porque varias producciones comienzan igual. En forma general si tengo
 - $A \rightarrow \alpha\beta_1 \mid \alpha\beta_2 \mid \dots \mid \alpha\beta_n$
- Convierto con un Nuevo no terminal auxiliar
 - $A \rightarrow \alpha B$
 - $B \rightarrow \beta_1 \mid \beta_2 \mid \dots \mid \beta_n$
- Ejemplo
 - $\langle \text{sentencia if} \rangle \rightarrow \text{if} (\langle \text{condición} \rangle) \langle \text{sentencia} \rangle \text{ else } \langle \text{sentencia} \rangle \mid \text{if} (\langle \text{condición} \rangle) \langle \text{sentencia} \rangle$

if ($\langle \text{condición} \rangle$) $\langle \text{sentencia} \rangle == \alpha$
else $\langle \text{sentencia} \rangle == \beta_1$
 $\varepsilon == \beta_2$

 $\langle \text{sentencia if} \rangle \rightarrow \text{if} (\langle \text{condición} \rangle) \langle \text{sentencia} \rangle \langle \text{opción else} \rangle$
 $\langle \text{opción else} \rangle \rightarrow \text{else } \langle \text{sentencia} \rangle \mid \varepsilon$



Elegir la producción siguiente

- En un parser predictivo debemos poder seleccionar la siguiente producción basados en el siguiente token. Para ello nos valdremos de los conjuntos **primero** y **siguiente**.
- El conjunto primero de un símbolo terminal o no terminal de la gramática es el conjunto de los primeros tokens que pueden ser derivados del símbolo analizado
- Si el símbolo analizado es un terminal, entonces su conjunto primero es ese mismo terminal:
 $\text{Primero}(a) = \{a\}$
- Si A solo tiene la producción $A \rightarrow \alpha$ entonces $\text{Primero}(A) = \text{Primero}(\alpha)$ que es el conjunto de tokens (terminales) que pueden dar inicio a α siguiendo todas las derivaciones posibles de α



Conjunto Primero

- Si $A \rightarrow \alpha \mid \beta$ entonces
 $\text{Primero}(A) = \text{Primero}(\alpha) \cup \text{Primero}(\beta)$. Para poder decidir que producción utilizar con un único símbolo de preanálisis, debe cumplirse que
 $\text{Primero}(\alpha) \cap \text{Primero}(\beta) = \emptyset$
- Más genéricamente, para que una gramática sea LL(1) si tengo $A \rightarrow \alpha \mid \beta \mid \gamma$
 - Quiero con el terminal siguiente saber que producción usar, para eso busco los conjuntos primeros de cada uno de los lados derechos.
 - Entonces $\text{Primero}(\alpha)$, $\text{Primero}(\beta)$ y $\text{Primero}(\gamma)$ deben ser conjuntos disjuntos
- Si A produce o puede derivar en ε entonces ε pertenece a $\text{Primero}(A)$



Armado del conjunto Primero

- Vamos a calcular $\text{Primero}(\alpha)$ suponiendo que $\alpha = X_1X_2\dots X_n$
- Si X_1 es terminal, entonces $\text{Primero}(\alpha) = \{X_1\}$
- Si X_1 es NO terminal, entonces calculamos $\text{Primero}(X_1)$ buscando todas las producciones con X_1 a la izquierda y hacemos la unión de los conjuntos primeros de las partes derechas de esas producciones y agregamos el resultado a $\text{Primero}(\alpha)$
 - Si $\text{Primero}(X_1)$ incluye ε NO lo Agregamos a $\text{Primero}(\alpha)$
- Si X_1 puede generar ε entonces todo pasa a depender de X_2 a quien aplican las mismas reglas que a X_1 y en modo similar si X_2 genera ε iré por X_3 .
- Si $(X_1X_2\dots X_n)$ puede generar ε , entonces ε es parte del conjunto $\text{Primero}(X_1X_2\dots X_n)$
 - Este es el único caso en que agregamos ε a $\text{Primero}(\alpha)$



Conjunto Siguiente

- Si $A \rightarrow \alpha$ y α es o genera ε (X_i genera $\varepsilon \forall_i$) entonces no tengo un token dentro de su conjunto primero que me indique que esta es la producción a seleccionar. Este caso hace necesario fijarnos en el conjunto Siguiente(A)
- El conjunto siguiente de un NO terminal A es el conjunto de tokens que pueden aparecer inmediatamente después A
- Decimos que **a** pertenece al conjunto siguiente de **A** si: $S \Rightarrow^* \alpha A a \beta$
 - Notar que en algún momento puedo haber un no terminal entre A y a , que luego derivó en ε



Armado del conjunto Siguiente

- Buscamos todas las producciones donde en su parte derecha figure A:
 - Si tenemos una producción del tipo $T \rightarrow \alpha A \beta$ entonces incorporamos $\text{Primero}(\beta)$ a $\text{Siguiente}(A)$ pero sin incluir ε en el caso que fuese parte de $\text{Primero}(\beta)$
 - Si tenemos una producción del tipo $T \rightarrow \alpha A$, es decir A es el último símbolo de la producción, entonces agrego $\text{Siguiente}(T)$ a $\text{Siguiente}(A)$
 - Notar que si en nuestro primer caso ε forma parte de $\text{Primero}(\beta)$, cuando β deriva en ε nos queda en la forma del segundo caso, debiendo entonces agregar a $\text{Siguiente}(A)$ tanto $\text{Primero}(\beta)$ como $\text{Siguiente}(T)$.
- Comentario: podría ocurrir que no haya nada a derecha de A. El modo de resolverlo es considerando que FDT forma parte de $\text{Siguiente}(A)$
 - Esto nos lleva a que FDT siempre forma parte del conjunto siguiente del Axioma.



Función Predice

- Combina las dos anteriores para obtener finalmente el conjunto de terminales que nos permitan elegir la producción a utilizar
- $\text{Predice}(A \rightarrow X_1X_2...X_n)$:
 - $\text{Primero}(X_1X_2...X_n)$ si $\varepsilon \notin \text{Primero}(X_1X_2...X_n)$
 - $(\text{Primero}(X_1X_2...X_n) - \{\varepsilon\}) \cup \text{Siguiente}(A)$ en caso contrario
- Por tanto para que la gramática sea LL(1)
Si $A \rightarrow \alpha_1 \mid \alpha_2 \mid \dots \mid \alpha_n$ debe cumplirse:
 - Los conjuntos $\text{Primero}(\alpha_i)$ son disjuntos
 - Esto implica que ε solo puede pertenecer a uno de esos conjuntos
 - Si $\varepsilon \in \text{Primero}(\alpha_k)$ entonces
 $\text{Siguiente}(A) \cap \text{Primero}(\alpha_i) = \emptyset \quad \forall i \neq k$



Parser predictivo no recursivo

- Vamos a usar una pila (como en los AFP) y una tabla que permita seleccionar la producción a utilizar (basada en la función predice)
- La Tabla tiene No terminales como filas y los terminales más FDT como columnas. En cada intersección pondremos la producción a aplicar (o el número la misma).
- Supongamos la gramática de $a^n b^n / n \geq 0$
- $S \rightarrow aSb \mid \varepsilon$



Armado de la tabla

- Regla **1**: $S \rightarrow aSb$
Primero(aSb) = $\{a\}$ por tanto
Predice(aSb) = $\{a\}$
- Regla **2**: $S \rightarrow \varepsilon$
Primero(ε) = $\{\varepsilon\}$ por tanto debo calcular
Siguierte(S) = $\{b, fdt\}$ entonces
Predice(ε) = $\{b, fdt\}$
- Tabla

	a	b	fdt	otros
S	1	2	2	-



Algoritmo

- Comenzamos poniendo en la pila \$ y el Axioma.
- Repetimos
 - Sea **t** el siguiente símbolo (token) en la cadena a analizar y **X** el símbolo en el tope de la pila
 - Si **X** es \$ y **t** es fdt entonces paramos y reportamos éxito
 - Si **X** es un terminal
 - Si **X** = **t** entonces avanzamos al siguiente token a analizar y hacemos pop en la pila para eliminar **X** del tope.
 - Sino: error sintáctico (similar al que daría match(**t**))



Algoritmo (continuación)

- Repetimos (continuación)
 - Si **X** es un NO terminal consultamos la Tabla(X,t).
 - Si indica una regla: $X \rightarrow Y_1 Y_2 \cdots Y_n$ hacemos pop para eliminar X y push en orden inverso de los Y_i de modo que Y_n quede al fondo e Y_1 al tope de la pila.
 - Notar: no avanzamos en la cadena a analizar.
 - Si está indefinido, o sea, no hay regla que aplicar: error sintáctico.



Pseudocódigo

```
push($)  
push(axioma)  
token <- leer token()  
repetir indefinidamente  
    tope <- copia tope_pila()  
    si tope == $ Y token == fdt  
        retornar éxito  
    si tope es terminal  
        si tope == token  
            pop()  
            token <- leer_token()  
        sino  
            error sintáctico  
    sino //tope es NO terminal  
        si tabla(tope, token) definida como  $A \rightarrow B_1B_2 \dots B_n$   
            pop()  
            push( $B_n$ ) ... push( $B_2$ ) push( $B_1$ )  
        sino  
            error sintáctico
```



Ejemplo

	a	b	fdt	otros
S	1: $S \rightarrow aSb$	2: $S \rightarrow \varepsilon$	2: $S \rightarrow \varepsilon$	-

Pila	Cadena remanente	Regla - Acción
S\$	aabb fdt	1 - pop + push
aSb\$	aabb fdt	pop + avanzar
Sb\$	abb fdt	1 - pop + push
aSbb\$	abb fdt	pop + avanzar
Sbb\$	bb fdt	2 - pop (push ε)
bb\$	bb fdt	pop + avanzar
b\$	b fdt	pop + avanzar
\$	fdt	Reconoce



Licencia

*Esta obra, © de Eduardo Zúñiga, está protegida legalmente bajo una licencia Creative Commons, **Atribución-CompartirDerivadasIgual 4.0 Internacional**.
<http://creativecommons.org/licenses/by-sa/4.0/>*

***Se permite: copiar, distribuir y comunicar públicamente la obra; hacer obras derivadas y hacer un uso comercial de la misma.
Siempre que se cite al autor y se herede la licencia.***

