

ASYNTH

FUNCIÓN DEL ANALIZADOR SINTÁCTICO

Construye el árbol de análisis sintáctico (implícito)



Algunas funciones son:

- Recoger info sobre los componentes léxicos en la tabla de símbolos.
- Realizar la verificación de tipo & otras clases de análisis semántico.
- Generar código intermedio
- Manejar errores.

Manejo de errores

- Compilar programas correctos sería lo ideal, pero es alejado de la realidad, los lenguajes no incluyen manejo de errores comunes en su descripción (se deja al compilador).

Algunos errores pueden ser:

- Léxicos, escribir mal ID, operador o palabra clave.
- Sintáctico, paréntesis no equilibrados.
- Semántico, un operador con un operando no compatible.
- Lógico, una llamada recursiva infinita.

Objetivos:

Informar de errores con claridad & exactitud

Recuperarse del error & continuar.

No retrasar programas correctos.

Algunos métodos de análisis sintáctico detectan el error lo antes posible (métodos LL & LR).

Algunas estrategias de recuperación son:

- Modo pánico

Desechar símbolos de entrada hasta encontrar un componente léxico de sincronización (delimitadores, palabras o símbolos de terminación).

- A nivel de frase

Corregir con alguna cadena que permita continuar (., ;)

- Producciones de error

- Corrección global

GRAMATICAS

Las reglas de cada lenguaje indican la sintaxis.

La sintaxis puede especificarse mediante gramatical libres de contexto (notación BNF - Backus-Naur Form).

- Especificación precisa & fácil de entender
- Puede eliminar ambigüedades
- El diseño gramatical apropiado es útil para traducción & manejo de errores

Una gramática permite evolución iterativa de lenguaje.

Integra fácilmente nuevas construcciones

Hay 3 tipos de analizadores sintácticos para gramática

- Los métodos universales
- Métodos ascendentes
- Métodos descendentes

De las hojas a la raíz
de la raíz a las hojas

GRAMATICA LLR & LLD

→ left (construcción del árbol sintáctico).

LL De la raíz a la hoja (descendente)

LR De la hoja a la raíz (ascendente)

↳ right

REPRESENTACION DE GRAMÁTICAS

- Para indicar la sintaxis de un ciclo while:
 - while (condición) {sentencia;} + }
- El token while indica básicamente por donde "comenzar a buscar" la sintaxis de la sentencia. Lo mismo con if, do/while o switch.
- El ej. de análisis será con expresiones aritméticas que podrían representar problemas de asociatividad en las prioridades.

GRAMÁTICA PARA EXPRESIONES

En el ej., E representa expresión, F factor & T término.

$$\begin{array}{l} E \rightarrow E + T \mid T \\ T \rightarrow T * F \mid F \\ F \rightarrow (E) \mid id \end{array}$$

parte de la gramática
del lenguaje.

GRAMÁTICA LR

Existen métodos para eliminar la recursividad de este tipo de gramáticas obteniendo su equivalente.

AMBIGUEDAD

Una gramática permite 2 ó más árboles de análisis sintáctico para una misma expresión.
Suponga:

$$E \rightarrow E + E \mid E * E \mid (E) \mid id$$

Escriba las derivaciones para Vi + vt

necesito id + id + id con

$$E \rightarrow (\textcircled{E}) + E \rightarrow id + (\textcircled{E}) \rightarrow id + id * \textcircled{E} \rightarrow id + id * id$$

GRAMATICA LIBRE DE CONTEXTO

→ una gramática se visualizó como:

instr → if (expr) instr -else instr

→ if, else, (y) son terminales
instr y expr son variables sintácticas
+ Deben describirse con otras producciones

→ Formalmente consiste en:

- Terminales (nombre del token - o simplemente token)
- No terminales o var. sintácticas
- Un símbolo inicial en la primera producc.
- Un conjunto de producciones

$$\begin{aligned} E &\rightarrow E + T \mid T \\ T &\rightarrow T^* F \mid F \\ F &\rightarrow (E) \mid id \end{aligned}$$

TERMINALES Y NO TERMINALES

izq de la producc.
no terminales
derecho
terminal

1. Estos símbolos son terminales

- Las primeras letras abecedario, como abc.
- los símbolos de operadores como +, *, etc.
- Los símbolos de puntuación como parentesis, coma, etc.
- Los dígitos 0, 1, 9.
- Las cadenas en negritas como id o if, cada una de las cuales representa un solo símbolo terminal.

2 Estos símbolos son NO terminales :

- Las primeras letras MAYUSCULAS del abced. ABC
- La letra S que, al parecer es, el símbolo inicial.
- Los nombres en cursiva & minúscula por ej. exp o inst.

3. Las últimas letras mayúsculas del alfabeto, como X, Y, Z , representan *símbolos gramaticales*; es decir, pueden ser no terminales o terminales.
4. Las últimas letras minúsculas del alfabeto, como u, v, \dots, z , representan cadenas de terminales (posiblemente vacías).
5. Las letras griegas minúsculas α, β, γ , por ejemplo, representan cadenas (posiblemente vacías) de símbolos gramaticales. Por ende, una producción genérica puede escribirse como $A \rightarrow \alpha$, en donde A es el encabezado y α el cuerpo.
6. Un conjunto de producciones $A \rightarrow \alpha_1, A \rightarrow \alpha_2, \dots, A \rightarrow \alpha_k$ con un encabezado común A (las llamaremos *producciones A*), puede escribirse como $A \rightarrow \alpha_1 | \alpha_2 | \dots | \alpha_k$. A $\alpha_1, \alpha_2, \dots, \alpha_k$ les llamamos las *alternativas* para A .
7. A menos que se indique lo contrario, el encabezado de la primera producción es el símbolo inicial.

DERIVACIONES

Suponga

$$E \rightarrow E+E \mid E * E \mid -E \mid (E) \mid id$$

Encuentre las derivaciones para: $-id * (id + id)$

$$\begin{aligned} E &\rightarrow E * E \rightarrow -E * E \rightarrow -id * E \rightarrow -id * (E) \\ &\rightarrow -id * (E + E) \rightarrow -id * (id + E) \rightarrow -id * (id + id) \end{aligned}$$

Encuentre para $(id - id) + (id * -id)$

$$\begin{aligned} E &\rightarrow E + E \rightarrow (E) + E \rightarrow (E - E) + E \rightarrow (id - E) + E \\ &\rightarrow (id - id) + E \rightarrow (id - id) + (E) \rightarrow (id - id) + (E * E) \\ &\rightarrow (id - id) + (E * - E) \rightarrow (id - id) + (id * - E) \\ &\rightarrow (id - id) + (id * - id) \end{aligned}$$

VERIFICACION DE UNA EIRAMATICA

- La comprobación se hace en 2 pasos

$$S \rightarrow iEtSS' | a$$

$$S' \rightarrow eS | \epsilon$$

$$E \rightarrow b$$

$$P(S) = \{i, a\}$$

$$P(S') = \{e, \epsilon\}$$

$$P(E) = \{b\}$$

$$S(S) = \{\$, e\}$$

$$S(S') = \{g, eg\}$$

$$S(E) = \{t\}$$

si le sigue minuscula
ahí quedan

reglas

1. \$ si alpha es S (símbolo inicial)

2. P(beta) si $A \rightarrow \alpha B \beta$

3. $S(A)$ si $A \rightarrow \alpha B$ ó $\alpha B \beta$

$$\& P(\beta) = \epsilon$$

A = no terminal de la fila

B = lo que se busca (dentro parentesis)

alpha = antes de B (puede ser epsilon)

beta = lo que esta despues de B (no puede ser epsilon)

sync son de los siguientes

	e	t	a	b	i	\$
S	sync		$S \rightarrow a$		$S \rightarrow iEtSS'$	sync
S'	$S' \rightarrow eS$					$S' \rightarrow \epsilon$
E		sync		$E \rightarrow b$		

símb inicial	cadena	ibta	ibta \$	=	
	$S \$$	cruce	$ibta \$$	$S \rightarrow iEtSS'$	
	$iEtSS' \$$		$ibta \$$	sale i	
	$EtSS' \$$		$bta \$$	$E \rightarrow b$ (Ecambio a b)	
	$bTSS' \$$		$bta \$$	sale b	
	$tSS' \$$		$ta \$$	sale t	
	$SS' \$$		$a \$$	$S \rightarrow a$	
	$gS' \$$		$g \$$	sale a	
	$S' \$$		$\$$	$S' \rightarrow \epsilon$	
	X \$				

(tenemos que negar \$)

$S \rightarrow iEtSS'$

sale i

$E \rightarrow b$ (Ecambio a b)

sale b

sale t

$S \rightarrow a$

sale a

$S' \rightarrow \epsilon$

Valentina Balderas 4722762

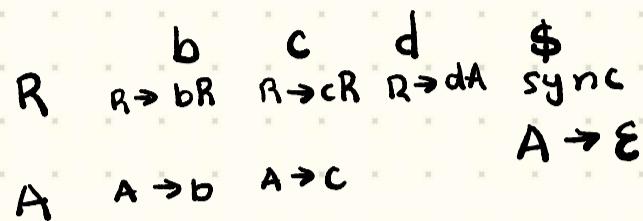
$R \rightarrow bR \mid cR \mid dA$ construir T.A.S.
 $A \rightarrow b \mid c \mid \epsilon$ encontrar bcdc

$$P(R) = \{b, c, d\}$$

$$P(A) = \{b, c, \epsilon\}$$

$$S(R) = \{\$\}$$

$$S(A) = \{\$\}$$



cadena bcdc

$R \$$
 $bR \$$
 $R \$$
 $cR \$$
 $R \$$
 $dA \$$
 $A \$$
 $c \$$
 $\$$

$bcdc\$$
 $bcdc\$$
 $cdc\$$
 $cdc\$$
 $dc\$$
 $dc\$$
 $c\$$
 $c\$$
 $\$$

$R \rightarrow bR$
sale b
 $R \rightarrow cR$
sale c
 $R \rightarrow dA$
sale d
 $A \rightarrow c$
sale c
 $\$$

Valentina Balderas Tones
Willaldo Pérez Oreguera

No terminal	Símbolo de entrada					
	a	b	e	i	t	\$
S	s → a			sync	s → iEtss'	sync
S'				$S' \rightarrow \epsilon$	$S' \rightarrow -eS$	
E			$E \rightarrow B$			sync

si es sync
desaparece 129

S \$
iEtss' \$
Etss' \$
btss' \$
tss' \$
ss' \$
es' \$
es \$
a \$

ibtea \$
iblea \$
btea \$
blea \$
tea \$
ea \$
ea \$
ea \$
a \$

s → iEtss'
sale i
 $E \rightarrow B$
sale b
sale t
botas
s → -eS
sale e
s → g
sale a
\$

OBSERVACIONES

ninguna.

recursividad por izq de rfactor

$r\text{factor} \rightarrow r\text{factor} * | r\text{primario}$

$r\text{factor} \rightarrow r\text{primario}, r\text{factor}'$
 $r\text{factor}' \rightarrow *r\text{factor}' | \epsilon$

$\text{rexPR} \rightarrow \text{rterm } \text{rexPR}$
 $\text{rexPR}' \rightarrow + \text{rterm } \text{rexPR}' | \epsilon$

primero de (N)

$$\begin{aligned} A &\rightarrow dE t \underline{A} N | a \\ N &\rightarrow eS | \epsilon \\ E &\rightarrow b \end{aligned}$$

$$P(N) \rightarrow \{e, \epsilon\}$$

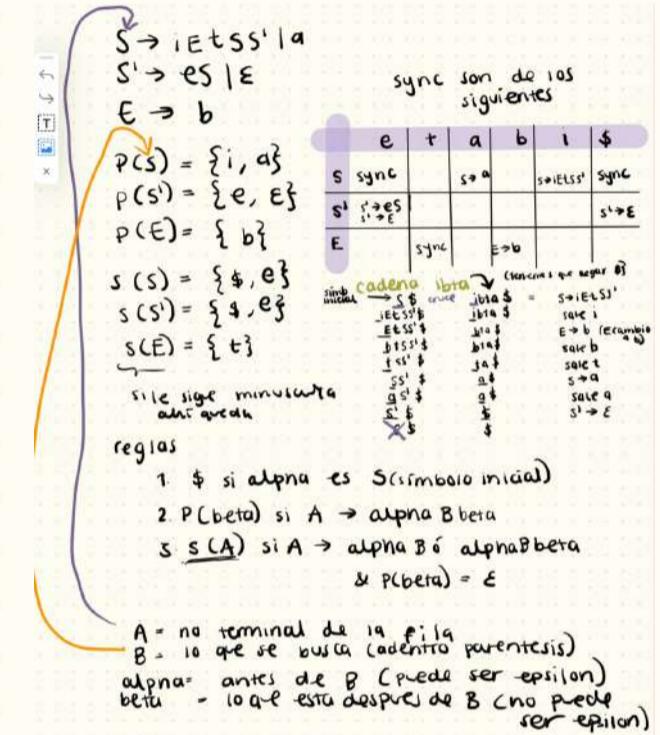
$$\begin{aligned} A &\rightarrow dE t \underline{\alpha} \underline{B} \underline{\beta} | a \\ N &\rightarrow eS | \epsilon \\ E &\rightarrow b \end{aligned}$$

$$\begin{aligned} P(A) &= \{d, a\} \\ P(N) &= \{e, \epsilon\} \\ P(E) &= \{b\} \end{aligned}$$

$$S(A) = \{\$, e\}$$

$\alpha \underline{B} \$$
 αB

$\alpha B \beta$



$r\text{term} \rightarrow r\text{term } \underline{r\text{factor}} | r\text{factor}$

$r\text{term} \rightarrow r\text{factor } r\text{term}'$
 $r\text{term}' \rightarrow r\text{factor } r\text{term}' | \epsilon$