Tema 5

Análisis Sintáctico

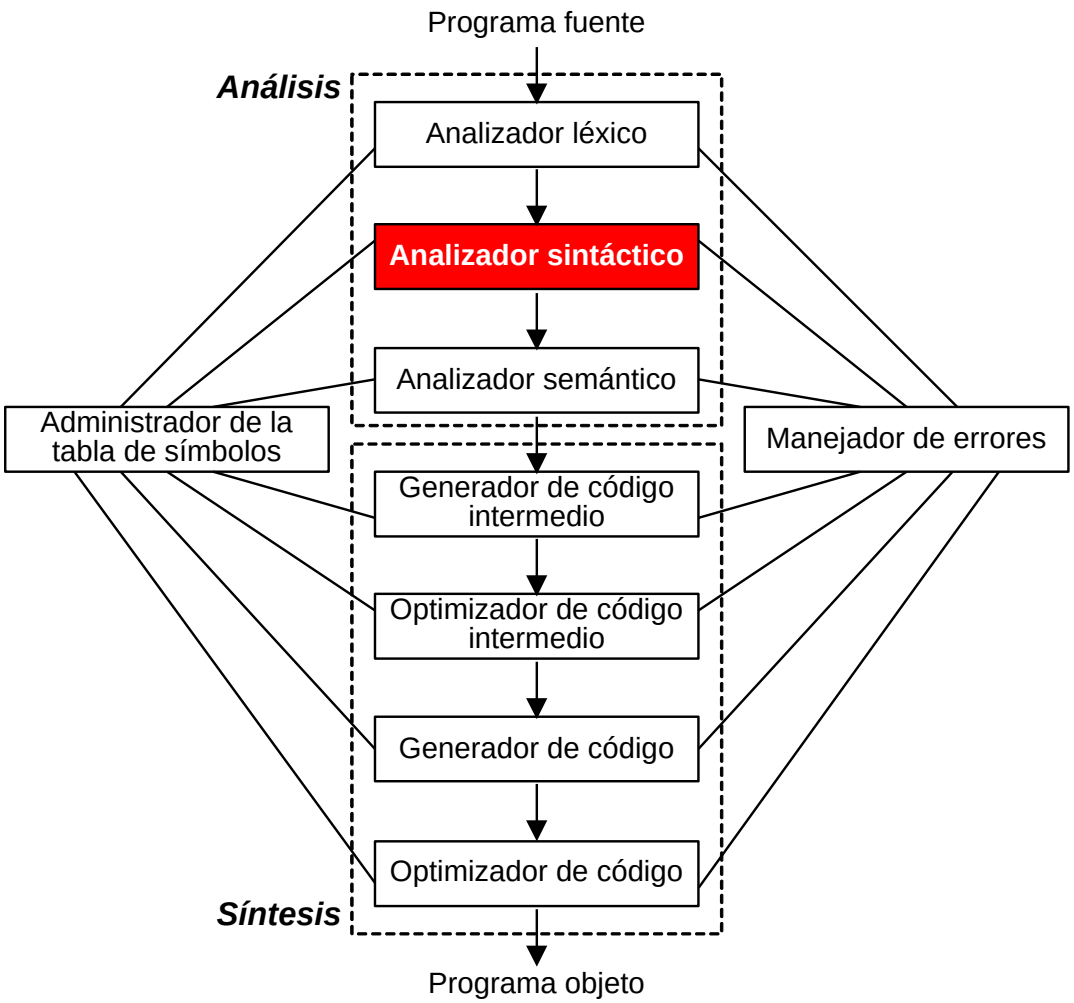
Universidad Europea
LAUREATE INTERNATIONAL UNIVERSITIES

Profesor: Leopoldo Santos 2020

Análisis Sintáctico

- Introducción
- Analizadores Descendentes
- Analizadores Ascendentes
- Generadores de Analizadores Sintácticos
- Recuperación de Errores Sintácticos
- Ambigüedad
- Conjuntos PRIMERO y SIGUIENTE
- Tabla de análisis LL(1)
- Reconocimiento de una sentencia

Introducción



Introducción

- **Sintaxis**.

(Del lat. *syntaxis,* y este del gr. σύνταξις, de συντάσσειν, coordinar).

1. f. *Gram.* Parte de la gramática que enseña a coordinar y unir las palabras para formar las oraciones y expresar conceptos.

2. f. *Inform.* Conjunto de reglas que definen las secuencias correctas de los elementos de un lenguaje de programación.

Diccionario de la RAE

Introducción

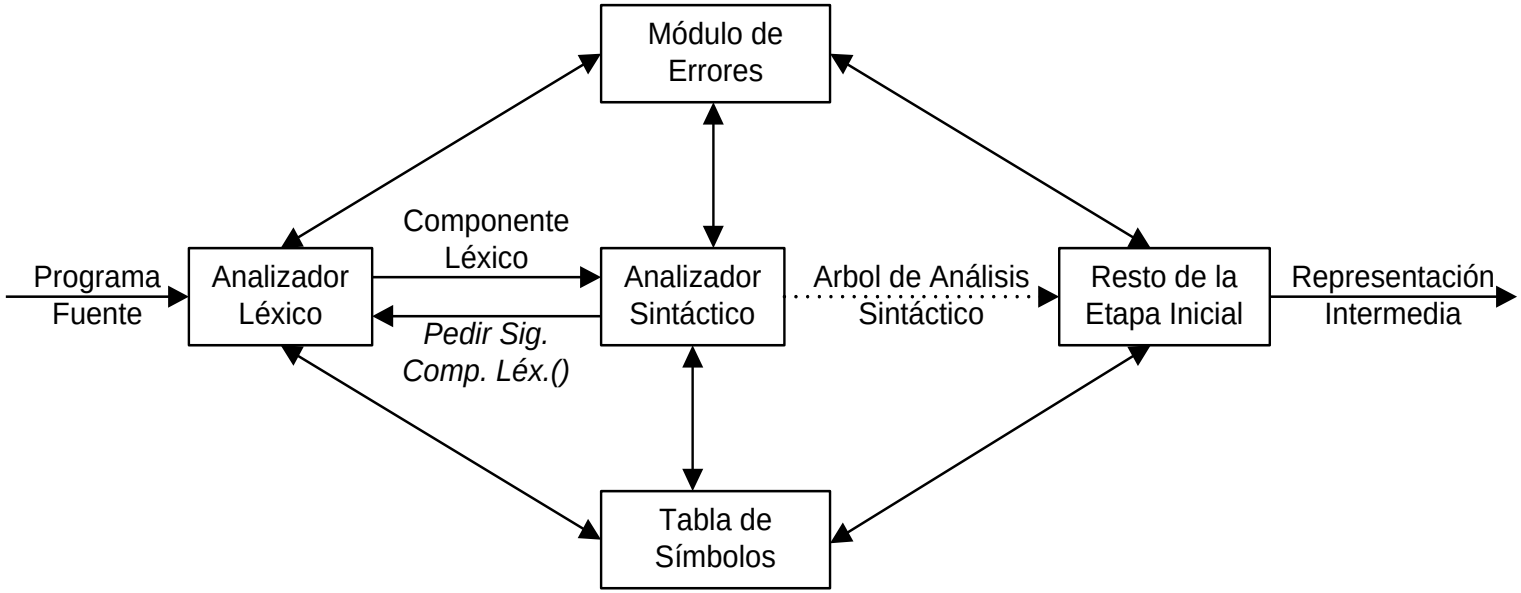
- Todo LP tiene reglas que prescriben la estructura sintáctica de programas bien formados.
  - Ejemplo: Bucle *while* en Java.
- La sintaxis de un lenguaje se puede describir por medio de GIC o notación BNF.

Introducción

- Motivaciones:
  - **Especificación sintáctica precisa y fácil de entender**. Describe de forma natural la estructura jerárquica de muchas construcciones de los lenguajes de programación.
  - A partir de algunas gramáticas **se puede construir automáticamente un analizador sintáctico eficiente** que determine si un programa fuente es sintácticamente correcto.
  - **Ayuda en el proceso de traducción**.
  - **Imparte estructura al lenguaje de programación**, siendo más fácil generar código y detectar errores.
  - **Facilidad de modificación del lenguaje de programación**.

Introducción

- Papel del Analizador sintáctico.
  - Obtiene una cadena de componentes léxicos del analizador léxico y comprueba si dicha cadena puede ser generada por la gramática del lenguaje fuente y “genera el árbol de análisis sintáctico”.
  - Debe informar si se produce algún error, recuperándose de forma adecuada.



Analizador Sintáctico. Tipos

- Tres tipos generales de analizadores sintácticos:
  - **Métodos Universales**: Cocke-Younger-Kasami y Earley
    - Sirven para cualquier gramática
    - Muy ineficientes
  - **Descendentes** (*top-down*)
    - Construyen el árbol de análisis sintáctico desde arriba (raíz, axioma) hasta abajo (hojas, terminales)
      - Analizadores Descendentes Recursivos
      - Analizadores LL(1) con tabla
  - **Ascendentes** (*bottom-up*)
    - Construyen el árbol de análisis sintáctico desde abajo hacia arriba
      - Analizadores de Precedencia de Operador
      - Analizadores LR(1)

Analizador Sintáctico. Tipos

¿Qué significa LL o LR?:

AD(P)

- A: Tipo de análisis de la cadena de entrada.
  - L: De izquierda a derecha.
  - R: De derecha a izquierda.
- D: Tipo de derivación.
  - L: Más a la izquierda (análisis descendente).
  - R: Más a la derecha (análisis ascendente).
- P: Número de símbolos de preanálisis.
  - 0, 1, 2...

– Ejemplos:

- LL(0), LL(1), LR(0), LR(1).

Analizador Sintáctico. Tipos

- **¿Porqué son populares los A. Descendentes?**
- Debido a que se pueden construir manualmente analizadores sintácticos eficiente con facilidad.
- TIPOS:
  - **A. Sintáctico por Descenso Recursivo**: Pueden incluir retroceso, es decir, que puede haber varios exámenes de la entrada → No es eficiente.
  - **A. Sintáctico Predictivo**: Es un analizador que no necesita retroceso. Para poder utilizarlos la gramática no puede ser ambigua. El símbolo de preanálisis determina sin ambigüedad el procedimiento seleccionado para cada No Terminal.
- **Símbolo de preanálisis**: El componente léxico en curso analizado en la entrada

Analizador Sintáctico. Tipos

- **Ventajas de los A. Sintácticos Ascendentes**:
  - Pueden manejar una clase mayor de gramáticas, de modo que las herramientas para generar analizadores sintácticos tienden a utilizar métodos ascendentes.
- Hay dos tipos:
  1. **Análisis por precedencia de operadores**
  2. **Análisis sintáctico LR** (mucho mas general)
- También se los conoce por **análisis sintáctico por desplazamiento y reducción**.
- Se dice **desplazamiento**, porque va a ir desplazando el símbolo de la entrada a la pila de análisis. Se dice **reducción** porque se trata de sustituir una producción de la gramática por la frase que la ha generado.

Introducción

- **Errores sintácticos**.
  - La mayoría de los errores que se producen en programas fuente son de naturaleza sintáctica.
  - El manejador de errores de un analizador sintáctico debe:
    - Informar de la presencia de errores con claridad y exactitud.
    - Recuperarse del error con rapidez, para poder detectar errores posteriores.
    - No producir un retraso significativo en el procesamiento de programas correctos.

Gramáticas Independientes de Contexto: Analizadores Descendentes

- Se puede considerar el AS descendente como un intento de encontrar una derivación por la izquierda para una cadena de entrada.

- a este tipo de análisis se le denomina análisis descendente recursivo.

- Se puede decir también que es un intento de construir el árbol de AS para la entrada comenzando desde la raíz, etiquetada con el no terminal inicial, hacia las hojas.

Gramáticas Independientes de Contexto: Analizadores Descendentes

- Se deben realizar de forma repetida los siguientes pasos:
  - En el nodo n, etiquetado con el no terminal a, seleccionar una de las producciones para a, y construir los hijos de n para los símbolos del lado derecho de la producción.
  - Encontrar el siguiente nodo en el que se debe construir el subárbol.
- Para algunas gramáticas, esta construcción se aplica durante un examen de izquierda a derecha de la cadena de entrada.
- El componente léxico analizado en cada momento se denomina *símbolo de preanálisis*. Inicialmente es el primero por la izquierda.

Gramáticas Independientes de Contexto: Analizadores Descendentes

- Ejemplo: Gramática que genera un subconjunto de los tipos de datos de Pascal.

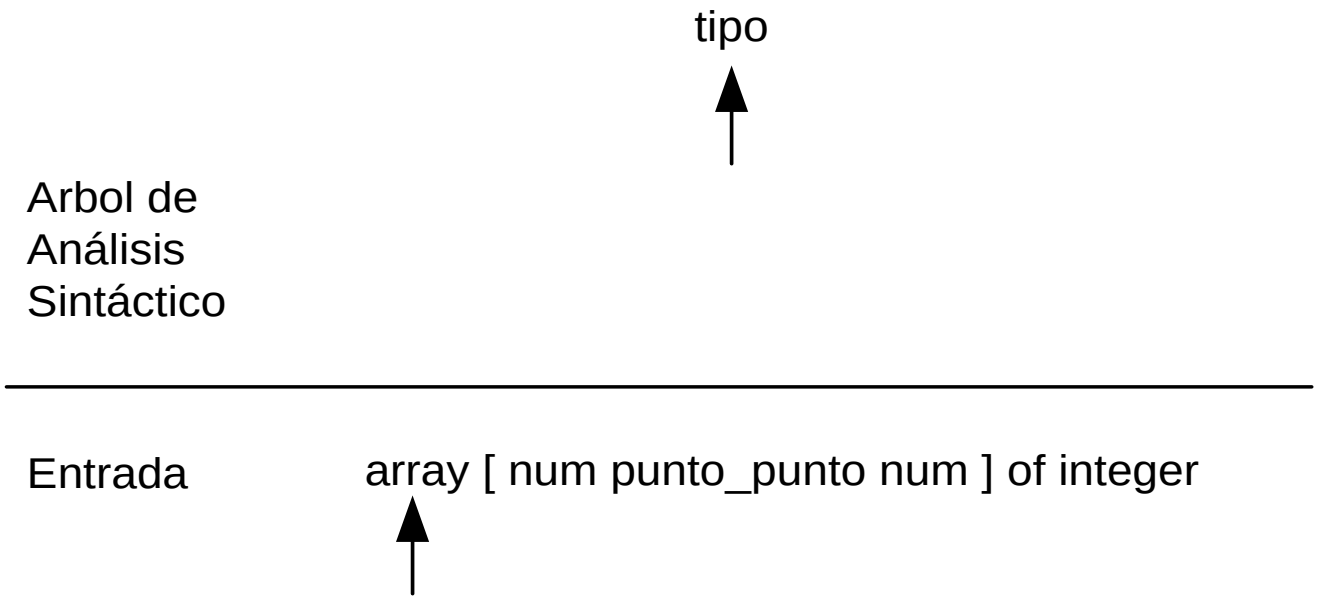
  tipo → tipo_simple |
  ^ IDENTIFICADOR |
  ARRAY [tipo_simple] OF tipo

  tipo_simple → INTEGER |
  CHAR |
  NUM PUNTO_PUNTO NUM

  – Completar el árbol de AS para la cadena de entrada:

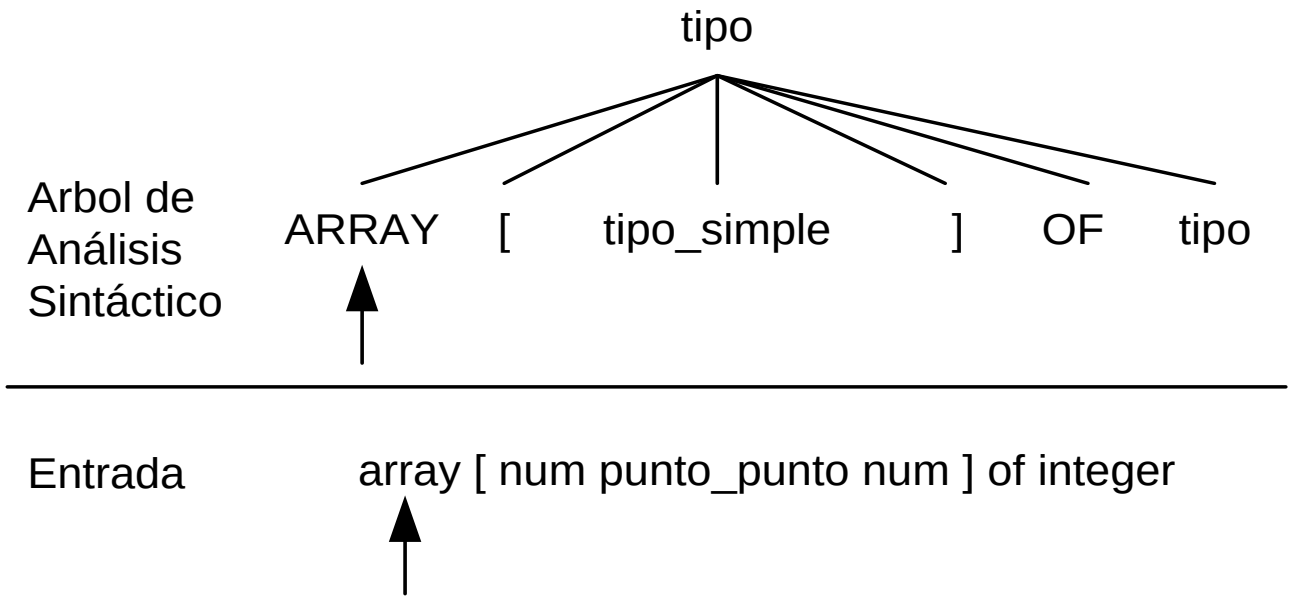
    array [NUM punto_punto NUM] of integer

Gramáticas Independientes de Contexto: Analizadores Descendentes



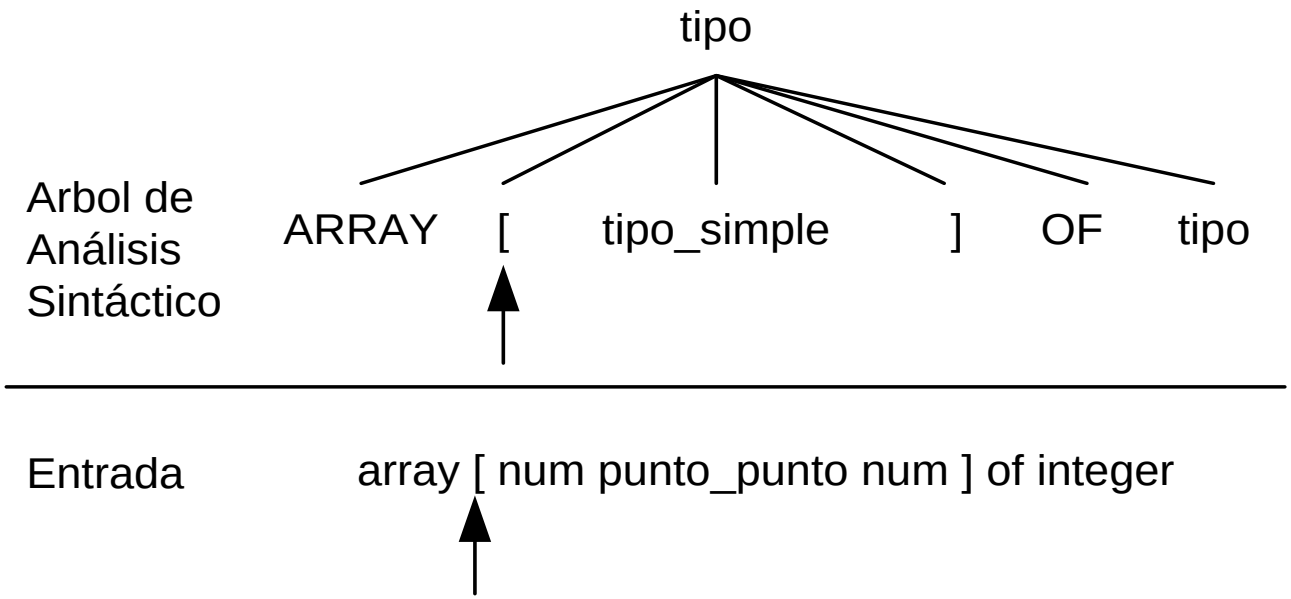
- Para que ocurra una concordancia, el no terminal ***tipo*** debe derivar una cadena que comience con el símbolo de preanálisis ***ARRAY***.
- En la gramática existe una única producción para ***tipo*** cuya parte derecha empieza por ***ARRAY*** ➔ Se elige y se construyen los hijos de la raíz.

Gramáticas Independientes de Contexto: Analizadores Descendentes



- Las flechas indican el símbolo de preanálisis de la entrada y el nodo que se está considerando.
- Cuando el nodo considerado en el árbol AS es un terminal y concuerda con el preanálisis, se avanza en el árbol AS y en la entrada.

Gramáticas Independientes de Contexto: Analizadores Descendentes

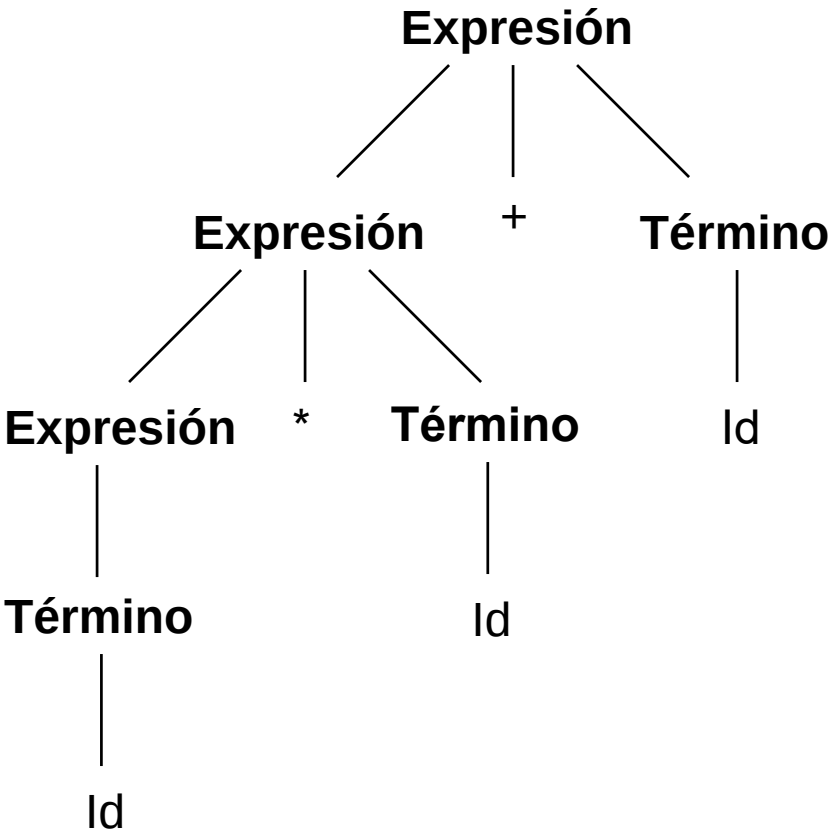


Análisis Sintáctico Predictivo

- No necesita realizar retroceso para analizar bien las sentencias del lenguaje
- Sólo con ver el siguiente carácter de la entrada puede decidir cuál va a ser la siguiente producción a emplear
- Condiciones
  - Diseñar bien la gramática
  - Eliminar la recursividad izquierda
  - Factorizar por la izquierda
- No está asegurado el tener una gramática predictiva
- Las gramáticas son difíciles de leer
- Para las partes de las gramáticas que no son predictivas se pueden utilizar otros analizadores

Análisis Sintáctico Predictivo

- Árbol sintáctico
  - Ejemplo: Id.*.Id.+.Id



Análisis Sintáctico Predictivo: Descendente Recursivo

- Se ejecuta un conjunto de procedimientos recursivos para procesar la entrada
- A cada NO Terminal de una gramática se le asocia un procedimiento
  - Decide la producción que utilizará analizando el símbolo de preanálisis,
    - si está en PRIMERO(α) entonces se usa la producción con lado derecho α
    - si no está en ningún PRIMERO entonces se usa una producción λ
  - Usa una producción imitando al lado derecho
    - no terminal da como resultado una llamada a otro procedimiento
    - terminal (que coincide con el símbolo de preanálisis) produce otra lectura de otro token. Si el token no coincide entonces Error
- La secuencia de procedimientos llamados para procesar la entrada define implícitamente un árbol de análisis sintáctico

Análisis Sintáctico Predictivo: Descendente Recursivo

- Ejemplo:

  S → if **B** then **S** | write **B** | i := **B**

  B → i = i | i <> i | true | false

```
Procedure S()
{
if (car== if)
   { scan();
      B();
      if (car== then) {scan();} else error();
      S();
    }
  else if (car==write)
  {scan();
    B();
   }
 else if (car== i)
  { scan();
     if (car == asig) {scan();} else error();
     B();
   }
  else error()
}
```

```
Procedure B()
{
   if (car= i)
     { scan();
        if (car in [igual, noigual]) scan(); else error();
        if (car == i) then scan else error;
     }
  elseif (car in [true, false])
        scan();
  else
        error();
}
```

Análisis Sintáctico Predictivo. DEFINICIONES: PRIMERO

- Si α es una cadena de símbolos gramaticales, **PRIMERO(α)** es el conjunto de terminales que inician las cadenas derivadas de α.
  - PRIMERO(α)={x | ($\alpha \rightarrow_* x.\beta$), ($x \in \Sigma_T \cup \{\lambda\}$), ($\alpha \in \Sigma^*$)}
- Conjunto PRIMERO(X) para todos los símbolos gramaticales X
  1. Repetir hasta que no se puedan añadir más terminales o λ a ningún conjunto PRIMERO
  2. Si $X \in \Sigma_T \Rightarrow$ PRIMERO(X) es { X }
  3. Si $X \rightarrow \lambda \Rightarrow$ añadir λ a PRIMERO(X)
  4. Si $X \in \Sigma_N$ y $X \rightarrow Y_1Y_2...Y_KY \Rightarrow a \in$ PRIMERO(X) si $a \in$ PRIMERO(Y_i) y $\lambda \in$ PRIMERO(Y_1), PRIMERO(Y_2),..., PRIMERO(Y_{i-1})
     - Si Y_1 deriva a $\lambda \Rightarrow$ se añade PRIMERO(Y_2)
     - Si Y_1 no deriva a $\lambda \Rightarrow$ no se añade más a PRIMERO(X)

Análisis Sintáctico Predictivo, DEFINICIONES: PRIMERO

- PRIMERO(α), Ejemplo:

  E ::= **T**.**E'**

  E' ::= +.**T**.**E'** | λ

  T ::= **F**.**T'**

  T' ::= *.**F**.**T'** | λ

  F ::= (.**E**.) | Id

  - PRIMERO(**E**) = { (, Id }
  - PRIMERO(**T**) = { (, Id }
  - PRIMERO(**F**) = { (, Id }
  - PRIMERO(**E'**) = { +, λ }

  PRIMERO(**T**.*.Id) = { (, Id }

  PRIMERO(Id.+.Id) = { Id }

  PRIMERO(Id) = { Id }

  PRIMERO(**T'**) = { *, λ }

Análisis Sintáctico Predictivo, DEFINICIONES: SIGUIENTE

- Conjunto SIGUIENTE(A)
  - SIGUIENTE(A)={x|($S \rightarrow_* \alpha \cdot A \cdot \beta$), ($A \in \Sigma_N$), ($\alpha \in \Sigma^*$), ($\beta \in \Sigma^+$), ($x \in$ PRIMERO(β)-{λ})}
  - Conjunto de terminales que pueden aparecer inmediatamente a la derecha de A en alguna forma sentencial, si A es el último símbolo entonces se incluye el separador $
- Algoritmo
  1. SIGUIENTE(S)={$}
  2. La regla $A \rightarrow \alpha B \beta \Rightarrow$
     SIGUIENTE(B) = (PRIMERO(β)-{λ}) $\cup$ SIGUIENTE(A)
  3. La regla $A \rightarrow \alpha B \beta$ | $\beta = \lambda$, $\beta \rightarrow_* \lambda$ ($\lambda \in$ PRIMERO(β)) $\Rightarrow$
     SIGUIENTE(B) = SIGUIENTE(A) $\cup$ SIGUIENTE(B)
  4. Repetir hasta que no cambie ningún conjunto SIGUIENTE

Análisis Sintáctico Predictivo, DEFINICIONES: SIGUIENTE

- SIGUIENTE(A), Ejemplo:

  E ::= **T.E'**

  E' ::= +.**T.E'** | λ

  T ::= **F.T'**

  T' ::= *.**F.T'** | λ

  F ::= (.**E**.) | Id

Σ_N	SIGUIENTE
E	$,)
E'	$,)
F	$, *,), +
T	$, +,)
T'	$, +,)

Análisis Sintáctico Predictivo: Condiciones

- Pregunta:
  - ¿Que debe cumplir una gramática para que pueda ser reconocida sin retroceso, con solo mirar el siguiente elemento de la entrada, de forma descendente?
- Respuesta:
  - Si A::= α | $\beta \Rightarrow$
    - PRIMERO(α) $\cap$ PRIMERO(β) = $\emptyset$. para ningún terminal *a* tanto α y β derivan a la vez cadenas que comiencen con *a*
    - No puede ocurrir que $\alpha \rightarrow_* \lambda$ y $\beta \rightarrow_* \lambda$
    - Si $\beta \rightarrow_* \lambda$, entonces α no deriva ninguna cadena que comience con un terminal en SIGUIENTE(A)
  - Condición LL(1)

Análisis Sintáctico Predictivo: Condiciones

- Condición LL(1)
  - La gramática no puede ser ambigua
  - No puede haber conflictos PRIMERO/PRIMERO
    - $\forall N \in \Sigma_N$, el conjunto PRIMERO de todas sus alternativas debe ser disjunto
  - No puede haber múltiples alternativas nulas
    - $\forall N \in \Sigma_N$, solo pueden tener una producción $N \rightarrow \lambda$
  - No puede haber conflictos PRIMERO/SIGUIENTE
    - $\forall N \in \Sigma_N$, con una alternativa nula, SIGUIENTE(N) debe ser disjunto de los conjuntos PRIMERO de todas sus alternativas
  - No puede haber entradas con definiciones múltiples en la tabla de análisis

Análisis Sintáctico Predictivo: Tabla de Análisis Sintáctico

- Funcionamiento
  - Sea $A \rightarrow \alpha$ con $a \in \Sigma_T$ | $a \in$ PRIMERO(α). El analizador sintáctico expandirá A por α cuando el símbolo actual de la entrada sea a
- Algoritmo

ForAll (A::= α) $\in$ P do

ForAll a $\in$ PRIMERO(α) do

tabla(A,a)= α

Si $\lambda \in$ PRIMERO(α)

Entonces ForAll b $\in$ SIGUIENTE(A) do

tabla(A,b)= λ

ForAll $A \in \Sigma_N$ y $c \in \Sigma_T$ do

If tabla(A,c)= $\varnothing$

Then tabla(A,c)= error

Análisis Sintáctico Predictivo: Tabla de Análisis Sintáctico

- Ejemplo

E ::= **T.E'**

E' ::= +.**T.E'** | λ

T ::= **F.T'**

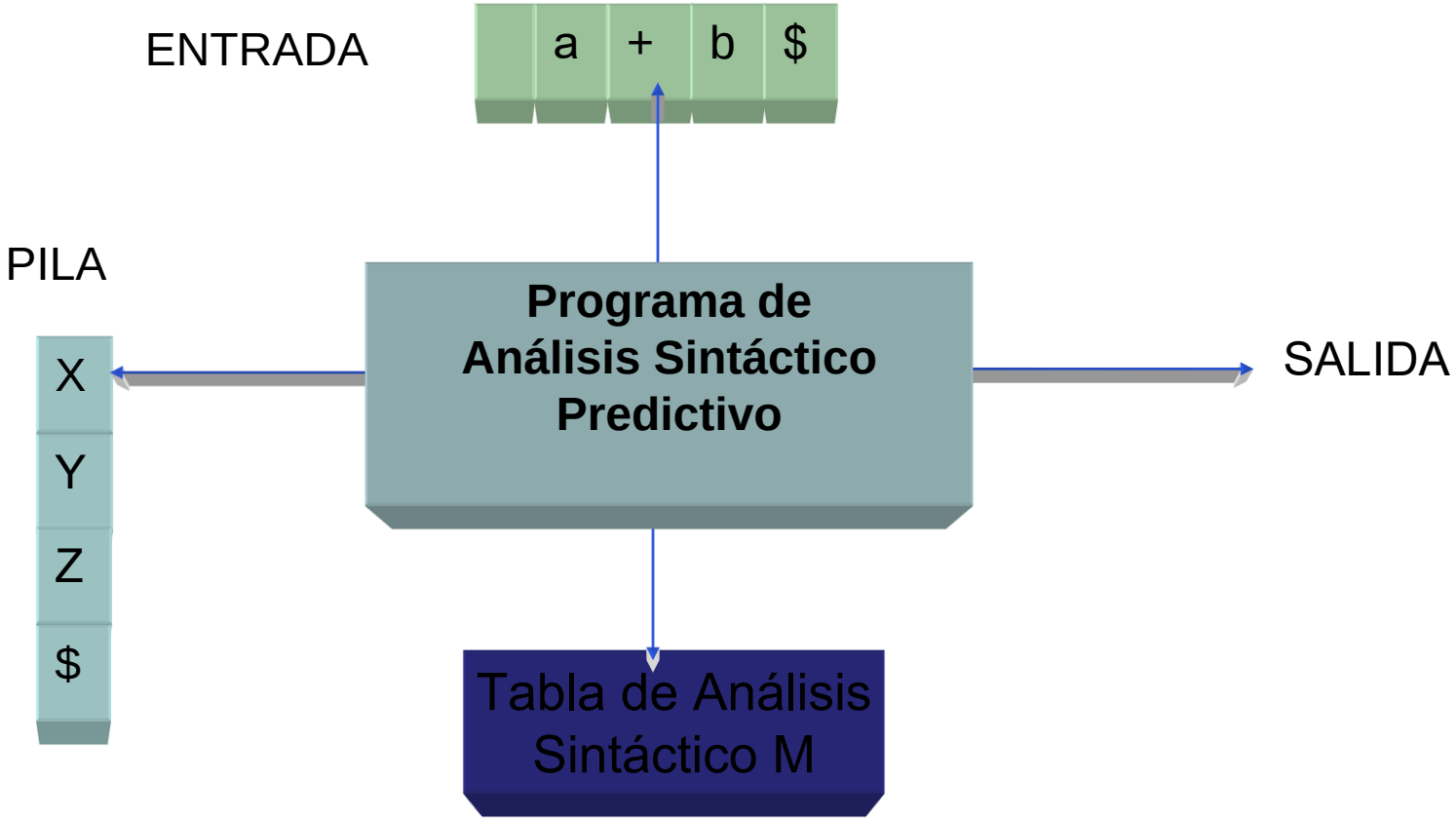
T' ::= *.**F.T'** | λ

F ::= (.**E.**) | Id

	Id	**+**	*****	**(**	**)**	**$**
E	T.E'			T.E'		
E'		+.T.E'			λ	λ
T	F.T'			F.T'		
T'		λ	*.F.T'		λ	λ
F	Id			(.E.)		

Análisis Sintáctico Predictivo No Recursivo; LL(1)

- Modelo de analizador sintáctico predictivo no recursivo



Análisis Sintáctico Predictivo No Recursivo; LL(1)

- Los símbolos de la entrada actual “a” y cima de la pila “X” determinan la acción del analizador
- Hay tres posibilidades:
  - X=a=$, el analizador se detiene y anuncia el éxito del análisis
  - X=a$\neq$$, el analizador saca X de la pila y mueve el apuntador de la entrada al siguiente símbolo de entrada
  - $X \in \Sigma_N$, el programa consulta la entrada M[X,a]
    - Si M[X,a]=UVW, se sustituye la X de la pila por WVU (U queda como cima de la pila)
    - Si M[X,a]= error, se llama a la rutina de recuperación de error

Análisis Sintáctico LL(1)

- Algoritmo:

```
pila =$;
meter$ al final de la entrada;
a:= GetToken;
Push S;
Repeat
    If X ∈Σ_T or            X=$ then
        If X=a then
            Pop;
            a:= GetToken;
        Else
            error;
    Else
        If M[X,a]=X→Y_1Y_2..Y_k then
            Pop;
            Push Y_k,Y_k-1,...,Y_1
            Emitir la producción X
        else
            error();
until X=$
```

```
If X=$ and a=$ then
        Aceptar;
else
        error();
```

Análisis Sintáctico LL(1)

- Ejemplo:

Pila	Entrada	Producción
$ · E	Id · * · Id · + · Id · $	E::= T · E’
$ · E’ · T	Id · * · Id · + · Id · $	T::= F · T’
$ · E’ · T’ · F	Id · * · Id · + · Id · $	F::= Id
$ · E’ · T’ · Id	Id · * · Id · + · Id · $	
$ · E’ · T’	* · Id · + · Id · $	T’::= * · F · T’
$ · E’ · T’ · F · *	* · Id · + · Id · $	
$ · E’ · T’ · F	Id · + · Id · $	F::= Id
$ · E’ · T’ · Id	Id · + · Id · $	
$ · E’ · T’	+ · Id · $	T’::= λ
$ · E’	+ · Id · $	E’::= + · T · E’
$ · E’ · T · +	+ · Id · $	
$ · E’ · T	Id · $	T::= F · T’
$ · E’ · T’ · F	Id · $	F::= Id
$ · E’ · T’ · Id	Id · $	
$ · E’ · T’	$	T’::= λ
$ · E’	$	E’::= λ
$	$	