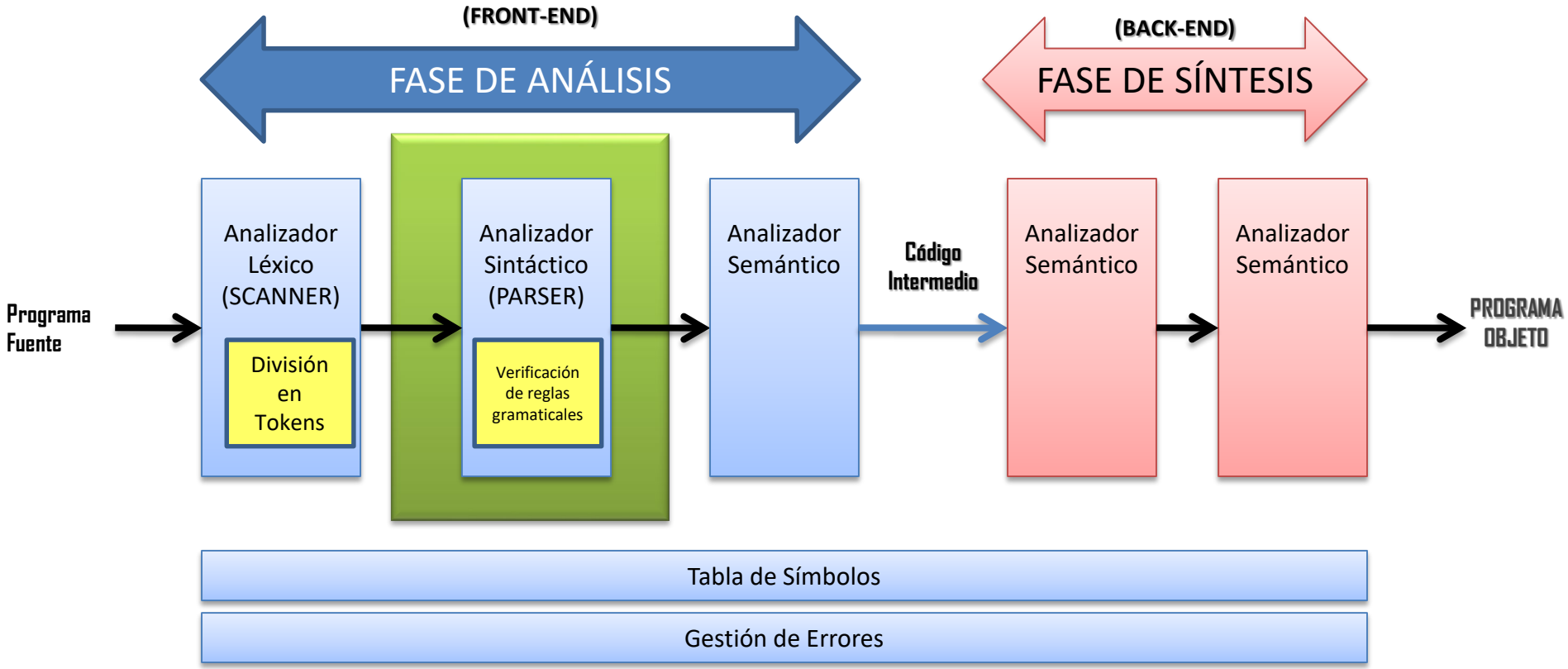
Sesión 5
Analizador Sintáctico
ASD Recursivo y Predictivo

Antonio Moratilla Ocaña

Resumen del tema

- Objetivo:
  - Estudiar cómo se produce el análisis descendente de una expresión para una gramática, tanto recursivo como predictivo.

Posición en el diagrama



Retomando el hilo...

- Ya sabemos
  - Que las gramáticas nos permiten definir un lenguaje
- Lo que queremos saber
  - ¿Cómo sabemos si una entrada corresponde con un lenguaje definido por una gramática?

Métodos de análisis sintáctico

- El análisis sintáctico comprueba que la entrada cumple las condiciones impuestas por la gramática.
- Dos tipos:
  - **Análisis Sintáctico Descendente (ASD).** Parten del símbolo inicial de la gramática (axioma) y van expandiendo producciones hasta llegar a la cadena de entrada.
  - **Análisis Sintáctico Ascendente.** Parten de los terminales de la entrada y mediante reducciones llegan hasta el símbolo inicial.

- Notas:
  1. Los métodos descendentes se pueden implementar más fácilmente sin necesidad de utilizar generadores automáticos.
  2. Los métodos ascendentes pueden manejar una mayor gama de gramáticas por lo que los generadores automáticos suelen utilizarlos.
  3. Para cualquier gramática independiente de contexto hay un analizador sintáctico "general" que toma como máximo un tiempo de $O(n^3)$ para realizar el análisis de una cadena de n componentes léxicos. Se puede conseguir un análisis lineal O(n)para la mayoría de lenguajes de programación.

Análisis Sintáctico Descendente - ASD

Métodos que parten del axioma y, mediante derivaciones por la izquierda, tratan de encontrar la entrada.

- Existen dos formas de implementarlos:
  - **Análisis descendente recursivo**
    - Es la manera más sencilla, implementándose con una función recursiva aprovechando la recursividad de la gramática.
  - **Análisis descendente predictivo**
    - Para aumentar la eficiencia, evitando los retrocesos, se predicen cada momento cuál de las reglas sintácticas hay que aplicar para continuar el análisis
    - En la práctica apenas se emplea el recursivo (o con retroceso) debido a diversos inconvenientes.

ASD - Recursivo

A. Mediante un método de ***backtracking*** se van probando todas las opciones de expansión para cada no-terminal de la gramática hasta encontrar la correcta.

B. Cada retroceso en el árbol sintáctico tiene asociado un retroceso en la entrada: Se deben eliminar todos los terminales y no terminales correspondientes a la producción que se “elimina” del árbol.

C. Si el terminal obtenido como consecuencia de probar con una opción de las varias de una producción no coincide con el componente léxico leído en la entrada, hay que retroceder.

Algoritmo del ASD - Recursivo

1) Se colocan las reglas en orden, de forma que si la parte derecha de una producción es prefijo de otra, esta última se sitúa detrás.
2) Se crea el nodo inicial con el axioma y se considera nodo activo.
3) Para cada nodo activo A:
   a) Si A es un no-terminal, se aplica la primera producción asociada a A.
      1) El nodo activo pasa a ser el hijo izquierdo.
      2) Cuando se terminan de tratar todos los descendientes, el siguiente nodo activo es el siguiente hijo por la izquierda.
   b) Si A es un terminal :
      1) Si coincide el símbolo de la entrada, se avanza el puntero de entrada y el nodo activo pasa a ser el siguiente "hermano" de A.
      2) Si no, se retrocede en el árbol hasta el anterior no-terminal (y en la entrada si se ha reconocido algún componente léxico mediante la producción que se elimina) y se prueba la siguiente producción.
         – Si no hay más producciones para probar, se retrocede hasta el anterior no-terminal y se prueba con la siguiente opción de éste.
4) Si se acaban todas las opciones del nodo inicial, error sintáctico. Si por el contrario se encuentra un árbol para la cadena de entrada, éxito.

Algoritmo del ASD recursivo

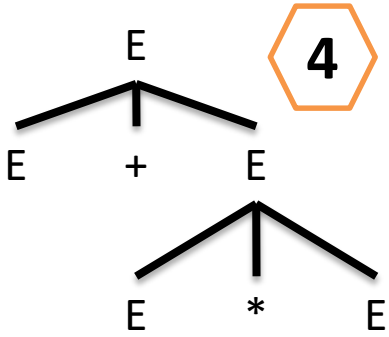
Ejemplo :

Comprobar si la cadena **ccd** pertenece al lenguaje de la gramática:

G(S → c X d ; X → c k | c)

Algoritmo del ASD recursivo

Elementos auxiliares:



2

PILA

Entrada 1

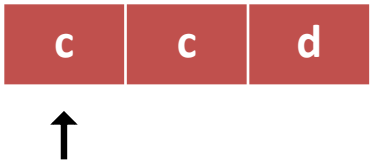
3

ccd para G(S → c X d ; X → c k | c)

Entrada	Pila	Regla a aplicar	Se obtiene en la pila	Se empareja con la entrada	Queda en la pila	Queda en la entrada

Algoritmo del ASD recursivo

Entrada



PILA
S

ccd para G(S → c X d ; X → c k | c)

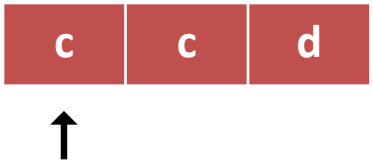
Entrada	Pila	Regla a aplicar	Se obtiene en la pila	Se empareja con la entrada	Queda en la pila	Queda en la entrada
c c d	S					

Algoritmo del ASD recursivo

Regla

- S → c X d

Entrada



PILA
d
X
c

ccd para G(S → c X d ; X → c k | c)

Entrada	Pila	Regla a aplicar	Se obtiene en la pila	Se empareja con la entrada	Queda en la pila	Queda en la entrada
c c d	S	S → c X d	c X d			

Algoritmo del ASD recursivo

Regla

- S → c X d

Entrada

c	c	d

↑

PILA
d
X
c

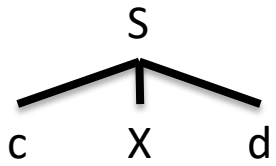
ccd para G(S → c X d ; X → c k | c)

Entrada	Pila	Regla a aplicar	Se obtiene en la pila	Se empareja con la entrada	Queda en la pila	Queda en la entrada
c c d	S	S → c X d	c X d	c	X d	c d

Algoritmo del ASD recursivo

Regla

- S → c X d



Entrada

c	c	d

↑

PILA
d
X

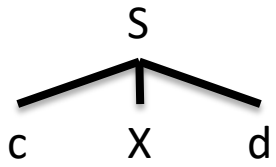
ccd para G(S → c X d ; X → c k | c)

Entrada	Pila	Regla a aplicar	Se obtiene en la pila	Se empareja con la entrada	Queda en la pila	Queda en la entrada
c c d	S	S → c X d	c X d	c	X d	c d

Algoritmo del ASD recursivo

Regla

- S → c X d



Entrada

c	c	d

↑

PILA
d
X

ccd para G(S → c X d ; X → c k | c)

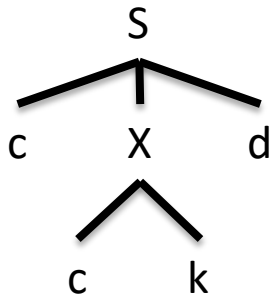
Entrada	Pila	Regla a aplicar	Se obtiene en la pila	Se empareja con la entrada	Queda en la pila	Queda en la entrada
c c d	S	S → c X d	c X d	c	X d	c d
c d	X d					

Algoritmo del ASD recursivo

Regla
X → c k

Entrada

c	c	d
↑		



PILA
d
k
c

ccd para G(S → c X d ; X → c k | c)

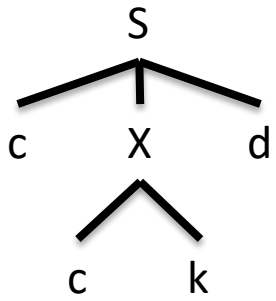
Entrada	Pila	Regla a aplicar	Se obtiene en la pila	Se empareja con la entrada	Queda en la pila	Queda en la entrada
c c d	S	S → c X d	c X d	c	X d	c d
c d	X d	X → c k	c k d			

Algoritmo del ASD recursivo

Regla
X → c k

Entrada

c	c	d
↑		



PILA
d
k
c

ccd para G(S → c X d ; X → c k | c)

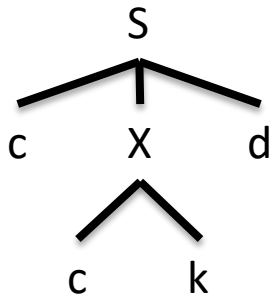
Entrada	Pila	Regla a aplicar	Se obtiene en la pila	Se empareja con la entrada	Queda en la pila	Queda en la entrada
c c d	S	S → c X d	c X d	c	X d	c d
c d	X d	X → c k	c k d	c	k d	d

Algoritmo del ASD recursivo

Regla
X → c k

Entrada

c	c	d
		↑



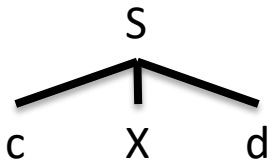
PILA
d
k

ccd para G(S → c X d ; X → c k | c)

Entrada	Pila	Regla a aplicar	Se obtiene en la pila	Se empareja con la entrada	Queda en la pila	Queda en la entrada
c c d	S	S → c X d	c X d	c	X d	c d
c d	X d	X → c k	c k d	c	k d	d
d	d k	d y k no se emparejan, hay que deshacer el último paso (2) y probar la siguiente regla				

Algoritmo del ASD recursivo

Regla
X → c k



Entrada

c	c	d
↑		

PILA
d
X

ccd para G(S → c X d ; X → c k | c)

Entrada	Pila	Regla a aplicar	Se obtiene en la pila	Se empareja con la entrada	Queda en la pila	Queda en la entrada
c c d	S	S → c X d	c X d	c	X d	c d
c d	X d	X → c k	c k d	c	k d	d
d	d k	d y k no se emparejan, hay que deshacer el último paso (2) y probar la siguiente regla				
c d	X d	X → c				

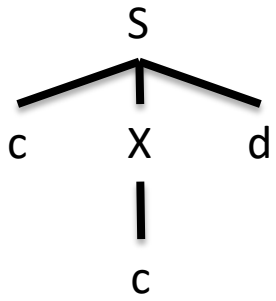
Algoritmo del ASD recursivo

Regla
X → c k

Entrada

c	c	d

↑



PILA
d
c

ccd para G(S → c X d ; X → c k | c)

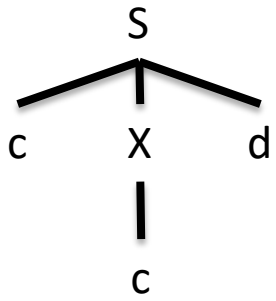
Entrada	Pila	Regla a aplicar	Se obtiene en la pila	Se empareja con la entrada	Queda en la pila	Queda en la entrada
c c d	S	S → c X d	c X d	c	X d	c d
c d	X d	X → c k	c k d	c	k d	d
d	d k	d y k no se emparejan, hay que deshacer el último paso (2) y probar la siguiente regla				
c d	X d	X → c	c d	c	d	d

Algoritmo del ASD recursivo

Regla
X → c k

Entrada

c	c	d
	↑	



PILA
d

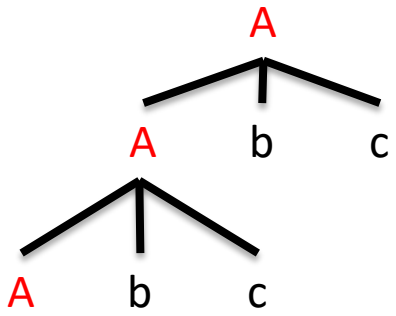
ccd para G(S → c X d ; X → c k | c)

Entrada	Pila	Regla a aplicar	Se obtiene en la pila	Se empareja con la entrada	Queda en la pila	Queda en la entrada
c c d	S	S → c X d	c X d	c	X d	c d
c d	X d	X → c k	c k d	c	k d	d
d	d k	d y k no se emparejan, hay que deshacer el último paso (2) y probar la siguiente regla				
c d	X d	X → c	c d	c	d	d
d	d			d		

ASD Recursivo - Problemas

- No puede tratar gramáticas con recursividad a izquierdas.

A → Abc



- Acaba la ejecución cuando se encuentra el primer error
  - Difícil proporcionar mensajes más elaborados que “correcto” o “incorrecto”, como por ejemplo especificar dónde se ha encontrado el error.
- Aunque la programación es simple, utiliza muchos recursos
  - Como consecuencia del retroceso necesita almacenar los componentes léxicos ya reconocidos por si es necesario volverlos a tratar.
- Cuando un analizador sintáctico se utiliza para comprobar la semántica y generar código, cada vez que se expande una regla, se ejecuta una acción semántica. Al retroceder esa regla o producción se deben deshacer las acciones semánticas, lo que no es fácil ni siempre posible.

ASD Recursivo

Eliminación de la recursividad por la izquierda

Recursión inmediata

Si la gramática recursiva tiene la forma siguiente:

$$A \rightarrow A\alpha_1 | A\alpha_2 | \ldots | A\alpha_n | \beta_1 | \beta_2 | \ldots | \beta_m$$

A es el no-terminal recursivo.

$\boldsymbol{\alpha_i}$, partes derechas de las reglas recursivas del no terminal A.

$\boldsymbol{\beta_i}$, partes derechas de las reglas no recursivas del no terminal A.

La gramática no recursiva equivalente será:

$$A \rightarrow \beta_1 A' | \beta_2 A' | \ldots | \beta_m A'$$

$$A' \rightarrow \alpha_1 A' | \alpha_2 A' | \ldots | \alpha_m A' | \varepsilon$$

ASD Recursivo

Eliminación de la recursividad por la izquierda

Recursión indirecta

Para eliminarla recursividad indirecta se debe encontrar el elemento conflictivo y sustituirlo por su definición.

Ejemplo

S → A a | b

A → A c | **S** d | Ɛ (A → S d es recursiva por S → Aa)

Sustituimos

S → A a | b

A → A c | A a d | b d | Ɛ

Y eliminamos la recursión inmediata como antes:

S → A a | b

S → b d A' | A'

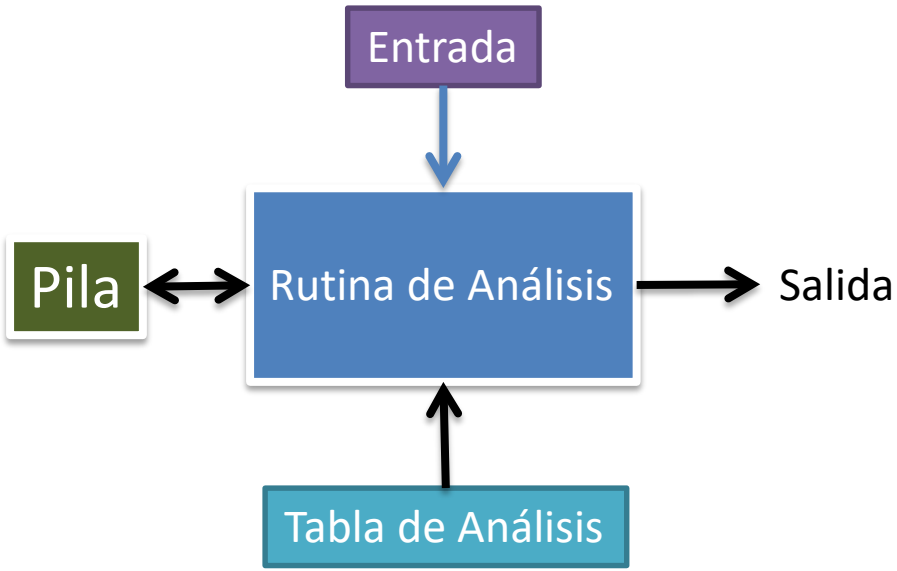
A' → c A' | a d A' | Ɛ

ASD Predictivo

- Intentan predecir la siguiente construcción a aplicar leyendo uno o más componentes léxicos por adelantado
  - "Saben" con exactitud qué regla deben expandir para llegar a la entrada
- Este tipo de analizadores se denomina LL(k)
  - Leen la entrada de izquierda a derecha (Left to rigth)
  - Aplican derivaciones por la izquierda para cada entrada (Left)
  - Utilizan k componentes léxicos de la entrada para predecir la dirección del análisis.
- Están formados por:
  - Un buffer para la entrada.
  - Una pila de análisis.
  - Una tabla de análisis sintáctico.
  - Una rutina de control.

ASD Predictivo

- En función de la entrada, de la tabla de análisis y de la pila decide la acción a realizar.



Posibles Acciones:

1. **Aceptar la cadena.**
2. **Aplicar Producción.**
3. **Pasar al siguiente símbolo de entrada.**
4. **Notificar Error.**

ASD Predictivo Tabla de análisis sintáctico

- Se trata de una matriz **M [Vn, Vt]** donde se representan las producciones a expandir en función del estado actual del análisis y del símbolo de la entrada.
- Las entradas en blanco indican errores

	a	b	c	d	e	$
S	S → BA	error	error	S → BA	error	error
A	error	A → bSC	error	error	A → Ɛ	A → Ɛ
B	B → DC	error	error	B → DC	error	error
C	error	C→ Ɛ	C→ cDC	error	C→ Ɛ	C→ Ɛ
D	D→ a	error	error	D→ dSe	error	error

ASD Predictivo - Ejemplo

GRAMATICA		Cadena entrada	Símbolo en la entrada	Regla a aplicar *	Pila	Derivaciones aplicadas
A→B **A→aBc** **A→xC** **B→bA** **C→c**	1	babxcc	b	A→B	B	B
	2	babxcc	b	B→bA	bA	bA
	3	abxcc	a	A→aBc	aBc	baBc
	4	bxcc	b	B→bA	bAc	babAc
	5	xcc	x	A→xC	xCc	babxCc
	6	cc	c	C→c	cc	babxcc
	7	c	c		c	babxcc
	8					babxcc

* La regla a aplicar vendría dada por la tabla de análisis sintáctico.

Ejercicios

- Dada la gramática G(A → a B b ; B → c d | c)
- Realice en proceso de análisis sintáctico descendente recursivo para reconocer las expresiones
  - acdb
  - abcd
  - acb

Fuentes

- Para la elaboración de estas transparencias se han utilizado:
  - Transparencias de cursos previos (elaboradas por los profesores Dr. D. Salvador Sánchez, Dr. D. José Luis Cuadrado).
  - Libros de referencia (en especial capítulos 2 y 4 de Aho).