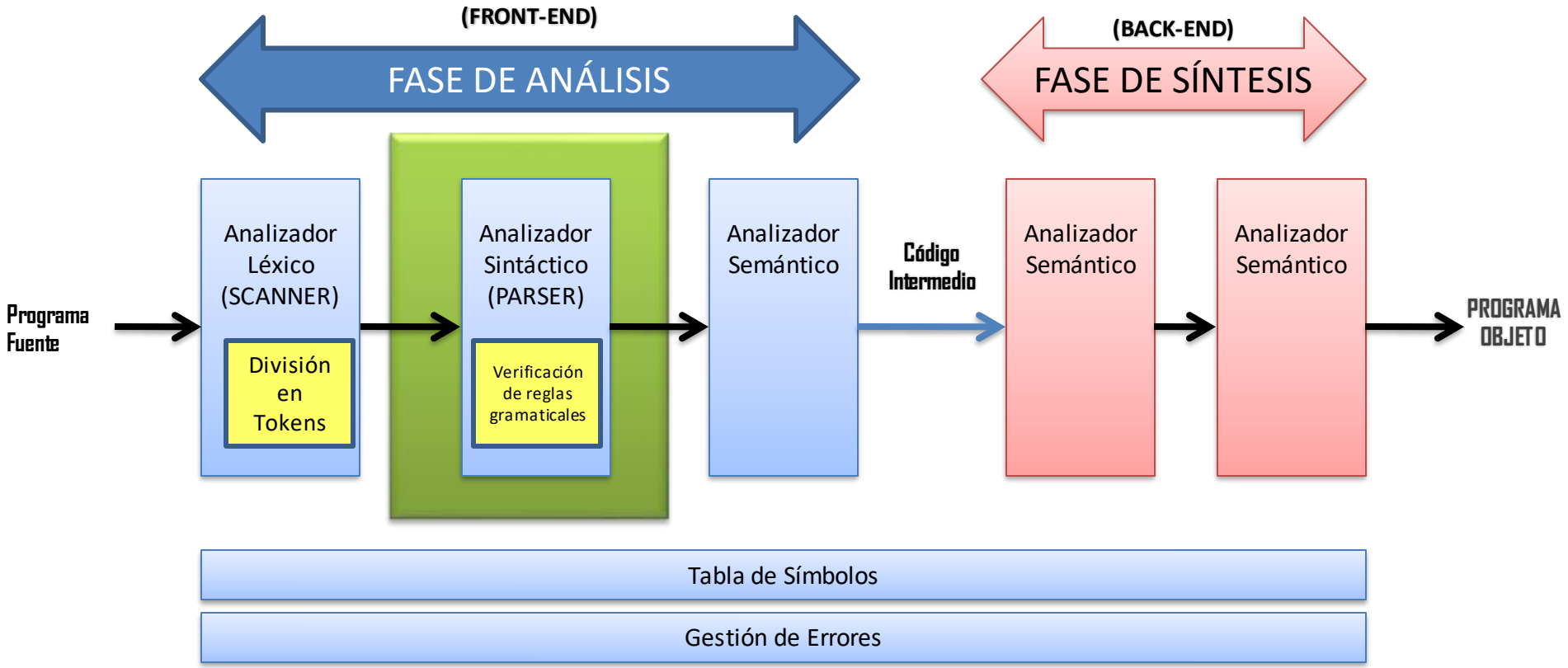
Sesión 6
Analizador Sintáctico
ASD Predictivo LL

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...

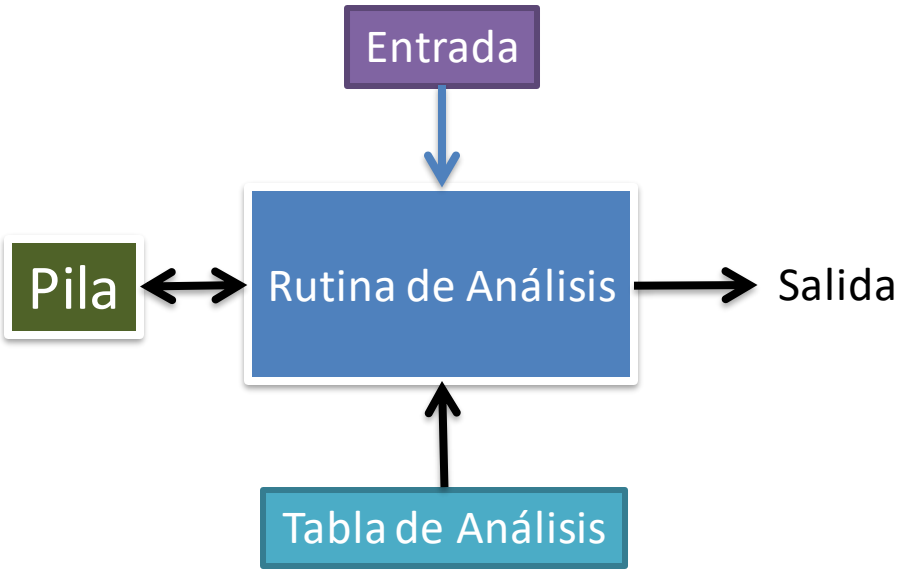
- Ya sabemos
  - Que el ASD Recursivo tiene limitaciones por los retrocesos
  - Que el ASD Predictivo elimina esos retrocesos mediante una tabla de análisis sintáctico donde se predice el salto para un símbolo de entrada.
- Lo que queremos saber
  - ¿Cómo construimos esa tabla de análisis sintáctico para el ASD Predictivo?

RECORDANDO: 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.

RECORDANDO: 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.**

RECORDANDO: 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

RECORDANDO: 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.

Clasificación de gramáticas y lenguajes

- Los lenguajes definidos por las gramáticas se clasifican en función de las posibilidades de los métodos de análisis de los mismos:

Analizador LL(1)

- Una gramática es **LL(1)** si la tabla de análisis sintáctico asociada tiene como máximo una producción en cada entrada de la tabla
  - Es decir, sólo se necesita mirar el primer token (testigo) de la entrada para decidir qué regla aplicar
- El análisis descendente más común es LL(1) pues con k>1 la tabla de análisis es mucho más complicada
  - No se usan LL(2), LL(3), etc. en la práctica
- Sin embargo, no todas las gramáticas se pueden analizar mediante métodos descendentes predictivos.

Para determinar si a una gramática se le puede aplicar el método **LL(1)**, debe comprobarse que:

- No es recursiva por la izquierda
- Si existe un conjunto de reglas de la forma $A \rightarrow A\alpha_1 \mid \alpha_2 \mid \ldots \mid \alpha_n$, es posible decidir la opción a escoger mirando un token de la entrada, i.e., dos productores de un mismo terminal no pueden dar lugar al mismo testigo (símbolo terminal) inicial.

Analizador LL(1) Conjuntos de Predicción

Para construir la Tabla de Análisis y probar si la gramática es LL(1) se construyen dos conjuntos:

A. Conjuntos de **Primeros.**

B. Conjuntos de **Siguientes.**

- Los conjuntos **Primeros** se calculan para **todos los símbolos** de la gramática(terminales y no terminales). Los Primeros de los símbolos terminales son triviales, al ser el mismo símbolo.

- Los conjuntos **Siguiente** se definen sólo para los **no terminales**.

Analizador LL(1) - Conjunto Primero

Dado un símbolo de la gramática A, su **conjunto Primero** es el conjunto de terminales por los que comienza cualquier frase que se genere a partir de A mediante derivaciones por la izquierda de las producciones.

$$a \in Primero(\alpha_p) \text{ si } a \in (V_t \cup \varepsilon) \mid \alpha_p \rightarrow^* a\,\beta$$

- Algoritmo de cálculo del conjunto primero:
  - Si $A \in V_t$ => Prim(A) = A
  - Si $\exists$ $A \rightarrow \varepsilon$ => $\varepsilon \in$ Prim(A)
  - Si $\exists$ $A \rightarrow Y_1Y_2...Y_n$ =>{Prim(Y_1) - ε} $\in$ Prim(A)
  - Si $\varepsilon \in$ Prim (Y_1) =>{Prim (Y_2) - ε} $\in$ Prim(A)
  - Si $\varepsilon \in$ Prim (Y_2) =>{Prim (Y_3) - ε} $\in$ Prim(A)
  - …
  - Si $\forall$ i, $\varepsilon \in$ Prim(Y_i) => $\varepsilon \in$ Prim(A).

Analizador LL(1) - Conjunto Primero

Solo símbolos terminales o épsilon que puedan aparecer en la primera posición aplicando las reglas de derivación

Ejemplo 1

Para la gramática:

$A \rightarrow aBc$

$A \rightarrow xC$

$A \rightarrow B$

$B \rightarrow bA$

$C \rightarrow c$

Conjuntos *Primeros:*

- $Prim(A) = \{a, x\} \cup \{Prim(B) - \varepsilon\} = \{a, x, b\}$
- $Prim(B) = \{b\}$
- $Prim(C) = \{c\}$

Analizador LL(1) - Conjunto Primero

Ejemplo 2

Para la gramática:

E → T E'
E' → + T E' | ε
T → F T'
T' → * F T' | ε
F → (E) | n

Conjuntos *Primeros:*

- PRIMERO(E) = PRIMERO(TE') (*en adelante escribiremos simplemente PRIMERO(E)*).
  PRIMERO(E) = PRIMERO(T), así que pasamos a calcular PRIMERO(T).
- PRIMERO(T) = PRIMERO(F), así que pasamos a calcular PRIMERO(F).
- PRIMERO(F) = { (, n }.

  En este punto ya sabemos que:
- PRIMERO(E) = PRIMERO(T) = PRIMERO(F)={ **(, n**}
- PRIMERO(E') = { **+ , ε** }.
- PRIMERO(T') = { *** , ε** }.

Analizador LL(1) - Conjunto Siguientes

Dado un símbolo **A, su Conjunto SIGUIENTE** es el conjunto de terminales que aparecen inmediatamente después de **A** en alguna sentencia por la que se pasa al realizar una derivación por la izquierda.

$$a \in \text{Siguientes } (A) \text{ si } a \in (V_t \cup \{\$\}) \mid S \rightarrow^* \ldots Aa \ldots$$

- Algoritmo de cálculo del conjunto de Siguientes aplicando las siguientes reglas :

1. Siguiente (S) = $, siendo S es el axioma.

– Para cada símbolo **no-terminal** A de la gramática, añadir

2. Si $\exists$ A → **αBβ** => {Primero(β) – ε } $\in$ Siguiente(B)
3. Si $\exists$ A → **αB** ó (A → αBβ donde $\varepsilon \in$ Prim(β)) => Sig (B) $\in$ Sig (A)

Nota : ε nunca es elemento de un conjunto siguiente.

Analizador LL(1) - Conjunto Siguientes

Ejemplo 1

Para la gramática:

E → T E'
E' → + T E' | ε
T → F T'
T' → * F T' | ε
F → (E) | n

Se aplica a los no terminales de la gramática. Devuelve el conjunto de terminales que aparecen a continuación de **A** en alguna forma sentencial derivada del símbolo inicial y el símbolo ($) que representa el final de la cadena de entrada.

Prim(E)=Prim(T)=Prim(F)={(, n} ; Prim(E')={+, ε }; Prim(T')={+, ε }

Conjuntos *Siguientes :*

- SIGUIENTE(E) = {$ + SIGUIENTE(E)} = SIGUIENTE(E) U {) } = { $,) } ,hemos buscado E en las partes derechas de las reglas y hemos encontrado F → (E) , comprobando que PRIMERO( )) = {) }. Como E no aparece en la parte derecha de ninguna otra regla, cerramos su conjunto.
- SIGUIENTE(E') = SIGUIENTE(E) = { $,) } , E' aparece el último en la parte derecha de dos reglas de producción, aunque el No Terminal que las origina es el mismo: E. Por tanto, se añade SIGUIENTE(E) a SIGUIENTE(E') y, como E' no aparece más, cerramos su conjunto.
- SIGUIENTE(T) = PRIMERO(E') U SIGUIENTE(E') = { + , $,) }, hemos aplicado E → +TE' pues el No Terminal T podría ser el último a la derecha, ya que E' deriva la palabra vacía.
- SIGUIENTE(T') = SIGUIENTE(T) = { + , $,) }.
- SIGUIENTE(F) = PRIMERO(T') U SIGUIENTE(T) U SIGUIENTE(T') = { * ,+ , $,) }.

El resultado será:

SIGUIENTE(E) = { $,) } , SIGUIENTE(E') = { $,) } , SIGUIENTE(T) = { + , $,) }
SIGUIENTE(T') = { + , $,) } , SIGUIENTE(F) = { * , + , $,) }

Analizador LL(1) Construcción de la tabla de análisis

REGLAS de construcción de la Tabla de Análisis (M [Vn,Vt])

Repita todo este proceso para cada producción **A→α** de la gramática

1. Para cada terminal **a ∈ Primero(α)**:
   - Agregue A→α a la tabla M[A,a]
2. Si **ε ∈ Primero(α)** (porque α deriva ε) :
   - Agregar la producción A→α en la posición M[A,b] de la tabla para cada terminal b ∈ Siguiente(A).
3. Si **ε ∈ Primero(α) y $ ∈ Siguiente(A)** :
   - Agregue **A→α** en la posición M[A,$]

Notas:

- Cada entrada en blanco es una posición de error
- Si alguna de las entradas tiene más de una producción es porque la gramática no es LL(1)

Analizador LL(1) - Ejemplo

1. Comprobar si la siguiente gramática es LL(1) y construir la tabla, calculando todos los conjuntos (Siguiente y Primero).

   $S \rightarrow c\ A$

   $A \rightarrow a\ B$

   $B \rightarrow b \mid \varepsilon$

2. Reconocer la cadena "**cab**" con el analizador construido.

Analizador LL(1) - Ejemplo

Pasos para ver si es LL(1):

1. No es recursiva izquierda
2. $B \rightarrow b \mid \varepsilon$
   I. $Prim(b) \cap Prim(\varepsilon) = \emptyset$
   II. Si $\varepsilon \in Prim(\varepsilon) \Rightarrow Prim(b) \cap Sig(B) = \emptyset$

Para determinar si a una gramática se le puede aplicar el método LL(1), debe comprobarse que:

1. No es recursiva por la izquierda
2. Si existe un conjunto de reglas de la forma $A \rightarrow A\alpha_1 \mid \alpha_2 \mid \ldots \mid \alpha_n$ se cumple que no hay dos partes derechas que comiencen por un mismo terminal:
   $Primero(\alpha_i) \cap Primero(\alpha_j) = \{\}, \forall \; i \neq j$
   - A lo sumo un α_i puede derivar la cadena vacía .
   - Si un no terminal $\alpha_i \Rightarrow \varepsilon$, entonces no hay otro α_j que comience por un elemento que esté en Siguiente(A):
     $Primero(\alpha_i) \cap Siguiente(A) = \emptyset$

Conjuntos primero:

Prim(S)={c} , Prim(A)={a} , Prim(B)={b, ε}

Conjuntos siguiente:

Sig(S) = {$} , Sig(A) = Sig(S) = {$} , Sig(B) = Sig(A) = {$}

Analizador LL(1) - Ejemplo

Gramática

S → c A

A → a B

B → b | ε

Prim(S)={c} , Prim(A)={a} , Prim(B)={b, ε }

Sig(S) = {$} , Sig(A) = Sig(S) = {$} , Sig(B) = Sig(A) = {$}

Tabla de Análisis

Regla1

Para cada terminal **a ∈ Primero(α)**:
Agregue A→α a la tabla M[A,a]

	c	**a**	**b**	**$**
S	S → c A			
A		A → a B		
B			B → b	B → ε

Si **ε ∈ Primero(α)** y **$ ∈ Siguiente(A)**:
Agregue **A→α** en la posición M[A,$]

Regla 3

Analizador LL(1) - Ejemplo

Reconocer la cadena **“cab”**

Gramática

S → c A

A → a B

B → b | ε

Pila	Acción	Entrada
$S		cab$
$Ac	Reconocimiento	cab$
$A	A → a B	ab$
$Ba	Reconocimiento	ab$
$B	B → b	b$
$b	Reconocimiento	b$
$	éxito	$

Analizador LL(1) – Ejemplo 2

Pasos para ver si es LL(1):

1. Comprobar que No es recursiva por la izquierda

   Cuando se da alguno de estos casos hay que modificar la gramática:

   1. Eliminando la recursión por la izquierda
   2. Sacando algún factor común por la izquierda

Para determinar si a una gramática se le puede aplicar el método LL(1), debe comprobarse que:

1. No es recursiva por la izquierda
2. Si existe un conjunto de reglas de la forma $A \rightarrow A\alpha_1 \mid \alpha_2 \mid \ldots \mid \alpha_n$ se cumple que no hay dos partes derechas que comiencen por un mismo terminal:

   Primero $(\alpha_i) \cap$ Primero $(\alpha_j) = \{\}, \forall \quad i \neq j$

   • A lo sumo un α_i puede derivar la cadena vacía .

   • Si un no terminal $\alpha_i \Rightarrow \varepsilon$, entonces no hay otro α_j que comience por un elemento que esté en Siguiente(A):

   Primero $(\alpha_i) \cap$ Siguiente $(A) = \emptyset$

Gramática original	Gramática modificada
A → A a A → bB	A →bB A' A' → aA' | ε se elimina la recursividad
B → b c B → b b B → b	B → b B' B' →c | b | ε se saca los factores común y se agrupan

Resumen de Primeros y Siguientes

- **Cálculo del conjunto Primero**

  Primero(α) es el conjunto de *tokens* que están al principio de las hileras derivadas a partir de α.

  Si α ==>* ε, entonces ε ∈ Primero(α)

  X es cualquier símbolo gramatical Comience con Primero(X) == { }

  – Si X==a es terminal Primero(X) == {X} == {a}

  – Si X→ ε es una producción, agregue ε a Primero(X)

  – Si X es no terminal y $X \rightarrow Y_1\ Y_2 \ldots Y_n$ es una producción
    Repita desde 1 hasta n lo siguiente (1 ≤ i ≤ n)
    - Agregue Primero(Y_i) - { ε } a Primero(X)
    - Si ε ∉ Primero(Y_i) termine el cálculo
    - Si llegó hasta (i == n)
      Agregue ε a Primero(X) (pues Y_i deriva ε para todos los valores 1 ≤ i ≤ n)

  – Si ε ∈ Primero(X) es porque X deriva ε

http://www.di-mare.com/pub/ayc/

Resumen de Primeros y Siguientes

- **Cálculo del conjunto Siguiente**

  Siguiente(B) se calcula únicamente para los no terminales B.

  - Calcule Primero(X) para todo símbolo gramatical X.
  - Agregue $ a Siguiente(S) para el símbolo inicial S
  - Si hay una producción **A→αBβ**, todo lo que esté en Primero(β) excepto ε se pone en Siguiente(B):
    Siguiente(B) ← Siguiente(B) ∪ (Primero(β) - { ε })
  - Si hay una producción **A→αB** o una producción **A→αBβ** donde ε ∈ Primero(β) (porque β deriva ε):
    Agregue en Siguiente(B) todo lo que esté en Siguiente(A)
    Siguiente(B) ← Siguiente(B) ∪ Siguiente(A)

http://www.di-mare.com/pub/ayc/

Resumen de Primeros y Siguientes

Construcción de las Tablas LL(1)

- Repita todo este proceso para cada producción A→α de la gramática para cada terminal a ∈ Primero(α)
  1. Agregue A→α a la tabla M[A,a]
  2. Si ε ∈ Primero(α) (porque α deriva ε)
     Agregue la producción A→α en la posición M[A,b] de la tabla M[] para cada terminal b ∈ Siguiente(A).
  3. Si ε ∈ Primero(α) y $ ∈ Siguiente(A)
     Agregue A→α en la posición M[A,$]
- Cada entrada en blanco en M[] es una posición de error
- Si alguna de las entradas de M[] tiene más de una producción es porque la gramática no es LL(1)

http://www.di-mare.com/pub/ayc/

Ejercicios

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

Ejercicios

- Dada la gramática

$$A \rightarrow B\ C$$
$$B \rightarrow D\ E$$
$$C \rightarrow \varepsilon \mid z\ B\ C$$
$$D \rightarrow y\ A\ x \mid v$$
$$E \rightarrow w\ D\ E \mid \varepsilon$$

- Realice en proceso de análisis sintáctico descendente predictivo para reconocer las expresiones
  - ywx
  - v
  - yvxwvzvw

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.

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).