



**Universidad
Europea de Madrid**

LAUREATE INTERNATIONAL UNIVERSITIES

ANÁLISIS SINTÁCTICO II

SLR

© Todos los derechos de propiedad intelectual de esta obra pertenecen en exclusiva a la Universidad Europea de Madrid, S.L.U. Queda terminantemente prohibida la reproducción, puesta a disposición del público y en general cualquier otra forma de explotación de toda o parte de la misma.

La utilización no autorizada de esta obra, así como los perjuicios ocasionados en los derechos de propiedad intelectual e industrial de la Universidad Europea de Madrid, S.L.U., darán lugar al ejercicio de las acciones que legalmente le correspondan y, en su caso, a las responsabilidades que de dicho ejercicio se deriven.

Índice

Presentación	4
Funcionamiento básico del analizador SLR	5
Reconocimiento de una sentencia	7
Conceptos básicos SLR	8
¿En qué consiste el autómata LR(0)?	8
¿Qué significa este punto desplazándose a lo largo de la producción?	8
Operaciones básicas	10
Operación cerradura	10
Ejemplo con la gramática Operación ir_a:	10
Construcción de conjuntos de elementos LR(0)	13
Construcción de la tabla de análisis SLR	16
¿Cómo sabemos si la gramática para la que se ha obtenido la tabla de análisis sintáctico no es ambigua?	16
Ejemplo de construcción de tabla de análisis SLR	18
Límites del método SLR	23
Resumen	24

Presentación

El objetivo de este tema es **entender el funcionamiento del análisis sintáctico ascendente SLR, también denominado LR sencillo (en inglés simple LR)**. Comenzaremos entendiendo cómo se reconoce una frase a partir del funcionamiento del autómata a pila que lo gestiona, siendo este método válido para todos los analizadores sintácticos ascendentes, y continuaremos viendo cómo se construye el AFD LR (0).

Entenderemos los conceptos y operaciones básicos de un analizador SLR y aprenderemos a construir la tabla de análisis SLR, la cual revisaremos mediante un ejemplo. Además conoceremos qué problemas o límites presentan los analizadores SLR.

Los objetivos a conseguir al finalizar este tema se resumen en los siguientes conceptos:

- Conocer el funcionamiento básico de un analizador SLR.
- Entender cómo se reconoce una sentencia.
- Aprender los conceptos necesarios para el SLR y sus operaciones básicas.
- Aprender cómo se construyen los conjuntos de elementos LR (0).
- Aprender a construir la tabla de análisis SLR.
- Conocer los límites del método SLR.



Funcionamiento básico del analizador SLR

El analizador sintáctico ascendente SLR, también denominado LR sencillo (del inglés simple LR), se basa en el AFD LR(0). Sin embargo, incrementa de manera importante la potencia del análisis sintáctico LR(0) al utilizar el *token* siguiente en la cadena de entrada para dirigir sus acciones (Louden, 2004).

Por defecto, se considera que los LR llevan un símbolo de análisis por anticipado de la entrada (símbolo de *lookahead*), por eso algunos autores lo denominan SLR (1) o LR(1) sencillo. En nuestro caso, lo denominaremos SLR para no crear confusión con el método LR(1), que es diferente del LR(1) sencillo, puesto que el símbolo por anticipado lo selecciona de otra manera.

Al igual que el analizador sintáctico descendente LL(1) utiliza una pila para hacer el análisis sintáctico, donde además de los símbolos de la gramática (terminales y no terminales) se incorporan los estados (S_n), basándose en la arquitectura del autómata a pila. También utiliza una tabla de análisis, pero con dos divisiones (**acción** e **ir_a**) que indica si la entrada es correcta y qué acción realizar con ella.



[Esquema básico de funcionamiento \(Aho et al. 1986\)](#)

El objetivo del analizador es reconocer si la frase pertenece a la gramática y para ello utiliza una pila para almacenar los distintos símbolos gramaticales (terminales y no terminales) donde incorpora el estado

0 (S_0) en lugar de \$ en el inicio de la pila. En la entrada nos encontramos \$ al final de la cadena.

Es ahora la tabla la que indica si para un estado en concreto la acción es ACEPTAR la entrada y terminar el análisis con éxito. La parte de acción de la tabla de análisis indica una acción del analizador, mientras que la parte de ir_a indica transiciones entre los estados (contendrá los números de estos estados).

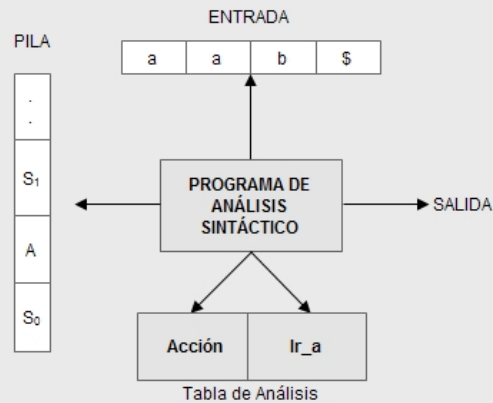


[Entradas de la parte de acción](#)



(S_n)Se utiliza S para los estados (*State*)**Gráfico**

El esquema básico de funcionamiento se representa por la siguiente figura (Aho et al, 1986):

**En detalle**

La parte de acción tendrá las siguientes entradas:

1. **Desplazar** un estado (dX, siendo X el estado).
2. **Reducir** por una producción de la gramática (rX, siendo X la producción de la gramática una vez se numere).
3. **Aceptar** la cadena de entrada.
4. **Error** (las casillas vacías).

Veremos un ejemplo de funcionamiento a partir de una tabla ya construida en la siguiente pantalla, es decir, lo que significan estas acciones.

Reconocimiento de una sentencia

Para entender cómo funciona el análisis sintáctico LR utilizando el autómat a pila y la tabla de análisis, partiremos de una sentencia a reconocer y la [tabla de análisis](#) con sus dos partes. La gramática será la relacionada con las listas y a la que es necesario numerar sus producciones y aumentarla con la producción 1 debido a la forma en la que trabajan todos los analizadores LR:

<ol style="list-style-type: none"> 1. $S' \rightarrow S$ 2. $S \rightarrow (L)$ 3. $S \rightarrow id$ 4. $L \rightarrow S L'$ 5. $L' \rightarrow , S L'$ 6. $L' \rightarrow \lambda$ 	Sentencia a reconocer: (a, b)
--	-------------------------------

Partimos de la siguiente tabla de análisis, que es una matriz (M) de dos dimensiones:

Estados	ACCIÓN					IR_A		
	()	id	,	\$	S	L	L'
0	d2		d3			1		
1					ACEPTAR			
2	d2		d3			5	4	
3		r3		r3	r3			
4		d6						
5		r6		d8				7
6		r2		r2	r2			
7		r4						
8	d2		d3			9		
9		r6		d8				10
10		r5						



Reconocimiento de una sentencia

Documentos

Tabla de análisis

Posteriormente veremos cómo se construye esta tabla.

Conceptos básicos SLR

El analizador sintáctico SLR utiliza para su funcionamiento un autómata finito determinista denominado LR(0), aunque mejorado puesto que incorpora el conjunto SIGUIENTE en la construcción de la tabla de análisis, para conocer por anticipado cuál es el "siguiente" símbolo que debería encontrarse el analizador. Esto implica que la gramática no puede ser ambigua.

¿En qué consiste el autómata LR(0)?

Para construir este AFD LR(0), se utiliza la **gramática aumentada** y dos operaciones: la operación **cerradura** y la operación **ir_a**. La gramática aumentada consiste en añadir una producción inicial en la que si es S el símbolo inicial, la producción que se añade es $S' \rightarrow S$. Esta producción es el equivalente a \$ en las gramáticas LL(1) y sirve para indicar al analizador cuando se ha reconocido una sentencia y por tanto detener el análisis sintáctico.

Este AFD LR(0) se compone de elementos, con los que posteriormente construiremos conjuntos y estos a su vez representarán los estados del autómata.



[Representación](#)

En detalle

Y para la producción $L \rightarrow \lambda$, ¿cuál sería el elemento LR(0)? $L \rightarrow \cdot$.

¿Qué significa este punto desplazándose a lo largo de la producción?

El punto indica en cada estado del elemento, qué parte de la producción ha sido reconocida (lo que está a su izquierda) y qué parte falta por reconocer (lo que está en el lado derecho del punto), indicando por tanto qué símbolos se espera que vengan a continuación. Esto se denomina **prefijo viable** e identifica la parte que se ha reconocido.

En el ejemplo, el elemento $S \rightarrow (\cdot L)$ indica que se acaba de desplazar a la pila (por tanto se ha reconocido el símbolo) el paréntesis de apertura y a continuación se espera una cadena que se pueda derivar a partir de L. Es decir que el paréntesis de apertura es el prefijo viable. Cuando el punto está en el lado derecho, $S \rightarrow (L) \cdot$, quiere decir que el elemento está completo, por tanto se ha reconocido por completo esta producción y hay que reducir.

En detalle**Representación**

Estos elementos LR (0) se construyen a partir de una producción con un punto en alguna posición de su parte derecha:

Ejemplo: $S \rightarrow (L)$, donde los posible elementos que se derivan de esta producción son:

$S \rightarrow . (L)$

$S \rightarrow (. L)$

$S \rightarrow (L .)$

$S \rightarrow (L) .$

Operaciones básicas

Las dos operaciones básicas de un AFD LR(0) son: **cerradura** e **ir_a**.

Operación cerradura

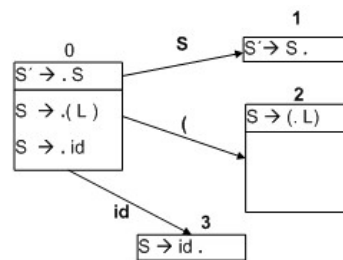
Esta operación se realiza cada vez que tenemos un punto a la izquierda de un no terminal para una producción cualquiera de una gramática, del tipo $A \rightarrow \alpha . N \beta$, (donde tanto α como β representan cadenas de terminales y no terminales incluyendo la cadena vacía y N representa un no terminal). En este caso el no terminal N , por tener un punto a su izquierda, se le tiene que hacer la operación cerradura, con el objeto de averiguar qué cadenas se pueden derivar a partir de él y consiste en obtener todas las producciones en las que N está en el lado derecho (ej: $N \rightarrow . \omega$) de las mismas a las que inicializa con un punto al inicio del lado derecho de cada de estas producciones (si es que hubiera más de una).

 [Procedimiento](#)
En detalle

 [Conjunto que representa un estado del AFD](#)
Ejemplo

Operación ir_a:

Ahora necesitamos una operación que nos ayude a pasar de un estado a otro del AFD. Una vez hemos construido un conjunto de elementos (o estado) a partir de la operación cerradura, la operación **ir_a** consiste en generar un estado nuevo desde cada elemento que tiene un punto en su lado derecho y donde este punto no ha llegado al final de la producción.



Ejemplo con la gramática

 [Procedimiento](#)
En detalle

En detalle

Procedimiento

Esto se hará de forma sucesiva hasta que no queden no terminales con un punto a su izquierda. Es decir si la producción de la que se deriva N , tiene la forma $N \rightarrow C \omega$, entonces tendríamos que añadir $N \rightarrow . C \omega$, y realizar la cerradura de C . Aplicaríamos esta regla hasta que no se puedan añadir más elementos a cerradura.

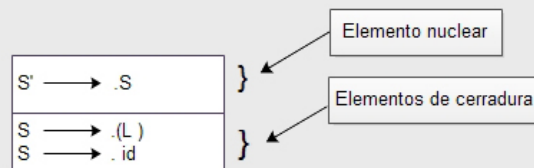
Ejemplo

Con todos los elementos LR(0) obtenidos de la operación cerradura constituimos un conjunto, que representa un estado del AFD. Un ejemplo sería el siguiente:

1. $S' \rightarrow S$
2. $S \rightarrow (L)$
3. $S \rightarrow id$
4. $L \rightarrow S L'$
5. $L' \rightarrow , S L'$
6. $L' \rightarrow \lambda$

El conjunto de elementos que forman el estado comienza incluyendo en el estado 0 el primer elemento, con un punto a la izquierda del lado derecho de la producción: $S' \rightarrow . S$. Este primer elemento constituye el **elemento nuclear** (o núcleo del conjunto) a partir del cual se generarán los demás elementos de ese estado haciendo la cerradura del mismo y que consistirá en incluir todas las producciones donde S esté en el lado izquierdo: $S \rightarrow . ($ L) y $S \rightarrow . id$

Como tanto el paréntesis derecho como id son terminales, estos elementos no generan más producciones para el estado 0. Esto se representaría así:



NOTA: si el punto hubiera quedado a la izquierda de un no terminal seguiríamos haciendo la operación cerradura con este nuevo símbolo.

No ha llegado al final de la producción

Esto último generaría un elemento completo y ya no se avanzaría el punto.

En detalle**Procedimiento**

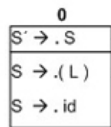
Es importante destacar que esto se hace para cada símbolo que tiene un punto en el lado izquierdo, es decir que si hay más de un elemento que tiene el mismo símbolo con un punto en su lado izquierdo, pasaríamos al siguiente estado con estos elementos de los que procede el símbolo con el punto en su lado izquierdo, formando el núcleo de este nuevo estado.

Al avanzar al siguiente estado donde volveremos a hacer la operación cerradura de los elementos que constituyen el núcleo, debemos avanzar el punto una posición y etiquetamos los arcos con los símbolos que han generado la operación `ir_a`.


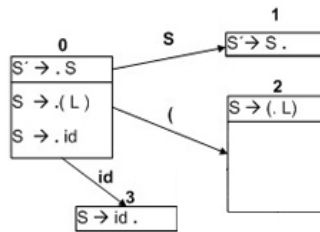

Construcción de conjuntos de elementos LR(0)

Una vez que sabemos cómo aplicar las operaciones de cerradura e ir_a, para avanzar en los estados, construimos los conjuntos de elementos LR(0) aplicándolas hasta que en todos los conjuntos de elementos (estados) no generen conjuntos nuevos y donde todas las producciones hayan generado elementos completos.

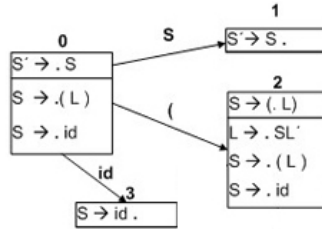
Hacemos la operación cerradura de la primera producción con la que hemos aumentado la gramática.

1/6 

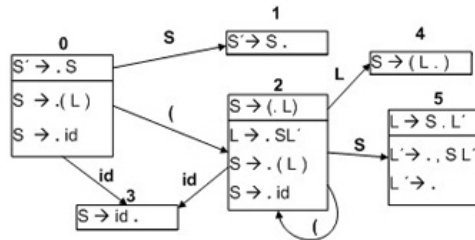
Hacemos la operación ir_a a cada uno de los símbolos que tienen un punto en su lado izquierdo. Obtenemos dos estados completos, el 1 y el 3, por tanto con estos no avanzamos.

 2/6 

Se obtiene S con un punto a su lado izquierdo lo que indica que hay que hacer su cerradura también.

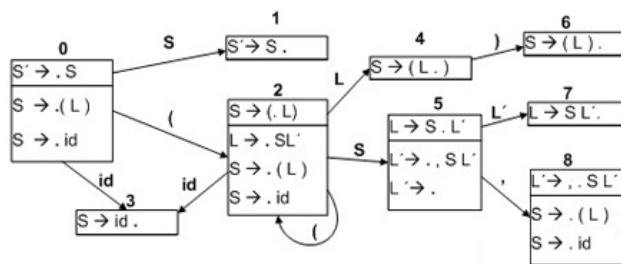


3/6



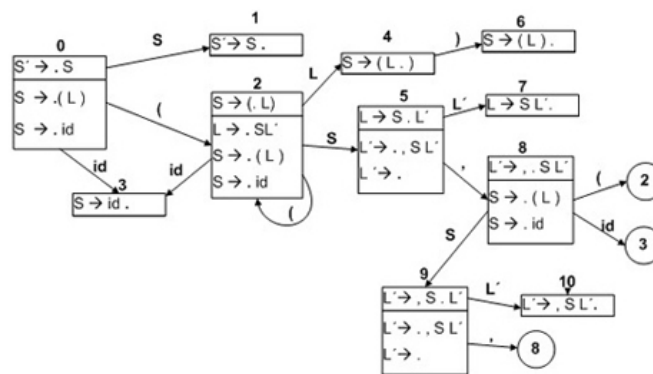
Vemos como los símbolos que producen estados ya existente al aplicarles la operación ir_a generan arcos a esos estados $\{id\}$. También es de mencionar la cerradura de $L' \rightarrow \lambda$, en el estado 5, convirtiéndose en $L' \rightarrow \cdot$

4/6



Hemos obtenido dos estados completos (el 6 y el 7).

5/6



Hemos obtenido todos los estados completos o saltos a estados que ya existen (con el (nos vamos al estado 2, con **id** al 3 y con la **coma** al 8), con lo que no se generan estados nuevos y se termina la creación de conjuntos correspondientes al AFD LR(0) de esta gramática.

6/6



Documentos

Construcción de conjuntos de elementos LR(0)

Elementos completos

Elementos con punto en su extremo derecho como por ejemplo $S' \rightarrow S \cdot$ ó $S \rightarrow id \cdot$.

Para aquellos elementos a los que en un estado les apliquemos la operación ir_a y veamos que ya se encuentren en otro estado formando parte del núcleo se generará un arco a ese estado. Estos arcos irán etiquetados con el símbolo que tiene el punto a su izquierda y genera esta transición, pudiendo ser terminales o no terminales.

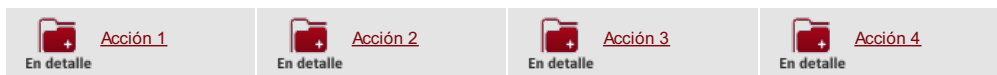
Veremos el ejemplo comentando todos los detalles. Partimos de la gramática aumentada siguiente:

1. $S' \rightarrow S$
2. $S \rightarrow (L)$
3. $S \rightarrow id$
4. $L \rightarrow S L'$
5. $L' \rightarrow , S L'$
6. $L' \rightarrow \lambda$

Construcción de la tabla de análisis SLR

Una vez hemos obtenido el AFD LR(0) se construye la tabla de análisis sintáctico SLR, a partir de los estados obtenidos y donde se obtendrán las acciones a realizar para las entradas de la matriz $M[\text{estado, terminal o no terminal}]$, donde M podrá ser la parte **acción** si los símbolos son terminales o \$, o bien la de **ir_a** si se trata de un no terminal.

El algoritmo de construcción consiste en realizar las acciones siguientes para cada uno de los estados:



¿Cómo sabemos si la gramática para la que se ha obtenido la tabla de análisis sintáctico no es ambigua?

Si en las entradas de la tabla no coinciden dos acciones, entonces la gramática es SLR y por tanto no ambigua. Si hubiera más de una entrada en la tabla se produciría un conflicto, que puede ser de dos tipos:

Conflicto de reducción/desplazamiento	Se produce cuando en un mismo estado existe una producción con un elemento completo, del tipo $A \rightarrow \alpha.$, y otra producción del tipo $A \rightarrow \alpha.T\beta$. Al calcular el conjunto SIGUIENTE(A), alguno de los símbolos es T, produciéndose el conflicto.
Conflicto de reducción/reducción	Se produce cuando en un mismo estado existen dos producciones con el elemento completo, del tipo $A \rightarrow \alpha.$, no pudiendo por tanto decidirse que reducción aplicar.

Realizando una simple inspección visual sobre los conjuntos de elementos y fijándonos en los que tienen elementos completos es fácil de ver si hay conflictos o no.

En detalle

Acción 1

Si el estado S contiene un elemento de la forma $A \rightarrow \alpha.T\beta$, donde T es un terminal, se inserta en la matriz **acción**[S, T] la acción de desplazar al estado al que va el arco etiquetado con ese terminal (dS_n) y que será el que contenga como núcleo $A \rightarrow \alpha.T.\beta$. La acción de desplazar indica que ese *token* se desplaza a la pila seguido del estado al que va el arco obtenido en el AFD LR(0).

En detalle**Acción 2**

Si el estado S contiene un elemento de la forma $A \rightarrow \alpha.N\beta$, donde N es un no terminal se inserta en la matriz $ir_a[S, N]$ el número del estado al que va el arco etiquetado con ese no terminal (S_n) y que será el que contenga como núcleo $A \rightarrow \alpha.N\beta$. Estas acciones en la parte de ir_a son transiciones del AFD por el no terminal correspondiente y el efecto que producen es la introducción en la cima de la pila de ese estado de transición para ese no terminal N .

En detalle**Acción 3**

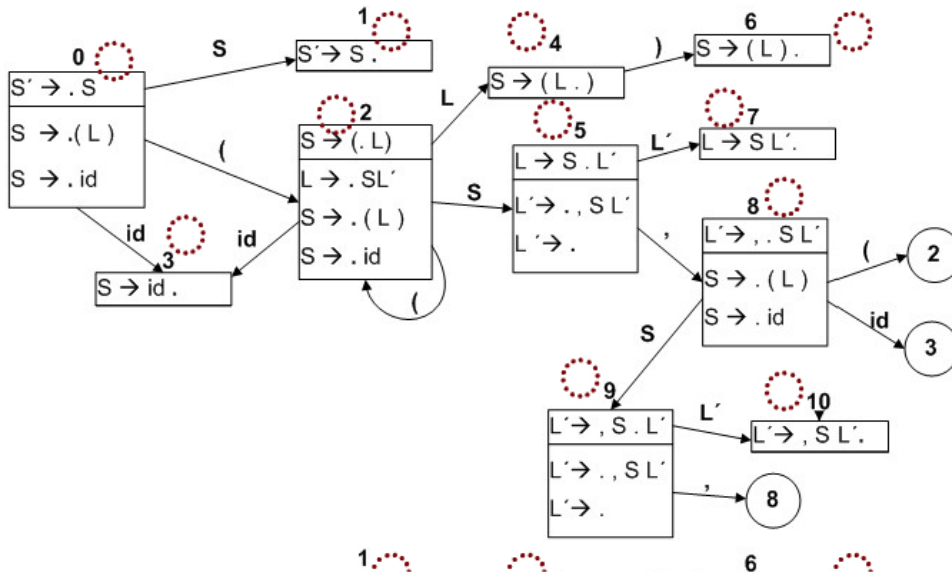
Si el estado S contiene un **elemento completo** de la forma $A \rightarrow \alpha.$, donde α representa cualquier combinación de terminales, no terminales o cadena vacía, la acción a incluir en la matriz es reducir por la producción X , donde X es el número de la producción en la gramática, para cada uno de los elementos que haya en $SIGUIENTE(A)$. Es decir, si los elementos en $SIGUIENTE(A)$ son a_1 y a_2 (siendo estos terminales de la gramática o $\$$), las entradas en la matriz serán **$acción[S, a_1] = acción[S, a_2] = rX$** .

En detalle**Acción 4**

Si el elemento completo se refiere a la producción con la que se ha aumentado la gramática, en nuestro ejemplo $S' \rightarrow S.$, la entrada correspondiente será **$acción[1, \$] = ACEPTAR$** .

Ejemplo de construcción de tabla de análisis SLR

La forma más rápida de construir la tabla es a partir del AFD LR(0) obtenido en el paso 6 anterior, por otro lado, la gramática aumentada y numerada, y por otro el algoritmo de construcción de la tabla.



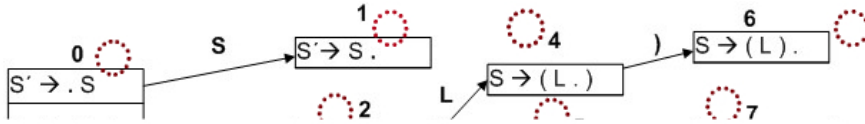
Estado 0: Mirando el LR(0) y aplicando el algoritmo, nos fijamos en la regla 1 ($A \rightarrow \alpha.T\beta$) y en el AFD LR(0) y vemos que tenemos las siguientes producciones:

- $S \rightarrow .(L)$, donde con el paréntesis de apertura vamos al estado 2, es decir, desplazamiento al estado 2 (d2).
- Por otro lado y también para la regla 1, tenemos $S \rightarrow .id$, con la que vamos al estado 3, es decir d3.

Finalmente, para la regla 2 ($A \rightarrow \alpha.N\beta$) tenemos la producción:

$S' \rightarrow .S$, con la que tenemos una transición al estado 1.

Estados	ACCIÓN					IR_A		
	()	id	,	\$	S	L	L'
0	d2		d3			1		
1								
2								
3								
4								
5								
6								
7								
8								
9								
10								



Estado 1: Mirando el LR(0) y aplicando el algoritmo y puesto que estamos tratando la producción correspondiente a la gramática aumentada, se aplica la regla 4:

$S' \rightarrow S.$, acción[1, \$] = ACEPTAR

Estados	ACCIÓN					IR_A		
	()	id	,	\$	S	L	L'
0	d2		d3			1		
1					ACEPTAR			
2								
3								
4								
5								
6								
7								
8								
9								
10								

Estado 2: Mirando el LR(0) y aplicando el algoritmo, nos fijamos en la regla 1 ($A \rightarrow \alpha.T\beta$):

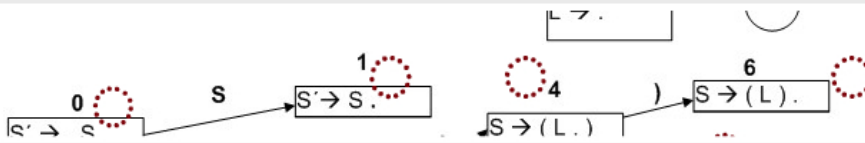
- $S \rightarrow (.L)$, donde con el paréntesis de apertura vamos al estado 2, (d2), nos quedamos donde estamos.
- Por otro lado y también para la regla 1, tenemos $S \rightarrow .id$, con la que vamos al estado 3, es decir d3.

Para la regla 2 ($A \rightarrow \alpha.N\beta$):

- $S \rightarrow (.L)$, con la que tenemos una transición al estado 4.

$L \rightarrow .SL'$, tenemos una transición al estado 5.

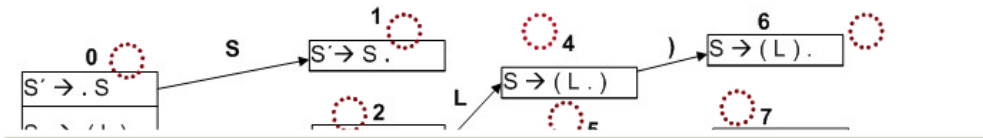
Estados	ACCIÓN					IR_A		
	()	id	,	\$	S	L	L'
0	d2		d3					
1					ACEPTAR			
2	d2		d3			5	4	
3								
4								
5								
6								
7								
8								
9								
10								



Estado 3: Mirando el LR(0) y aplicando el algoritmo, nos fijamos en la regla 3 ($A \rightarrow \alpha.$):

$S \rightarrow id.$, donde tenemos que obtener SIGUIENTE(S)={\$, ,,)} y esta producción es la número 3 en la gramática. Acción[3,\$] = Acción[3, ,] = Acción[3,)] = r3.


Estados	ACCIÓN					IR_A		
	()	id	,	\$	S	L	L'
0	d2		d3			1		
1					ACEPTAR			
2	d2		d3			5	4	
3			r3	r3	r3			
4								
5								
6								
7								
8								
9								
10								



Estado 4: Mirando el LR(0) y aplicando el algoritmo, nos fijamos en la regla 1 ($A \rightarrow \alpha.T\beta$):

$S \rightarrow (L .)$, donde con el paréntesis de cierre nos desplazamos al estado 6, (d6).

Estados	ACCIÓN				IR A			
	()	id	,	\$	S	L	L'
0	d2		d3			1		
1					ACEPTAR			
2	d2		d3			5	4	
3			r3	r3	r3			
4			d6					
5								
6								
7								
8								
9								
10								



Estado 5: Mirando el LR(0) y aplicando el algoritmo, nos fijamos en la regla 1 ($A \rightarrow \alpha.T\beta$):

- $L' \rightarrow . , SL'$ donde con la coma nos desplazamos al estado 8, (d6),

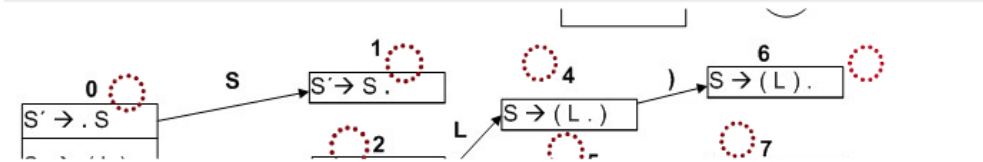
Para la regla 2 ($A \rightarrow \alpha.N\beta$):

- $L \rightarrow S . L'$ donde con L' tenemos una transición al estado 7

Por último para la regla 3 ($A \rightarrow \alpha.$):

$L' \rightarrow .$ donde tenemos que obtener $SIGUIENTE(L') = \{ \}$ y esta producción es la número 6 en la gramática. $Acción[6,] = r6$

Estados	ACCIÓN				IR A			
	()	id	,	\$	S	L	L'
0	d2		d3			1		
1					ACEPTAR			
2	d2		d3			5	4	
3			r3	r3	r3			
4			d6					
5			r3	d8			7	
6								
7								
8								
9								
10								



Estado 6: Mirando el LR(0) y aplicando el algoritmo, nos fijamos en la regla 3 ($A \rightarrow \alpha.$):

$S \rightarrow (L) .$, donde tenemos que obtener $SIGUIENTE(S) = \{ \$, ,, \}$ y esta producción es la número 2 en la gramática. $Acción[6, \$] = Acción[6, ,] = Acción[6,] = r2$.

Estados	ACCIÓN				IR A			
	()	id	,	\$	S	L	L'
0	d2		d3			1		
1					ACEPTAR			
2	d2		d3			5	4	
3			r3	r3	r3			
4			d6					
5			r6	d8			7	
6			r2	r2	r2			
7								
8								
9								
10								

0 → S → S . → 1 → 4 → 6

Estado 7: Mirando el LR(0) y aplicando el algoritmo, nos fijamos en la regla 3 ($A \rightarrow \alpha$):
 $L \rightarrow S L' .$, donde tenemos que obtener $SIGUIENTE(L)=\{ \}$ y esta producción es la número 4 en la gramática. $Acción[7,] = r4$.

Estados	ACCIÓN					IR A		
	()	id	,	\$	S	L	L'
0	d2		d3			1		
1					ACEPTAR			
2	d2		d3			5	4	
3		r3		r3	r3			
4		d6						
5		r6		d8				7
6		r2		r2	r2			
7		r4						
8								
9								
10								

1 → 4 → 6 → 8

Estado 8: Mirando el LR(0) y aplicando el algoritmo, nos fijamos en la regla 1 ($A \rightarrow \alpha.T\beta$) y tenemos las siguientes producciones:

- $S \rightarrow . (L)$, donde con el paréntesis de apertura vamos al estado 2, es decir, desplazamiento al estado 2 (d2).
- $S \rightarrow . id$, con la que vamos al estado 3, es decir d3.

Finalmente, para la regla 2 ($A \rightarrow \alpha.N\beta$) tenemos la producción:

$L' \rightarrow . , S L'$, con la que tenemos una transición al estado 9.

Estados	ACCIÓN					IR A		
	()	id	,	\$	S	L	L'
0	d2		d3			1		
1					ACEPTAR			
2	d2		d3			5	4	
3		r3		r3	r3			
4		d6						
5		r6		d8				7
6		r2		r2	r2			
7		r4						
8	d2		d3			9		
9								
10								

1 → 4 → 6 → 8 → 9

Estado 9: Mirando el LR(0) y aplicando el algoritmo, nos fijamos en la regla 1 ($A \rightarrow \alpha.T\beta$):

- $L' \rightarrow . , S L'$ donde con la coma nos desplazamos al estado 8, (d8),

Para la regla 2 ($A \rightarrow \alpha.N\beta$):

- $L \rightarrow S . L'$ donde con L' tenemos una transición al estado 10.

Por último para la regla 3 ($A \rightarrow \alpha$):

$L' \rightarrow .$ donde tenemos que obtener $SIGUIENTE(L')=\{ \}$ y esta producción es la número 6 en la gramática. $Acción[9,] = r6$.

Estados	ACCIÓN					IR A		
	()	id	,	\$	S	L	L'
0	d2		d3			1		
1					ACEPTAR			
2	d2		d3			5	4	
3		r3		r3	r3			
4		d6						
5		r6		d8				7
6		r2		r2	r2			
7		r4						
8	d2		d3			9		
9		r6		d8				10
10								

Estado 10: Mirando el LR(0) y aplicando el algoritmo, nos fijamos en la regla 3 ($A \rightarrow \alpha$):
 $L' \rightarrow , S L' .$, donde tenemos que obtener $SIGUIENTE(L') = ()$ y esta producción es la número 5 en la gramática. Acción[10, (] = r5.

Estados	ACCIÓN					IR_A		
	()	id	,	\$	S	L	L'
0	d2		d3			1		
1					ACEPTAR			
2	d2		d3			5	4	
3		r3		r3	r3			
4		d6						
5		r6		d8				7
6		r2		r2	r2			
7		r4						
8	d2		d3			9		
9		r6		d8				10
10		r5						

Ejemplo de construcción de tabla de análisis SLR

Gramática aumentada y numerada

1. $S' \rightarrow S$
2. $S \rightarrow (L)$
3. $S \rightarrow id$
4. $L \rightarrow S L'$
5. $L' \rightarrow , S L'$
6. $L' \rightarrow \lambda$

Algoritmo de construcción de la tabla

- Si el estado S contiene un elemento de la forma $A \rightarrow \alpha.T\beta$, donde T es un terminal, se inserta en la matriz acción[S, T] la acción de desplazar al estado al que va el arco etiquetado con ese terminal (dSn) y que será el que contenga como núcleo $A \rightarrow \alpha.T.\beta$. La acción de desplazar indica que ese token se desplaza a la pila seguido del estado al que va el arco obtenido en el AFD LR(0)
- Si el estado S contiene un elemento de la forma $A \rightarrow \alpha.N\beta$, donde N es un no terminal se inserta en la matriz ir_a[S, N] el número del estado al que va el arco etiquetado con ese no terminal (Sn) y que será el que contenga como núcleo $A \rightarrow \alpha.N.\beta$. Estas acciones en la parte de ir_a son transiciones del AFD por el no terminal correspondiente y el efecto que producen es la introducción en la cima de la pila de ese estado de transición para ese no terminal N
- Si el estado S contiene un **elemento completo** de la forma $A \rightarrow \alpha$, donde α representa cualquier combinación de terminales, no terminales o cadena vacía, la acción a incluir en la matriz es reducir por la producción X, donde X es el número de la producción en la gramática, para cada uno de los elementos que haya en SIGUIENTE(A). Es decir, si los elementos en SIGUIENTE(A) son a1 y a2 (siendo estos terminales de la gramática o \$), las entradas en la matriz serán acción[S, a1] = acción[S, a2] = rX
- Si el elemento completo se refiere al la producción con la que se ha aumentado la gramática, en nuestro ejemplo $S' \rightarrow S$, la entrada correspondiente será acción[1, \$] = ACEPTAR

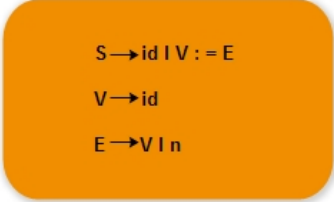
Límites del método SLR

Hay gramáticas no ambiguas que no pueden ser analizadas mediante el análisis sintáctico SLR, es decir, que la gramática no es ambigua pero al construir la tabla de análisis se producen conflictos:

- Conflicto de reducción/desplazamiento.
- Conflicto de reducción/reducción.

Esto no es debido a la forma en la que construye la tabla utilizando como símbolo de anticipación el que se obtiene del conjunto SIGUIENTE, y que le convierte en el método menos poderoso de los tres métodos ascendentes que veremos.

En Louden (2004) se propone una gramática donde se pueden ver los conflictos que se producen en uno de los estados. La gramática que se propone como ejemplo es la siguiente:



$S \rightarrow id \mid V : = E$
 $V \rightarrow id$
 $E \rightarrow V \mid n$

Al construir el AFD LR(0), ejercicio que se propone al estudiante, nos encontramos con un estado donde entre otras producciones coinciden las siguientes: $S \rightarrow id.$ y $V \rightarrow id.$, llegándose así a un conflicto de reducción/reducción. Se llega a un conflicto y la gramática no es ambigua.

Resumen

En este tema hemos visto como se reconoce una sentencia mediante un método de análisis ascendente LR, siendo idéntico procedimiento para el SLR, LR(1) y LALR.

A partir de aquí se han conocido los conceptos básicos como son la gramática aumentada añadiendo una producción a la gramática desde la que se obtiene el símbolo inicial de la gramática (axioma) , las operaciones de cerradura e ir_a y el concepto de prefijo viable representado este concepto por el punto desplazándose a lo largo de las producciones, que indica que parte de la producción ha sido reconocida (a la izquierda del punto) y cuanto queda por reconocerse (la derecha del punto).

La operación cerradura se aplica sobre todos los no terminales que están a la derecha del punto. Esto se hará de forma sucesiva hasta que no queden no terminales con un punto a su izquierda.

La operación ir_a sirve para transitar de un estado a otro del AFD. Una vez hemos construido un conjunto de elementos (o estado) a partir de la operación cerradura, la operación ir_a consiste en generar un estado nuevo desde cada elemento que tiene un punto en su lado derecho y donde éste punto no ha llegado al final de la producción (esto último generaría un elemento completo y ya no se avanzaría el punto).

También se ha visto cómo se construye la tabla de análisis SLR.