Universidad
Europea de Madrid

LAUREATE INTERNATIONAL UNIVERSITIES

ANÁLISIS SINTÁCTICO II

LALR

© 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

© Universidad Europea de Madrid. Todos los derechos reservados.

Presentación

El objetivo de este tema es **entender el funcionamiento del analizador sintáctico LALR.** Comenzaremos entendiendo los conceptos básicos del análisis LALR y cómo se obtienen los estados de fusión.

En este tema es importante entender el algoritmo de análisis LALR y la construcción de la tabla, que es de la misma manera que en el método LR(1).

Para cumplir los objetivos a del tema, abordaremos los siguientes contenidos:

- Conocer el analizador sintáctico ascendente LALR.
- Comprender el algoritmo de modificación del autómata.
- Aprender a construir la tabla de análisis LALR.
- Repasar con un ejercicio completo los tres métodos de análisis sintáctico ascendente.

© Universidad Europea de Madrid. Todos los derechos reservados.

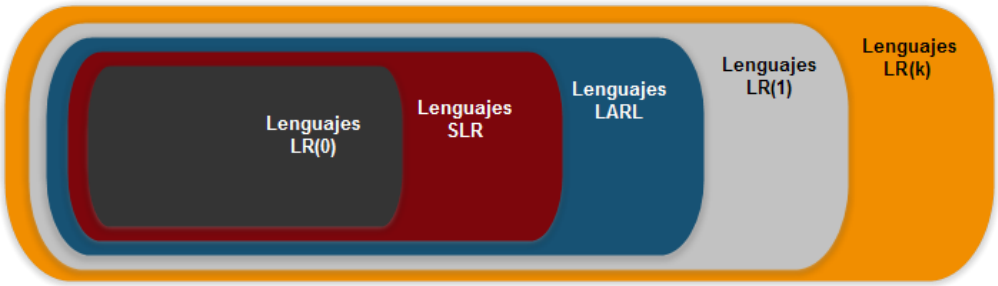
Introducción al analizador LALR

Al realizar análisis sintáctico SLR y LR(1), es importante saber que los LR(1) trabajan con **gramáticas que son ambiguas para los SLR,** aunque para ello necesitan generar autómatas con más estados. Esto se debe a la utilización de los símbolos de anticipación.

Podemos ver como estados que tienen el mismo núcleo, son diferentes debido a los símbolos de anticipación. Con los analizadores sintácticos LALR se trata de conseguir los beneficios que proporciona el método LR(1), **pero con el coste del método SLR** y utilizando, por tanto, el menor número de estados que este último necesita.

Es decir, **este método consigue reducir el número de estados.** Se debe tener en cuenta que, para un lenguaje como Pascal, estamos hablando de varios cientos de estados. Si en el ejemplo que hemos visto para una gramática pequeña el SLR generaba 11 estados y el LR(1) necesitaba 15, podemos ver que necesitamos un 25% más de estados y por tanto de coste a la hora de obtener los mismos resultados.

No todos son beneficios con el LALR, como vemos en el siguiente esquema no todas las gramáticas LR(1) pueden reconocerse con un analizador LALR, son por tanto un subconjunto de las LR(1).



En cuanto a los conflictos que pueden generarse con este tipo de gramáticas, estos son de reducción/reducción, nunca de desplazamiento/reducción.

© Universidad Europea de Madrid. Todos los derechos reservados.

Algoritmo de modificación del autómata

Para entender como funciona el algoritmo de análisis sintáctico LALR, partimos del AFD obtenido con el LR(1), el cual debemos modificar siguiendo el siguiente proceso:

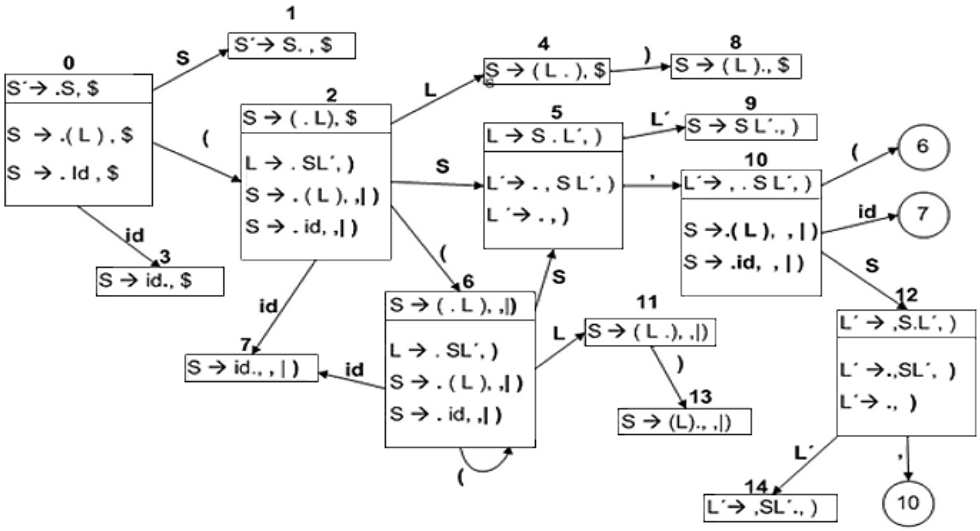
Primer paso	Se seleccionan en el autómata LR(1) los estados con el mismo núcleo y se crea uno nuevo que es la unión de estos. El núcleo del nuevo estado será el núcleo común y como símbolo de anticipación tendremos la unión de todos los símbolos de anticipación de cada uno de los estados del conjunto.
Segundo paso	Para cada estado del autómata modificado, hacemos lo siguiente: ● **Si es un estado no modificado (para cada arista que salga de él).** Si el estado de llegada es un estado modificado, (unión de otros o ha desaparecido) entonces la arista irá al estado unión correspondiente al modificado. ● **Si es un estado modificado.** Todas las aristas que parten de este estado tienen que partir de la unión correspondiente y si el estado al que llegaban estas aristas es modificado entonces llegarán a la unión correspondiente.

Como vemos, se fusionan los núcleos pero las aristas que entraban y salían del resto de los estados hacia los estados modificados tienen que seguir llegando y saliendo de estos.

© Universidad Europea de Madrid. Todos los derechos reservados.

Algoritmo de modificación del autómata. Ejemplo

Veamos un caso práctico en el que partimos del AFD LR(1) que hemos utilizado y buscamos los estados cuyos núcleos sean comunes y por tanto susceptibles de fusionarse:



Al buscar los estados cuyos núcleos sean comunes y por tanto susceptibles de fusionarse, encontraremos que los estados 2 y 6, 3 y 7, 4 y 11 y por último 8 y 13 tienen el núcleo común.

Estados con núcleo común

- Estados 2 y 6 (2_6): S ((. L) y como símbolo de anticipación {**$**} U {,, **)**}, quedando $ |, |).
- Estados 3 y 7 (3_7): S (id. y como símbolo de anticipación {**$**} U {,, **)**}, quedando $ |, |).
- Estados 4 y 11 (4_11): S ((L.) y como símbolo de anticipación {**$**} U {,, **)**}, quedando $ |, |).
- Estados 8 y 13 (8_13): S ((L). y como símbolo de anticipación {**$**} U {,, **)**}, quedando $ |, |).

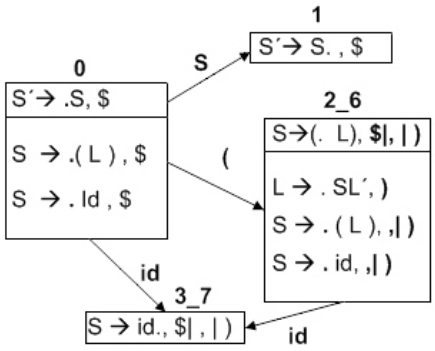
© Universidad Europea de Madrid. Todos los derechos reservados.

Ejemplo AFD LALR

Vamos a realizar un ejemplo en varios pasos a partir del AFD LR(1).

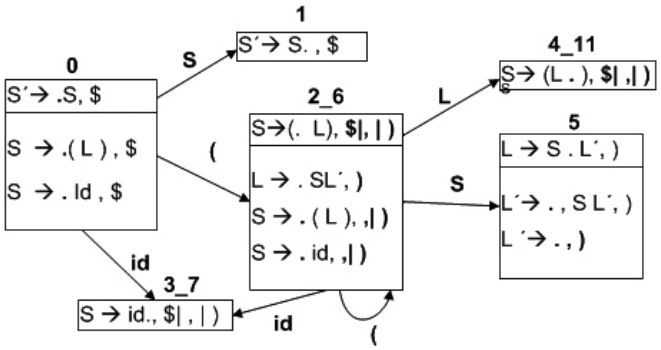
En el primer paso, aparecen dos estados fusionados (el 2_6 y el 3_7) que tienen que cumplir con la regla 2.b de un estado modificado: "todas las aristas que parten de este estado tienen que partir de la unión correspondiente y si el estado al que llegaban estas aristas es modificado entonces llegarán a la unión correspondiente".

En el primer paso, podemos comprobarlo con el estado 3_7, donde la arista con id, iba al estado 7 y ahora llega al estado fusión.



1/4

En el segundo paso incorporamos todos los estados y aristas que se obtienen de realizar la operación ir_a del estado 2_6.

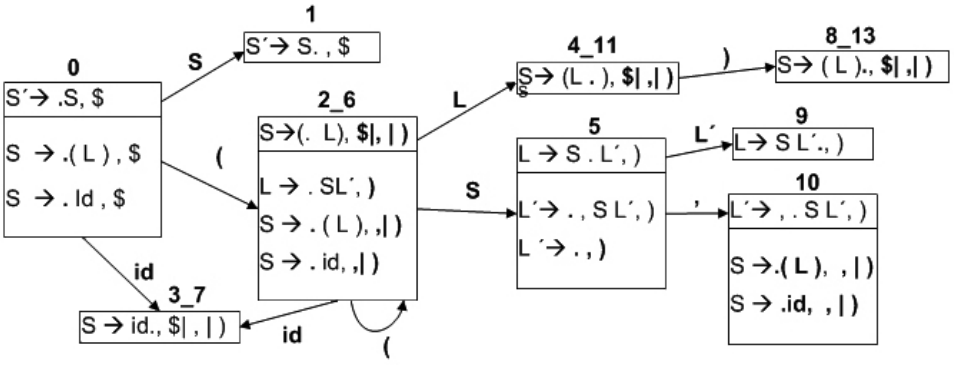


Se incorpora otro estado fusionado, el 4_11, y la arista que iba del estado 2 al estado 6 etiquetada con el "(", ahora se queda en el estado fusionado 2_6.

2/4

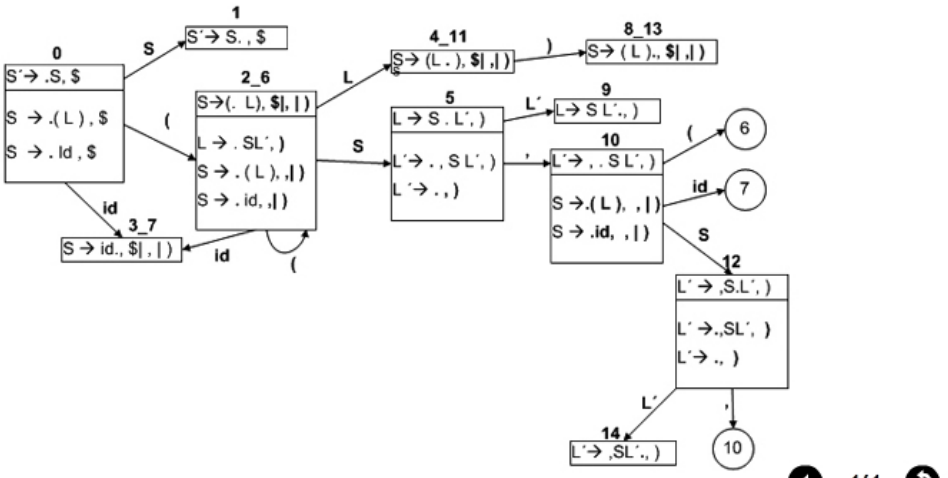
© Universidad Europea de Madrid. Todos los derechos reservados.

En el tercer paso, se realizan las operaciones ir_a del estado fusionado 4_11, obteniéndose otro estado fusionado, el 8_13, con el paréntesis de cierre ")". El resto se queda como estaba en el LR(1).



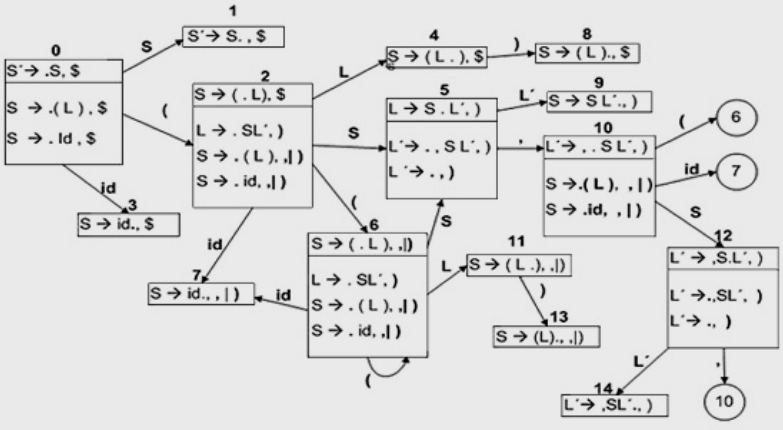
3/4

En el último paso, todo se queda igual que en el AFD LR(1) puesto que no hay más estados fusionados.



4/4

AFD LR(1)



© Universidad Europea de Madrid. Todos los derechos reservados.

Construcción de la tabla de análisis LALR

Esta tabla se construye de la misma manera que la del método LR(1). El algoritmo de construcción consiste en realizar las siguientes acciones **para cada uno de los estados.**

Estado 1: Si el estado S contiene un elemento de la forma [A → α.**T**β, a], donde T es un terminal, se inserta en la matriz ***acción[S, T]*** la acción de desplazar al estado (dS_n) al que va el arco etiquetado con ese terminal T y que será el que contenga como núcleo [A → αT.β, a].

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).

Estado 2: Si el estado S contiene un elemento de la forma [A → α.**N**β, a], 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 → α**N**.β, a].

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.

Estado 3: Si el estado S contiene un **elemento completo** de la forma **[A → α., a],** 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 **a.**

Es decir, si los elementos en **a** son $\mathbf{a_1}$ y $\mathbf{a_2}$ (siendo estos terminales de la gramática o $), las entradas en la matriz serán ***acción[S, a1] = acción[S, a2] = rX.***

Estado 4: Si el elemento completo se refiere a la producción con la que se ha aumentado la gramática, en nuestro ejemplo **S´→ S.** la entrada correspondiente será **acción[1, $] = ACEPTAR.**

Documentos Algoritmo de construcción

Quedando la tabla del AFD LALR, a partir de la gramática indicada a continuación, de la siguiente manera:

© Universidad Europea de Madrid. Todos los derechos reservados.

Gramática:

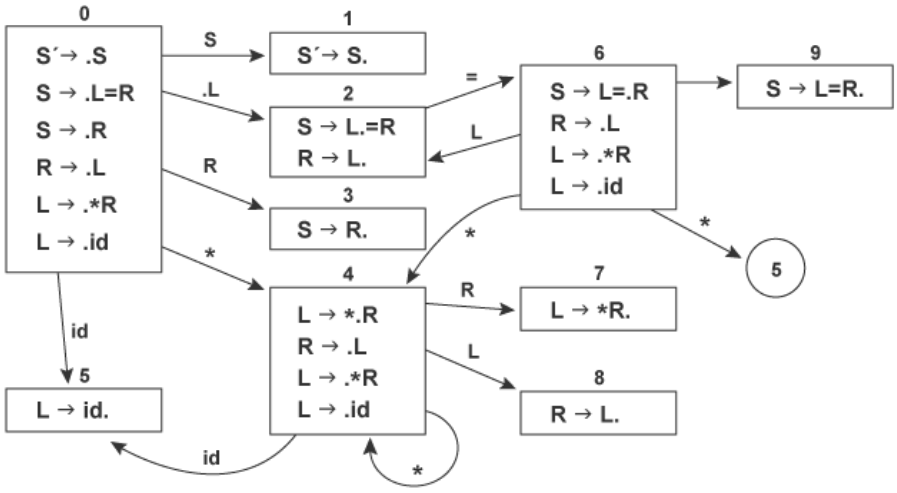
1. S´→ S
2. S → (L)
3. S → id
4. L → S L´
5. L´→ , S L´
6. L´→ λ

Estados	ACCIÓN					IR_A		
	id	(	)	,	$	S	L	L´
0	d3_7	d2_6				1		
1					Aceptar			
2_6	d3_7	d2_6						
3_7			r3	r3	r3			
4_11			d8_13					
5				d10				9
8_13			r2	r2	r2			
9			r4					
10	d3_7	d2_6				12		
12				d10				14
14			r5					

© Universidad Europea de Madrid. Todos los derechos reservados.

Ejemplo completo. Paso 1: SLR

En el siguiente ejemplo, se van a obtener los tres autómatas, SLR, LR(1) y LALR a partir de la gramática (definida en Aho, 1988) que puedes encontrar en este enlace. Comenzamos por el AFD SLR:



Si hiciéramos la tabla SLR correspondiente a este autómata, en el estado 2 observaríamos un conflicto de desplazamiento/reducción que sucede porque al método SLR le falta cierta información de contexto y no funciona bien con determinadas gramáticas aunque estas no sean ambiguas.

Para lo que nos sirve en cualquier caso, es para conocer cuántos estados tendrá el autómata LALR, que serán los mismos que tiene el SLR del que procede.

Conflicto en estado 2

- Apartado de acción: acción [2, =] = d6
- Debido a la producción R → L., donde se indica que tenemos un estado completo, al calcular SIGUIENTE(R) = { =, $}, esto implica que la acción [2, =] r6, es decir reducir por la producción R → L

© Universidad Europea de Madrid. Todos los derechos reservados.

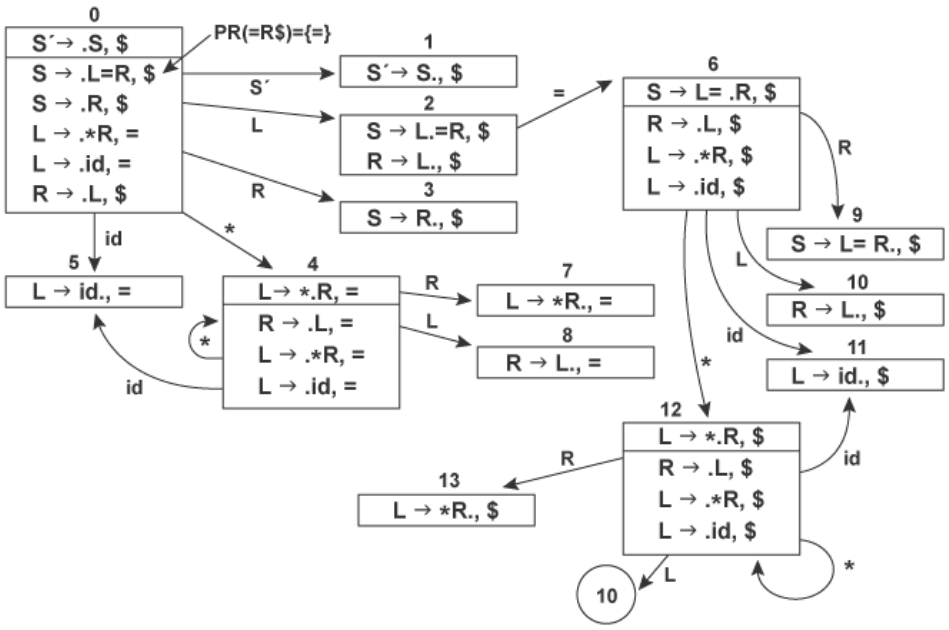
Gramática

1. S → L = R
2. S → R
3. L → *R
4. L → id
5. R → L

© Universidad Europea de Madrid. Todos los derechos reservados.

Ejemplo completo. Paso 2: LR(1)

Como podemos ver, si construimos la tabla de análisis LR(1) con este método ya no hay conflictos, aunque si algunos estados más.



© Universidad Europea de Madrid. Todos los derechos reservados.

Ejemplo completo. Paso 3: LALR

Observando el autómata LR(1), vemos que nos aparecen los siguientes estados comunes:

Estados 4 y 12	Estados 8 y 10
Estados 5 y 11	Estados 7 y 13

Con lo que el AFD LALR queda de la siguiente manera (ver tabla de estados):

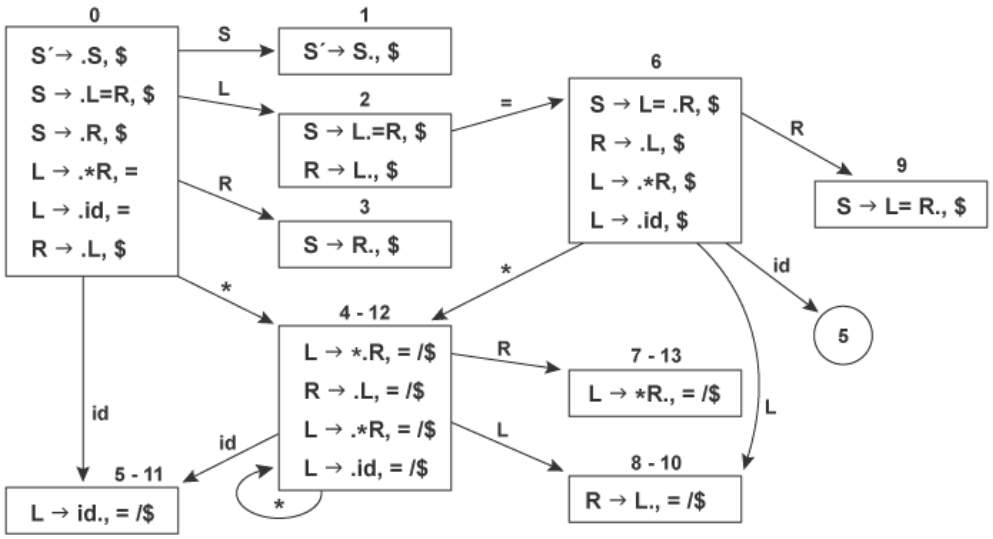


Tabla de estados

Estados	ACCIÓN					IR_A		
	id	(	)	,	$	S	L	L´
0		d4-12			1	2	3	0
1				Aceptar				1
2	d6			r6				2
3				r3				3
4-12		d4-12	d5-11			8-10	7-13	4-12
5-11	r5			r5				5-11
6		d4-12	d5-11			8-10	9	6
7-13	r4			r4				7-13
8-10	r6			r6				8-10
9				r2				9
0		d4-12			1	2	3	0

© Universidad Europea de Madrid. Todos los derechos reservados.

Resumen

En este tema hemos aprendido a realizar un análisis sintáctico LALR, a partir de la fusión de núcleos repetidos en el autómata LR(1) del que procede.

Como hemos demostrado, el número de estados del LALR, se reduce hasta alcanzar el del autómata SLR, lo cual permite mejorar en eficiencia de recursos y en tiempo a la hora de reconocer una sentencia.

Además, hemos podido comprobar que la construcción de la tabla de análisis es idéntica a la del LR(1) del que procede.

Finalmente, y a través de un ejemplo completo, hemos podido ver todos los pasos para llegar desde el autómata SLR al LALR, pasando por el LR(1).

© Universidad Europea de Madrid. Todos los derechos reservados.