Universidad
Europea de Madrid

LAUREATE INTERNATIONAL UNIVERSITIES

ANÁLISIS SINTÁCTICO II

LR1

Índice

Presentación

El objetivo de este tema es entender el funcionamiento del **analizador sintáctico LR(1).** Comenzaremos entendiendo los conceptos básicos del análisis LR(1) y las operaciones necesarias para llevarlo a cabo.

En este tema es importante entender el **algoritmo de análisis LR(1)** y la **construcción de la tabla.**

Los objetivos a conseguir en este tema son:

- Conocer el método de análisis sintáctico ascendente LR(1).
- Dominar las operaciones cerradura e ir_a.
- Aprender el algoritmo de análisis LR(1).
- Aprender a construir los conjuntos de elementos LR(1).
- Conocer cómo se construye la tabla de análisis.

Introducción al analizador LR(1)

El analizador sintáctico ascendente LR(1) es el analizador **más costoso** de implementar, aunque es el que reconoce más gramáticas. El método para construir los conjuntos de elementos es básicamente el mismo que el utilizado en el SLR, aunque incorpora el símbolo de *lookahead* o símbolo de anticipación a los elementos del conjunto.

Por tanto, el elemento LR(1) es un par formado por un elemento LR(0) y el símbolo de anticipación. Este elemento tiene la forma general **[A → α.Bβ, a]**, donde **A → αBβ** es una producción y **a** es un terminal o el $. Precisamente el 1 de LR(1) representa la longitud de a.

Es de destacar que todos los analizadores sintácticos ascendentes reconocen las sentencias de la gramática de la misma forma y **solo cambia la forma de obtener la tabla de análisis,** pero una vez obtenida, el tratamiento de la pila y la entrada para realizar el reconocimiento de una sentencia es el mismo.

De la misma forma que en el SLR, existe el concepto de **prefijo viable** (identificado por el punto) que indica la parte de la frase que se ha reconocido, es decir, la que queda a la izquierda del punto. En la forma general **[A → α .B β, a]** el prefijo viable está representado por α.

Debido a la incorporación del símbolo de anticipación, **cambia la forma de realizar las operaciones de cerradura e ir_a.** En el caso de la operación ir_a, el cambio no es grande, puesto que es la forma en la que se realizan las transiciones de un estado a otro, y conceptualmente no varía mucho.

Símbolo de *lookahead*

El símbolo de *lookahead* tiene varias denominaciones dependiendo de los autores. Otras denominaciones son: símbolo de busqueda anticipada, simbolo de preanálisis o símbolo de anticipación.

Conceptos LR(1)

Para entender cómo funciona el análisis sintáctico LR(1) es necesario tener claros algunos conceptos importantes, partiendo de la forma general **[A → α .B β, a]**, donde B puede ser un terminal o un no terminal y α y β son cadenas de terminales, no terminales o la cadena vacía.

¿Cómo se obtiene a?

Para ello vamos a utilizar la gramática aumentada que ya conocemos. Tenemos dos casos:

1. En el caso del estado 0, y partimos del primer elemento obtenido al aumentar la gramática (G´), es decir S´→ S. En este caso a = $. Por tanto el primer elemento del conjunto 0 será S´→ **.** S, $.
2. El resto de situaciones en las que al hacer la operación cerradura a partir del elemento inicial (o de cualquier otro), **a** se obtiene a partir del conjunto **PRIMERO (βa)** del elemento que origina la operación de cerradura. En este caso tendríamos que hacer la cerradura de S, puesto que tiene un punto en el lado izquierdo y es un no terminal y esto implica que calculamos a para los elementos S → . (L) y S → . id, a partir de S´→ **.**S, $. Para determinar a vamos a hacer una analogía con la forma general:

A→	α	**.**B	β	,	a
S´→		. S		,	$

En este caso tanto α como β de la forma general equivalen a la **cadena vacía** y **a** equivale a **$**, por tanto: PRIMERO (βa) = PRIMERO (λ$) = PRIMERO($) = $.

Es decir, que **a** para la operación cerradura de S será:

S → **.** (L), **$**
S → **.** id, **$**

Importante: cada vez que se haga una operación de cerradura se tiene que calcular **a** para los elementos que se obtengan como consecuencia de la operación.

Gramática aumentada

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

(G´)

G´ representa a la gramática aumentada.

Analogía

Se recomienda hacer esta analogía cada vez que sea necesario calcular **a** como consecuencia de una operación.

PRIMERO (βa)

A modo de recordatorio: Cuando se calcula **PRIMERO (βa)**, si (βa) es un terminal, PRIMERO de un terminal (T) es ese terminal. Si (βa) es un no terminal (N) hay que calcular el conjunto utilizando las reglas definidas en el tema correspondiente.

Operación cerradura e ir_a

Operación cerradura

Operación cerradura (N)**:** consiste en ampliar el conjunto LR(1) que estamos construyendo con todos aquellos elementos que tienen a ese N en la parte izquierda de una producción o regla de la gramática. Estos nuevos elementos se incorporan con un punto situado al inicio de la parte derecha de esta producción. Además debe volver a calcularse el símbolo de anticipación.

Si esta operación a su vez genera más elementos donde el punto queda a la izquierda de un no terminal, se debe seguir realizando la operación cerradura hasta que no se puedan añadir más elementos al conjunto.

Básicamente se realiza igual que en el caso del SLR, pero calculando para cada operación cerradura el símbolo de anticipación.

En detalle Algoritmo de la operación cerradura (Aho et al, 1986)

Operación ir_a

Operación ir_a: esta operación nos ayuda a pasar de un estado a otro del AFD. La operación consiste en avanzar el punto una posición hacia la derecha en todos los elementos de un estado que no sean elementos completos, constituyendo con cada uno de estos elementos los núcleos de los nuevos estados. A partir de estos núcleos se vuelve a realizar la operación cerradura del nuevo estado.

En detalle Algoritmo de la operación ir_a (Aho et al, 1986)

N

N es un no terminal cualquiera de la gramática aumentada.

En detalle

Algoritmo de la operación cerradura (Aho et al, 1986)

```
función Cerradura (I)
begin
        repeat
                for cada elemento [A → α.Bβ, a] en I,
                      cada producción B → γ en G´
                      y cada terminal b en PRIMERO(βa)
                      tal que [B → γ, b] no esté en I
añadir  [B → γ, b] a I;
        until no se puedan añadir mas elementos a I
        return I
end;
```

En detalle

Algoritmo de la operación ir_a (Aho et al, 1986)

```
función ir_a(I, X);
begin
        sea J el conjunto de elementos [A → αX . β, a] tal que
                [A → α . Xβ, a] esté en I;
        return cerradura(J)
End;
```

Algoritmo LR(1)

Una vez que conocemos todas las operaciones a realizar el algoritmo completo LR(1) es el siguiente:

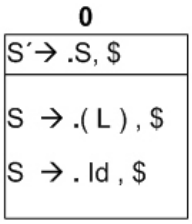
1. Se amplía la gramática (G) con la producción S´ → S, obteniéndose la gramática aumentada (G´).
2. Se crea el núcleo del estado 0 que contiene el elemento LR(1) siguiente: **S´ → .S, $.**
3. Para cada núcleo creado se realiza la **operación cerradura:**
   a. Se obtiene el símbolo de cerradura, donde **[A → α.Bβ, a]** y B pertenece a los no terminales, siendo por tanto este el símbolo de cerradura.
   b. Se introducen en el estado los elementos LR(1) que corresponden a B, que tendrán como elementos aquellas producciones que se derivan del símbolo de cerradura, es decir, **B → .γ** y como símbolo de anticipación aquellos símbolos que pertenezcan al conjunto PRIMERO de la cadena βa, siendo β los símbolos posteriores al símbolo de cerradura y a el símbolo de anticipación del elemento LR(1) **para el que se está realizando la cerradura** obteniéndose **[B → .γ, PRIMERO(βa)]. NOTA:** PRMERO(βa) se hace de la producción que ha originado la operación cerradura, es decir [A → α.Bβ, a].
   c. Para cada elemento LR(1) nuevo se vuelve a (a) hasta que no se puedan añadir más elementos LR(1).
4. Para cada estado, se realiza una partición de forma que **operación ir_a:**
   a. En cada partición, se encontrarán los LR(1) que tengan el mismo símbolo de transición (elementos a la derecha del punto).
   b. Para cada partición se crea un nuevo núcleo formado por los elementos LR(1) de la partición correspondiente con el punto desplazado un símbolo a la derecha:
      a. Si el núcleo creado es nuevo entonces se crea un nuevo estado, se traza una arista desde el estado que origina esta transición hasta el nuevo estado etiquetado con el símbolo de transición y se vuelve al paso 3.
      b. Si el núcleo ya existía se traza una arista desde el estado que origina la transición hasta el estado con dicho núcleo etiquetado con el símbolo de transición.
   c. Así sucesivamente hasta que no se puedan crear más estados.

Ejemplo de construcción de conjuntos de elementos LR(1)

Una vez que sabemos cómo aplicar las operaciones de cerradura e ir_a y conocemos el algoritmo LR(1) vamos a realizar un ejemplo con la gramática aumentada que ya conocemos.

cerradura estado 0

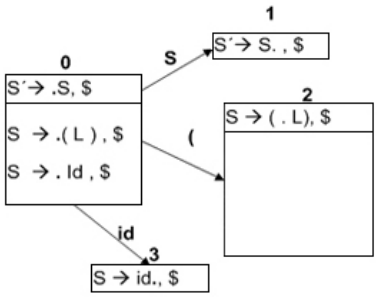
Hacemos la operación cerradura de la primera producción (con la que hemos aumentado la gramática).



1/7

ir_a estado 0

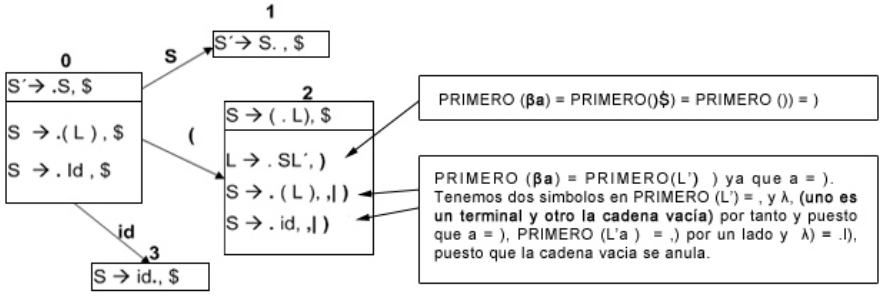
Hacemos la operación ir_a a cada uno de los símbolos que tienen un punto en su lado izquierdo.
Obtenemos dos estados completos, el 1 y el 3, por tanto con estos no avanzamos.



2/7

cerradura estado 2

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

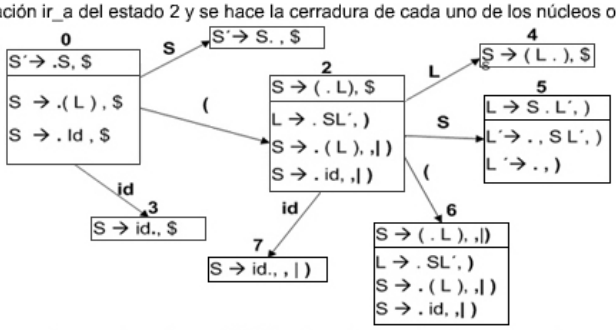


Tanto el estado 1 como el 3 son estados completos, por lo que tenemos que realizar la operación ir_a del estado 2.

3/7

ir_a del estado 2

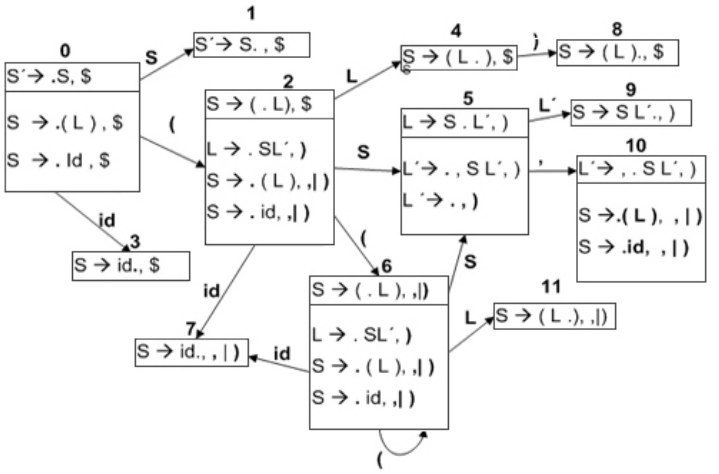
Se realiza la operación ir_a del estado 2 y se hace la cerradura de cada uno de los núcleos obtenidos.



Hemos obtenido un estado completo, el estado 7. En el estado 6, es de destacar que la cerradura de L proporciona el) como símbolo de lookahead a partir de PRIMERO(βa) = PRIMERO( )a) = {) }, mientras que la cerradura de S proporciona , y) a partir de PRIMERO(βa) = PRIMERO(La) = {,a y λa} donde a =), por tanto = coma y paréntesis de cierre = {, y)}.

4/7

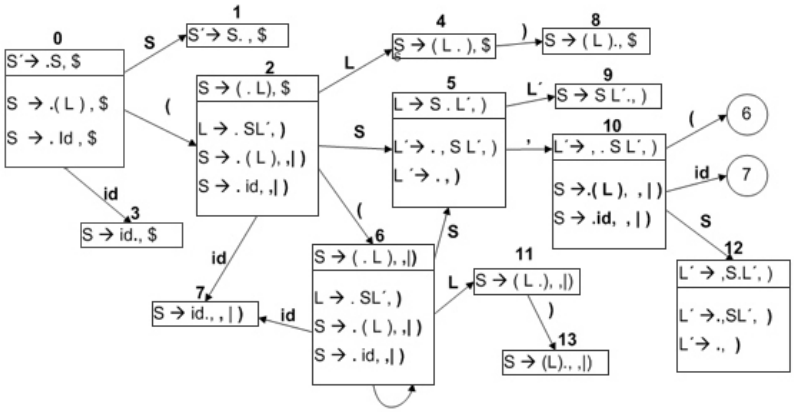
Obtenemos ir_a del estado 4, 5 y 6, así como sus operaciones de cerradura



Hemos obtenido como estados completos el 8 y el 9, faltando por cerrar los estados 10 y 11. En el estado 10 al hacer la cerradura de S, hemos obtenido dos elementos que tienen por símbolo de anticipación el PRIMERO(βa) = PRIMERO(L´a) el cual hemos calculado anteriormente en el estado 6 proporcionando la coma y el paréntesis de cierre.

5/7

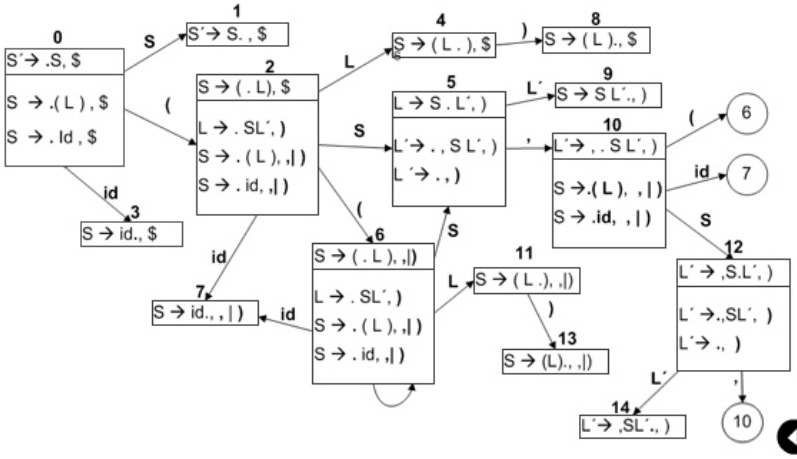
Obtenemos ir_a del estado 10, y 11, así como sus operaciones de cerradura



Hemos obtenido como estados completos el 13, faltando por cerrar el estado 12.

6/7

Obtenemos ir_a del estado 12, así como su operación de cerradura



7/7

Documentos Ejemplo de construcción de conjuntos de elementos LR(1)

Ejemplo con la gramática aumentada que ya conocemos

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

Construcción de la tabla de análisis LR(1)

Una vez hemos obtenido el AFD LR(1) se construye la tabla de análisis sintáctico LR(1), a partir de los estados obtenidos y donde se obtendrán las acciones a realizar para las entradas de la matriz M[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.**

Algoritmo

¿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 LR(1) y por tanto no ambigua.

Si hubiera más de una entrada en la tabla se produciría un conflicto al igual que en el SLR, y 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 → α., a]**, y otra producción del tipo **[A → α.Tβ, a],** donde T es alguno de los símbolos de anticipación **(a_1, a_2,..a_n)** de la producción **[A → α. a].**
Conflicto de reducción/reducción	Se produce cuando en un mismo estado existen dos producciones con el elemento completo, del tipo **[A → α., a]** y **[B → β., a]**, para el mismo símbolo de anticipación (a), no pudiendo por tanto decidirse qué reducción aplicar.

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

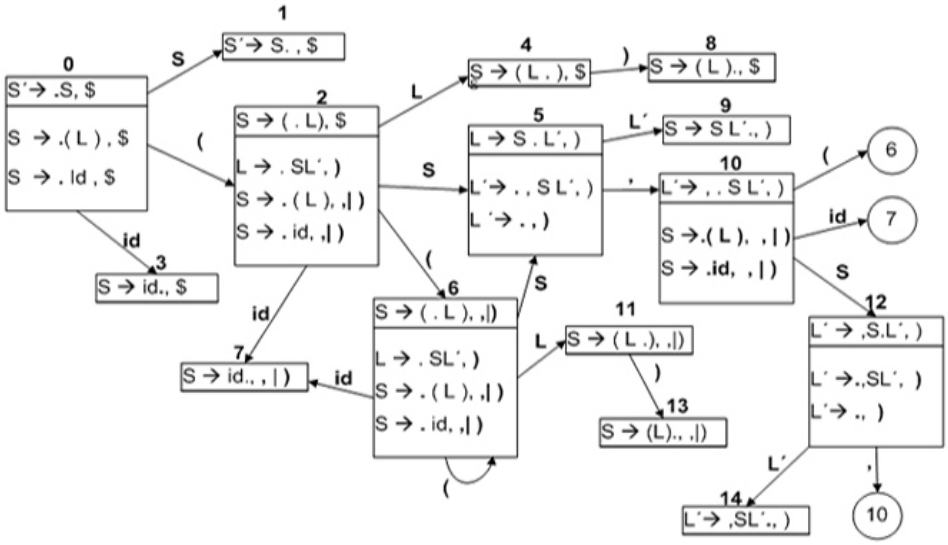
En detalle

Algoritmo

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 → α**T.** 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).
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.
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, a_1] = acción[S, a_2] = rX.***
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.***

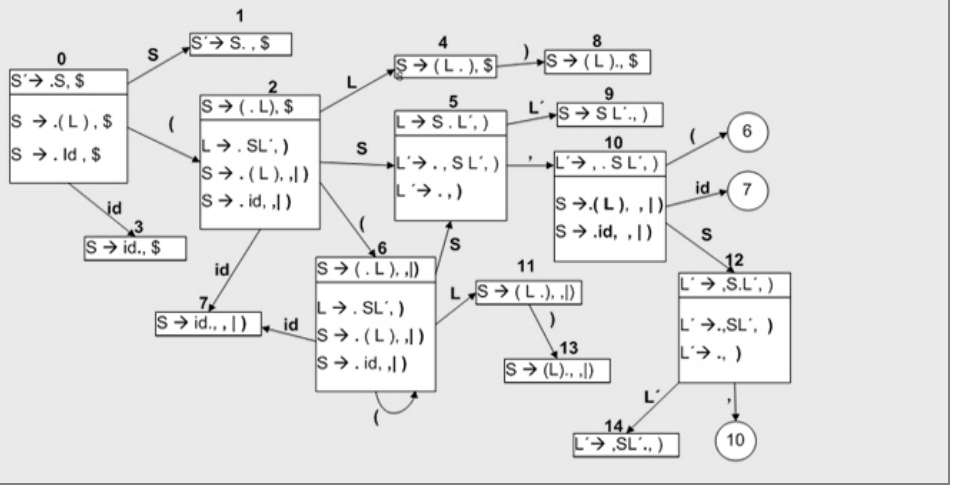
Ejemplo de construcción de tabla de análisis LR(1)

La forma más rápida de construir la tabla es a partir del AFD LR(1) obtenido en el paso 6 anterior, por otro lado la gramática aumentada y numerada, y por otro el algoritmo de construcción de la tabla.



Documentos Estados y tabla final

Tabla es a partir del AFD LR(1)



Gramática aumentada y numerada

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

Algoritmo

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 → α**T.** 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).
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.
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, a_1] = acción[S, a_2] = rX.***
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.***

Resumen

En este tema hemos aprendido a realizar un análisis sintáctico LR(1), a partir de la construcción del autómata que lo genera utilizando los conjuntos de elementos LR(1). Para ello hemos aprendido algunos conceptos como el cálculo de **a,** a partir de la forma general **[A → α.Bβ, a],** así como de las operaciones **cerradura** e **ir_a.** El dominar estas operaciones es crucial para posteriormente aplicar el algoritmo LR(1).

También hemos visto cómo se construye el conjunto de elementos LR(1) en varios pasos, llegando finalmente al autómata, donde siempre es necesario volver a calcular **a** cada vez que se hace la operación cerradura de un no terminal. Esto genera más estados, siendo la diferencia principal entre algunos de ellos los símbolos de anticipación (a) puesto que los núcleos de los estados son los mismos. Esto permite que este método sea capaz de trabajar con gramáticas que desde el punto de vista del SLR eran ambiguas, y por esto mismo se dice que este método tiene mayor potencia de tratamiento de gramáticas aunque genera más estados en el autómata.

Por otro lado hemos visto ejemplos de construcción del conjunto de estados y de la tabla de análisis LR(1), lo que nos permite asimilar estos algoritmos a través de un enfoque práctico.