Universidad
Europea de Madrid

LAUREATE INTERNATIONAL UNIVERSITIES

# ANÁLISIS SINTÁCTICO I

# ANÁLISIS SINTÁCTICO DESCENDENTE LL(1)

# Índice

## Presentación

El objetivo de este tema es entender el funcionamiento del **análisis sintáctico LL(1),** a partir del reconocimiento de una frase y del algoritmo y tabla de análisis. Veremos en primer lugar cómo funciona el autómata a pila de un analizador sintáctico descendente para entender el esquema de funcionamiento, y por otro lado, las condiciones que debe cumplir una gramática para poder ser tratadas con un analizador sintáctico LL(1).

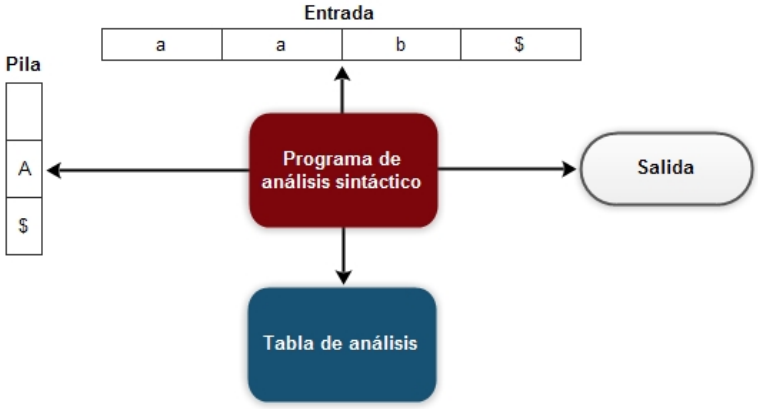
Veremos también varios **ejemplos de comprobación de las condiciones LL(1) y de construcción de la tabla.**

Los objetivos a conseguir en este tema son:

- Comprender el funcionamiento básico de un analizador LL(1).
- Entender cómo se reconoce una sentencia.
- Cómo deben ser las gramáticas LL(1) y qué condiciones deben cumplir.
- Construir la tabla de análisis LL(1).

## Funcionamiento básico del analizador LL(1)

El analizador sintáctico descendente LL(1) utiliza una **pila** para hacer el análisis sintáctico, basándose en la **arquitectura del autómata a pila.** Además, utiliza una **tabla de análisis** que indica si la entrada es correcta y qué acción realizar con ella. El esquema básico de funcionamiento se representa por la siguiente figura (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 $ en el inicio de la pila. En la entrada nos encontramos también $ al final de la cadena de entrada, de tal forma que si no ha habido errores y, tanto en la entrada como en la pila solo queda el $, quiere decir que **la frase ha sido reconocida** como perteneciente a la gramática y termina el análisis con éxito. El programa de análisis busca en la tabla de análisis qué producción aplicar a partir de los símbolos que hay en la pila y en la entrada.

Veremos un ejemplo de funcionamiento a partir de una tabla ya construida en la siguiente pantalla.

## Reconocimiento de una sentencia

Para entender cómo funciona el análisis sintáctico LL(1) utilizando el autómata a pila y la tabla de análisis, partiremos de una sentencia a reconocer y la tabla de análisis (posteriormente veremos cómo se construye esta tabla y qué condiciones debe cumplir la gramática).

La **gramática** será:

| | |
|---|---|
| **S → (L) \| id**<br>**L → S L´**<br>**L´→ , S L´\| α** | Sentencia a reconocer: (a, b) |

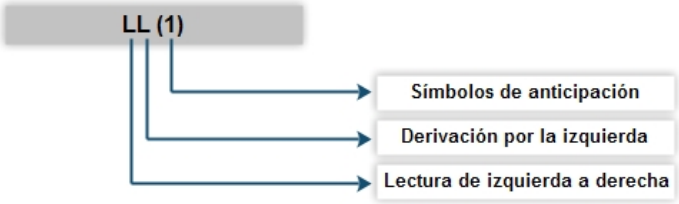
Partimos de la siguiente **tabla de análisis,** que es una matriz (M) de dos dimensiones:

| No Terminales | SIMBOLOS DE ENTRADA | | | | |
|---|---|---|---|---|---|
| | ( | ) | id | , | $ |
| S | S→ (L) | | S → id | | |
| L | L→ SL´ | | L → SL´ | | |
| L´ | | L´→ λ | | L´→ ,SL´ | |

Documentos Proceso de reconocimiento de la sentencia

## Gramáticas LL(1)

Recordemos que las gramáticas LL(1) forman parte de los analizadores sintácticos descendentes predictivos (distintos a los recursivos). Los significados de las siglas son:




Por tanto, este tipo de gramáticas recorren el árbol sintáctico de izquierda a derecha, y en ese recorrido selecciona las derivaciones más a la izquierda y son capaces de reconocer si la sentencia pertenece a la gramática con solo ver un (1) símbolo por anticipado.

Las gramáticas independientes del contexto (GIC), para poder analizarse con un analizador sintáctico descendente predictivo, LL(1) necesitan cumplir una serie de condiciones, puesto que no todas las GIC son gramáticas LL(1). Es decir, **las gramáticas LL(1) son un subconjunto de las GIC** y las condiciones son las siguientes:

1. **No puede ser ambigua.** Es decir, no puede tener producciones con varias alternativas que comiencen por lo mismos símbolos y si las hubiese habría que factorizar.
2. **No puede ser recursiva por la izquierda.**
3. **Cuando haya producciones del tipo A → α | β se tienen que cumplir las siguientes condiciones:**

   En detalle Condiciones

Si la gramática elegida cumple estas condiciones de no ambigüedad y además las que se indican en el apartado 3, podemos afirmar que la gramática es LL(1). Como podemos observar, es un subconjunto relativamente pequeño de las GIC no ambiguas.

**Recorren el árbol sintáctico de izquierda a derecha**

La primera "L" viene del inglés, **Left** to right.

**Selecciona las derivaciones más a la izquierda**

La segunda L viene del inglés **Leftmost.**

**En detalle**

**Condiciones**

1. No puede haber conflictos entre los conjuntos PRIMERO de estas alternativas, es decir **no puede haber conflictos PRIMERO/PRIMERO.**
2. **A lo sumo de una de las dos alternativas, α o β, se deriva la cadena vacía (λ), nunca de las dos.**
3. Si de β se deriva λ, entonces de α no se deriva ninguna cadena que comience con un terminal en SIGUIENTE(A). Es decir, **no puede haber conflictos PRIMERO/SIGUIENTE.**

## Ejemplo 1 de condiciones LL(1)

A partir de la gramática siguiente, vamos a verificar las condiciones de la misma para saber si es LL(1).

S → ( L ) | id  L → L , S | S

### 1.- ¿ Es ambigua la gramática?

No, puesto que no hay varias alternativas que comiencen igual.

### 2.- ¿Es la gramática recursiva por la izquierda?

Sí, debido a la producción L → L , S. Se aplica la regla ya conocida, quedando la gramática como sigue:

S → (L) | id  L → S L´  L´→ , S L´| λ

### 3.- ¿Hay producciones del tipo A → α | β?

Si las hay, es necesario calcular los conjuntos PRIMERO y SIGUIENTE, para comprobar las siguientes tres condiciones:

| | PRIMERO | SIGUIENTE |
|---|---|---|
| S | (, id | $, , , ) |
| L | (, id | ) |
| L´ | , , **λ** | ) |

- Conflictos PRIMERO/PRIMERO.
- Producción donde se derive de ambas alternativas la cadena vacía.
- Conflictos PRIMERO/SIGUIENTE.

A la vista del resultado que ha ofrecido el análisis de las condiciones, podemos decir que **la gramática elegida es LL(1).**

**Regla ya conocida**

A → Aα | β, se resuelve ⇒ A → β A´

A´ → α A´| λ

**Conflictos PRIMERO/PRIMERO**

**¿Hay conflictos PRIMERO/PRIMERO en estas producciones?**

Analizamos las dos producciones con varias alternativas:

- S → (L) | id: No tiene conflictos PRIMERO/PRIMERO, puesto que empiezan ambas por terminales "(" e "id" y son diferentes.
- L´→ , S L´| λ: No tiene conflictos PRIMERO/PRIMERO, puesto que "," es distinto de la cadena vacía, λ.

**Producción donde se derive de ambas alternativas la cadena vacía**

**¿Hay alguna producción donde se derive de ambas alternativas la cadena vacía?**

No, puesto que no aparece λ en los dos conjuntos PRIMERO de la producción L´→ , S L´| λ.

**Conflictos PRIMERO/SIGUIENTE**

**¿Hay conflictos PRIMERO/SIGUIENTE en las producciones en las que de β se derive λ?**

En la producción L´→ , S L´| λ: β = **λ** y α = **, S L´**

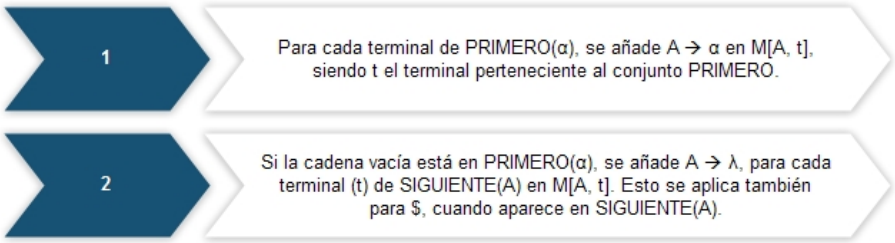
Analizamos PRIMERO(α), es decir PRIMERO(, S L´) = { **,** }, mientras que SIGUIENTE(A) = SIGUIENTE(L´) = { **)** }, por tanto no coinciden.

## Construcción de la tabla de análisis LL(1)

La tabla de análisis sintáctico LL(1) es una matriz de dos dimensiones indexada por no terminales (N, columna de la izquierda en la tabla que hemos utilizado) y terminales (T, fila superior de la tabla, que incluye el símbolo $). Por tanto accedemos a la tabla utilizando M[N, T]. En la tabla las entradas vacías son errores, que posteriormente cuando veamos la fase de tratamiento de errores se puede rellenar con llamadas a funciones o procedimientos que avisan del error.

### ¿Cómo se construye esta tabla?

A partir de cada producción simbolizada por A → α, siendo α cualquier cadena de terminales y no terminales, nos recorremos todas la producciones de la gramática y hacemos lo siguiente:




Documentos Ejemplo de contrucción de la tabla análisis LL(1)

## Ejemplo 2 de condiciones LL(1)

Vamos a hacer otro ejemplo completo, en este caso con una **gramática que no cumple las condiciones LL(1)** para ver los efectos que esto produce en la construcción de la tabla. La gramática es la siguiente:

S → i C t S | i C t S e S | a  C → b

Comenzamos a comprobar las condiciones LL(1)

### 1.- ¿ Es ambigua la gramática?

Sí, puesto que no hay varias alternativas que comiencen igual. Lo corregimos, factorizando la gramática:

S → i C t S S´ |a  S´→ e S | λ  C → b

### 2.- ¿Es la gramática recursiva por la izquierda?

No, puesto que no hay producciones que comiencen en el lado izquierdo por el mismo símbolo del lado derecho.

### 3.- ¿Hay producciones del tipo A → α | β?

Si las hay, es necesario calcular los conjuntos PRIMERO y SIGUIENTE, para comprobar las siguientes tres:

| | PRIMERO | SIGUIENTE |
|---|---|---|
| S | i, a | $, e |
| S´ | e, λ | $, e |
| C | b | t |

- Conflictos PRIMERO/PRIMERO.
- Producción donde se derive de ambas alternativas la cadena vacía.
- Conflictos PRIMERO/SIGUIENTE.

**Conflictos PRIMERO/PRIMERO**

**¿Hay conflictos PRIMERO/PRIMERO en estas producciones?**

Analizamos las dos producciones con varias alternativas:

- S → i C t S S´ |a: No tiene conflictos PRIMERO/PRIMERO, puesto que empiezan ambas por terminales "("i" y "a") y estos son diferentes.
- S´→ e S | λ: No tiene conflictos PRIMERO/PRIMERO, puesto que "e" es distinto de la cadena vacía, λ.

**Producción donde se derive de ambas alternativas la cadena vacía**

**¿Hay alguna producción donde se derive de ambas alternativas la cadena vacía?**

No, puesto que no aparece λ en los dos conjuntos PRIMERO de las alternativas de la producción S´→ e S | λ.

**Conflictos PRIMERO/SIGUIENTE**

**¿Hay conflictos PRIMERO/SIGUIENTE en las producciones en las que de β se derive λ?**

En la producción S´→ e S | λ: β = λ y α = **eS**

Analizamos PRIMERO(α), es decir PRIMERO(eS) = { e }, mientras que SIGUIENTE(A) = SIGUIENTE(S´) = { $, e }, por tanto e está en ambos ⇒ **no cumple la condición ya que hay conflicto PRIMERO/SIGUIENTE.**

## Ejemplo 2 de construcción de tabla LL(1)

A partir de la gramática anterior veremos las consecuencias que produce el que una gramática no cumpla las condiciones y por tanto no sea. una gramática LL(1). Partimos de la gramática a la que se han eliminado las ambigüedades:

S → i C t S S´ |a  S´→ e S | λ  C → b

De acuerdo con el algoritmo de creación de la tabla vamos analizando una a una las producciones de la gramática para construir la tabla, hasta que aparezca el problema que produce el conflicto detectado en la producción S´→ e S | λ.

En detalle Empezamos por S → i C t S S´ En detalle Para S → a En detalle Para S´ → e S En detalle Para S´→ λ

**En detalle**

**Empezamos por S → i C t S S´**

| No Terminales | SIMBOLOS DE ENTRADA | | | | | |
|---|---|---|---|---|---|---|
| | i | t | a | e | b | $ |
| S | S→iCtSS´ | | | | | |
| S´ | | | | | | |
| C | | | | | | |

El conjunto PRIMERO(α) equivale a PRIMERO(iCtSS) ={ **i** }, luego la entrada equivalente para M[A, t], será M[S, i] = S → i C t S S´.

**En detalle**

**Para S → a**

| No Terminales | SIMBOLOS DE ENTRADA | | | | | |
|---|---|---|---|---|---|---|
| | i | t | a | e | b | $ |
| S | S→iCtSS´ | | S→a | | | |
| S´ | | | | | | |
| C | | | | | | |

El conjunto PRIMERO(α) equivale a PRIMERO(a) ={ **a** }, luego la entrada equivalente para M[A, t], será M[S,a] = S → a.

**En detalle**

**Para S´ → e S**

| No Terminales | SIMBOLOS DE ENTRADA | | | | | |
|---|---|---|---|---|---|---|
| | i | t | a | e | b | $ |
| S | S→iCtSS´ | | S→a | | | |
| S´ | | | | S´→eS | | |
| C | | | | | | |

El conjunto PRIMERO(α) equivale a PRIMERO(eS) ={ **e** }, luego la entrada equivalente para M[A, t], será M[S´,e] = S´→ eS.

**En detalle**

**Para S´→ λ**

| No Terminales | SIMBOLOS DE ENTRADA | | | | | |
|---|---|---|---|---|---|---|
| | i | t | a | e | b | $ |
| S | S→iCtSS´ | | S→a | | | |
| S´ | | | | S´→eS<br>S´→λ | | S´→λ |
| C | | | | | C→b | |

El conjunto PRIMERO(α) equivale a PRIMERO(λ) ={ **λ** }, estamos en la regla 2 y hay que ir a SIGUIENTE(S´)= {$, e}, por tanto M[A, t], será M[S´,e|$] = S´→ λ

## Resumen

En este tema hemos visto las condiciones que debe cumplir una gramática para poder ser reconocida una sentencia por un analizador sintáctico descendente LL(1) y consiste en lo siguiente:

- No puede ser ambigua.
- No puede ser recursiva por la izquierda.
- Cuando haya producciones del tipo A → α | β se tienen que cumplir las siguientes condiciones:
  - No puede haber conflictos PRIMERO/PRIMERO.
  - A lo sumo de una de las dos alternativas, α ó β, se deriva la cadena vacía (λ), nunca de las dos.
  - No puede haber conflictos PRIMERO/SIGUIENTE.

Si estas condiciones se cumplen se puede construir la tabla de análisis LL(1) con la seguridad de que no habrá conflictos, es decir no habrá cuadrículas con mas de una producción.

¿Cómo se construye la tabla de análisis LL(1)?

1. Para cada terminal de PRIMERO(α), se añade A → α en M[A, t], siendo t el terminal perteneciente al conjunto PRIMERO.
2. Si la cadena vacía está en PRIMERO(α), se añade A → λ, para cada terminal (t) de SIGUIENTE(A) en M[A, t]. Esto se aplica también para $, cuando aparece en SIGUIENTE(A).

Para acabar de asimilarlo se han visto dos ejemplos completos, donde se ha visto toda la casuística posible.