Universidad Nacional de Cajamarca
"Norte de la Universidad Peruana"

UNIDAD III:
ANALISIS SINTÁCTICO

ING. SANDRA RODRIGUEZ AVILA

CONTENIDO

- ANALISIS SINTÁCTICO o PARSER
- FUNCIONES DEL ANALIZADOR SINTÁCTICO
- DERIVACIONES MÁS A LA IZQUIERDA Y MÁS A LA DERECHA
- TIPOS DE ANALIZADORES SINTÁCTICOS
- ANÁLISIS SINTÁCTICO DESCENDENTE: EL PROBLEMA DEL RETROCESO
- ALGORITMO DE A. SINTÁCTICO DESCENDENTE CON RETROCESO
- ALGORITMO DE A. SINTÁCTICO ASCENDENTE CON RETROCESO

ANALISIS SINTÁCTICO O PARSER

- El análisis sintáctico (parser en lengua inglesa) toma los tokens que le envía el analizador léxico y comprueba si con ellos se puede formar alguna sentencia válida del lenguaje.

- Se entiende por sintaxis como el conjunto de reglas formales que especifican como se construyen las sentencias de un determinado lenguaje.

- La sintaxis de los lenguajes de programación habitualmente se describe mediante gramáticas libres de contexto (o gramáticas tipo 2), G = (N, T, P, S); utilizando para ello algún tipo de notación como por ejemplo, las notaciones BNF y EBNF.

FUNCIONES DEL ANALIZADOR SINTÁCTICO

- Comprobar si la cadena de tokens proporcionada por el analizador léxico puede ser generada por la gramática que define el lenguaje fuente (gramática libre del contexto, GLC).
- Construir el árbol de análisis sintáctico que define la estructura jerárquica de un programa y obtener la serie de derivaciones para generar la cadena de tokens. El árbol sintáctico se utilizará como representación intermedia en la generación de código.
- Informar de los errores sintácticos de forma precisa y significativa. Deberá estar dotado de un mecanismo de recuperación de errores para continuar con el análisis.

FUNCIONES DEL ANALIZADOR SINTÁCTICO

- **Analizar sintácticamente** una cadena de *tokens* es encontrar para ella un **árbol sintáctico** o derivación, que tenga como raíz el símbolo inicial de la gramática libre de contexto y mediante la aplicación sucesiva de sus reglas de derivación se pueda alcanzar dicha cadena como hojas del árbol sintáctico. En caso de éxito la sentencia **pertenece al lenguaje** generado por la gramática, y puede proseguirse el proceso de compilación. En caso contrario, es decir cuando no se encuentra el árbol que genera dicha gramática, se dice entonces que la sentencia **no pertenece al lenguaje**, y el compilador emite un mensaje de error, pero el proceso de compilación trata de continuar. Algunos compiladores paran el proceso de compilación cada vez que encuentran un error, en ellos es necesario realizar tantas compilaciones como errores tendría el programa fuente.

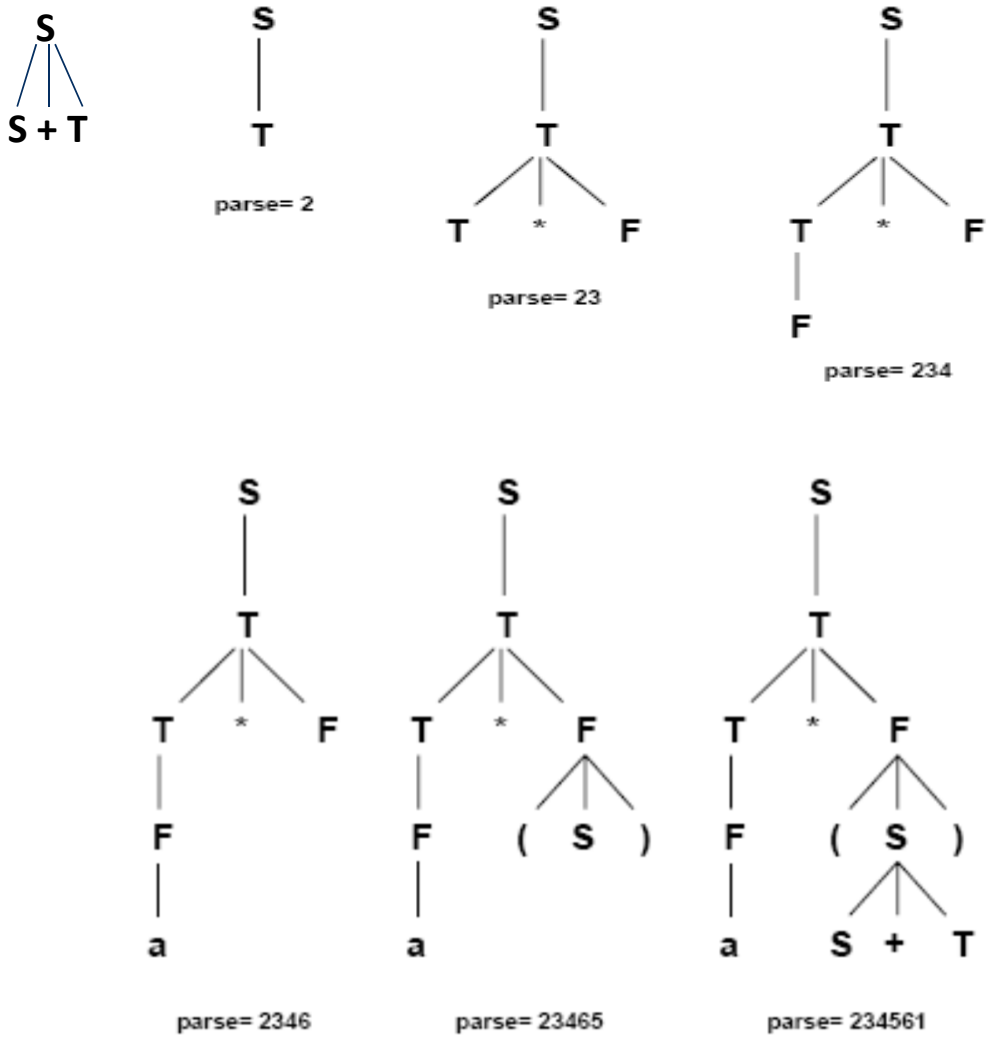
DERIVACIONES MÁS A LA IZQUIERDA Y MÁS A LA DERECHA

Ejemplo

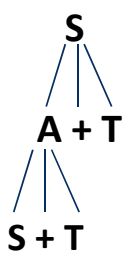
- Sea la gramática G = (N, T, P,S) para al generación de expresiones aritméticas: N = {S, T, F}, T = {a, b, +, *, (,)} y las reglas de producción P son:
  (1) S → S + T
  (2) S → T
  (3) T → T* F
  (4) T → F
  (5) F → (S)
  (6) F → a
  (7) F → b

- Determinar los árboles sintácticos más a la izquierda y más a la derecha que reconocen la cadena a*(a+b).

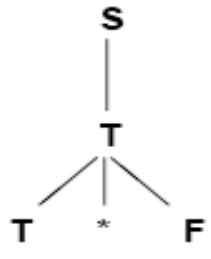
ARBOL SINTÁCTICO MAS A LA IZQUIERDA



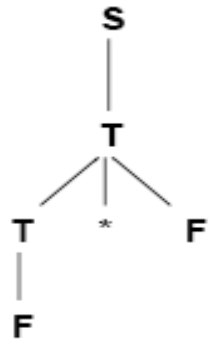
ARBOL SINTÁCTICO MAS A LA IZQUIERDA



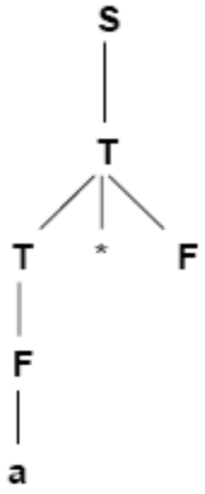
parse= 2



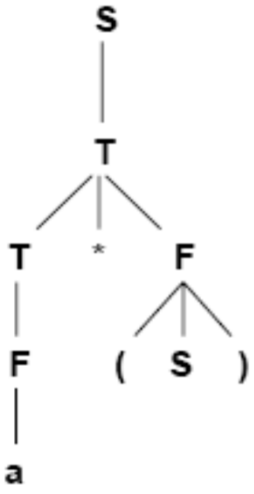
parse= 23



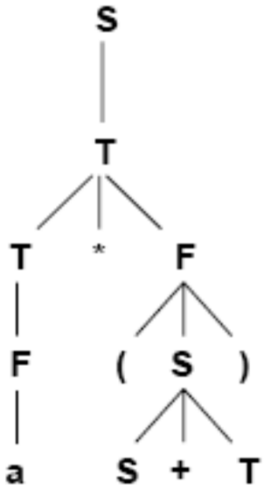
parse= 234



parse= 2346

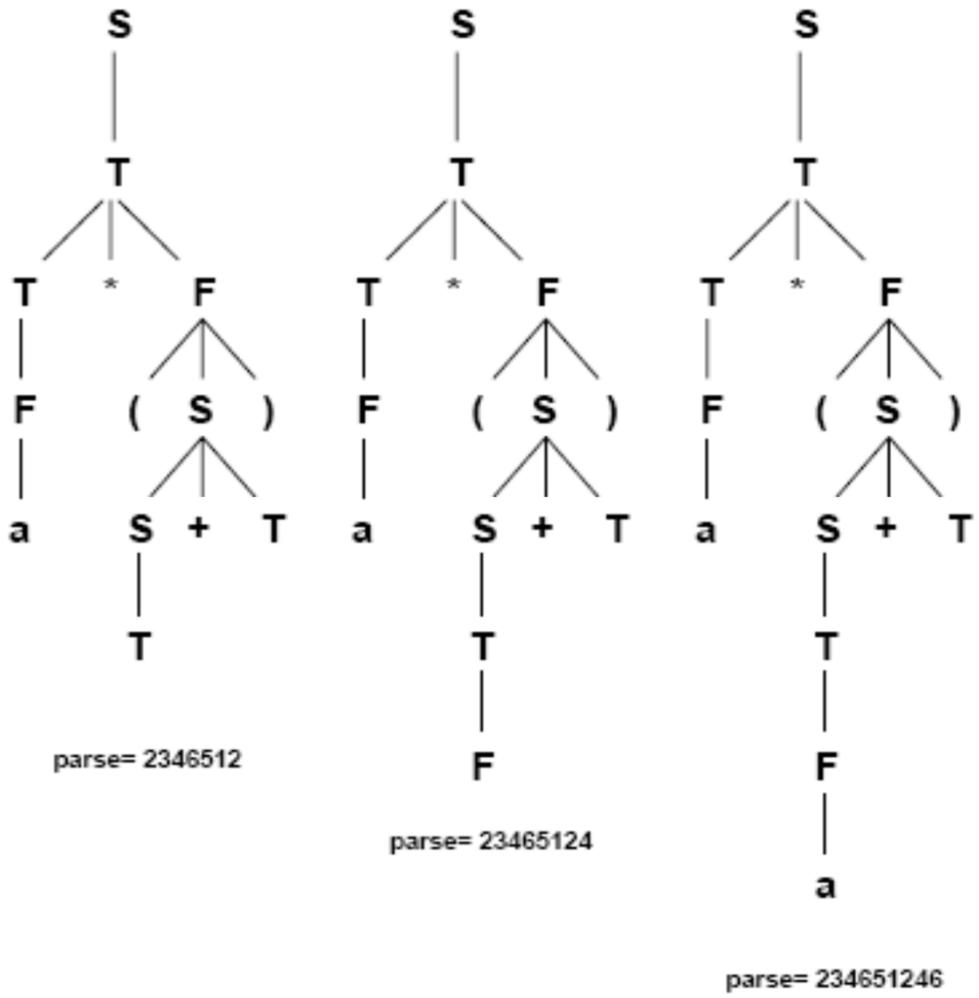


parse= 23465

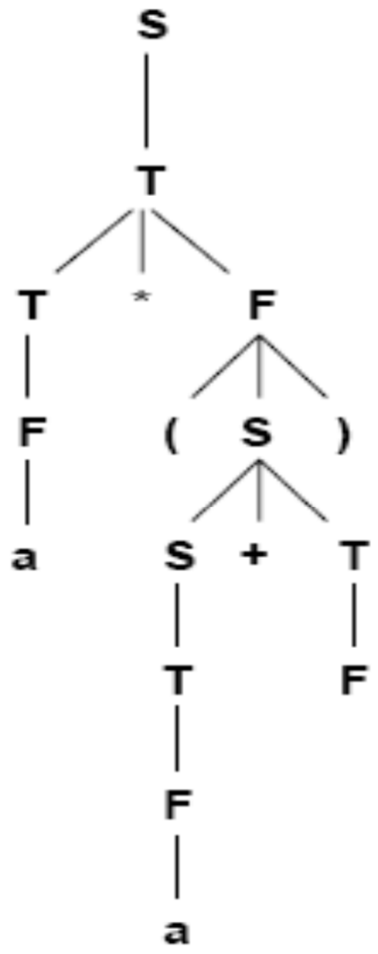


parse= 234561

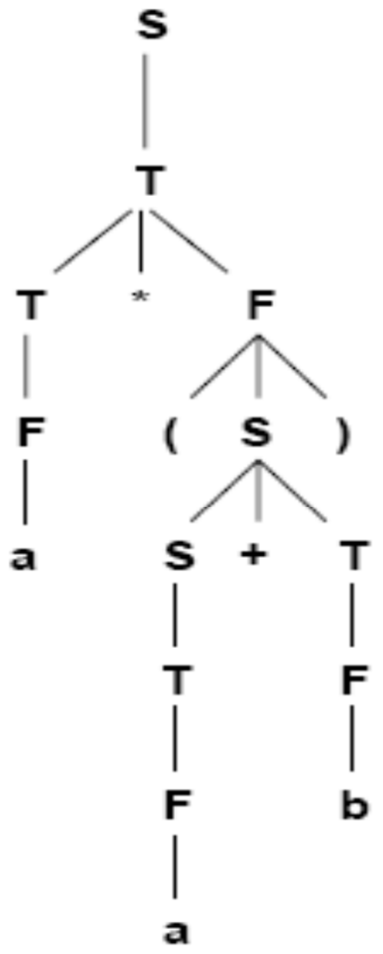
ARBOL SINTÁCTICO MAS A LA IZQUIERDA



ARBOL SINTÁCTICO MAS A LA IZQUIERDA

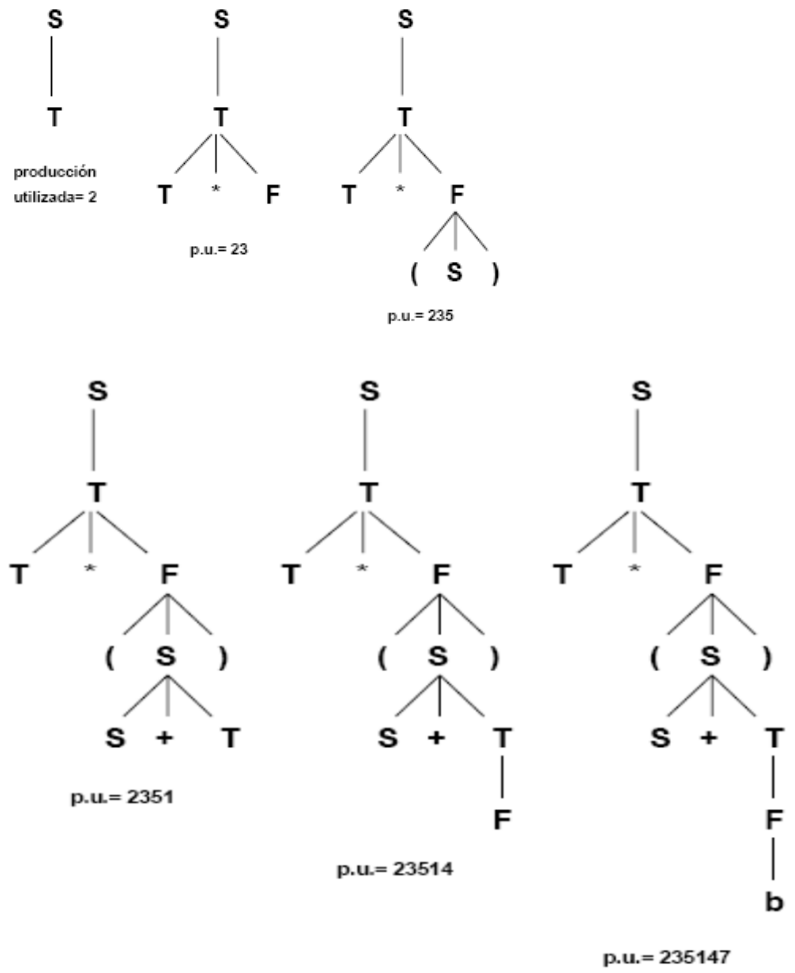


parse= 2346512464

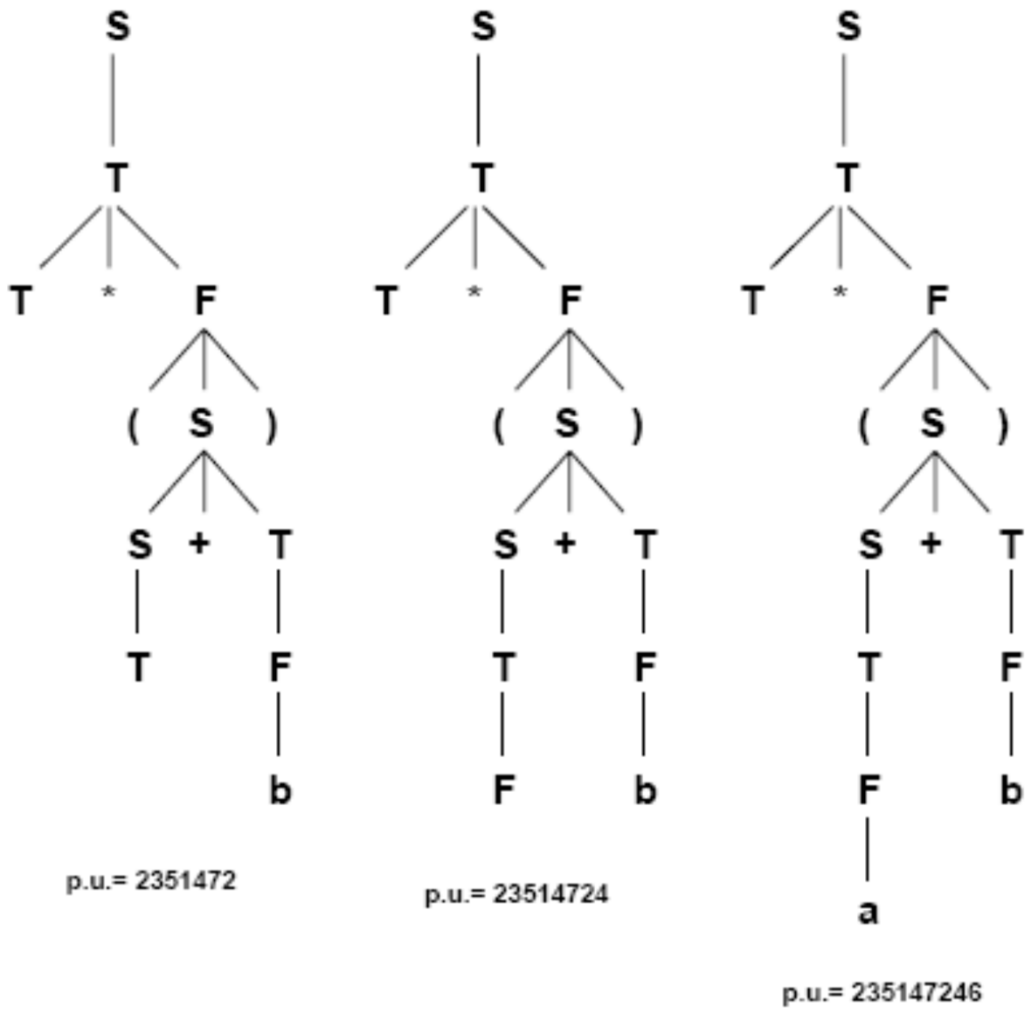


parse Izdo= 23465124647

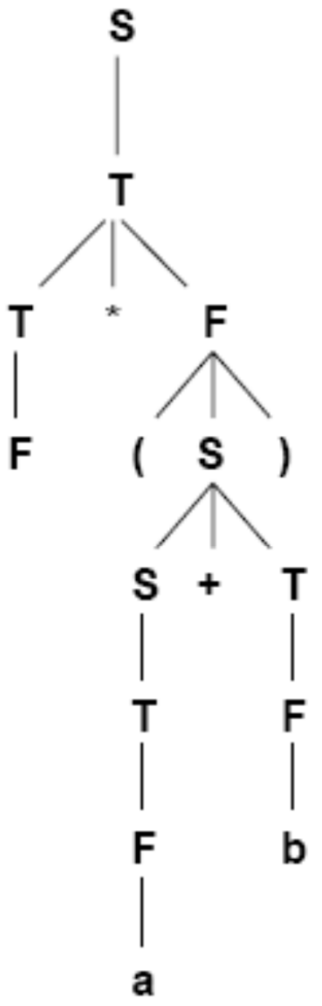
ARBOL SINTÁCTICO MAS A LA DERECHA



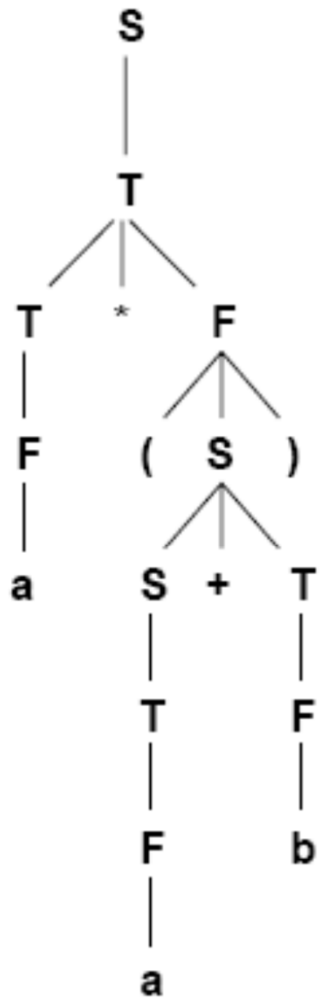
ARBOL SINTÁCTICO MAS A LA DERECHA



ARBOL SINTÁCTICO MAS A LA DERECHA



p.u.= 2351472464



p.u.= 23514724646

parse derecho: 64642741532

TIPOS DE ANALIZADORES SINTÁCTICOS

1. **Analizadores Descendentes (*top-down*)**

- Construyen el árbol sintáctico de la sentencia a reconocer desde el símbolo inicial (raíz), hasta llegar a los símbolos terminales (hojas) que forman la sentencia, usando derivaciones más a la izquierda.

- Los principales problemas que se plantean son dos: el retroceso (backtracking) y la recursividad a izquierdas. Para solventar este inconveniente, se opera con gramáticas LL(k), que pueden realizar un análisis sin retroceso en forma descendente.

TIPOS DE ANALIZADORES SINTÁCTICOS

2. Analizadores Ascendentes (*bottom-up*)

- Construyen el árbol sintáctico de la sentencia a reconocer desde las hojas hasta llegar a la raíz. Son analizadores del tipo reducción-desplazamiento (shift-reduce), parten de los distintos tokens de la sentencia a analizar y por medio de reducciones llegan al símbolo inicial de la gramática.
- El principal problema que se plantea en el análisis ascendente es el retroceso. Para solventar este inconveniente, se definieron distintos tipos de gramáticas entre las cuales las más utilizadas son las LR(k) ya que realizan eficientemente el análisis ascendente sin retroceso.

ANÁLISIS SINTÁCTICO DESCENDENTE: EL PROBLEMA DEL RETROCESO

- El primer problema que se presenta con el análisis sintáctico descendente, es que a partir del nodo raíz, el analizador sintáctico no elija las producciones adecuadas para alcanzar la sentencia a reconocer. Cuando el analizador se da cuenta de que se ha equivocado de producción, se tienen que deshacer las producciones aplicadas hasta encontrar otras producciones alternativas, volviendo a tener que reconstruir parte del árbol sintáctico. A este fenómeno se le denomina *retroceso, vuelta atrás* o en inglés *backtracking.*
- El proceso de retroceso puede afectar a **otros módulos** del compilador tales como tabla de símbolos, código generado, etc. teniendo que deshacerse también los procesos desarrollados en estos módulos.

ALGORITMO DE A. SINTÁCTICO DESCENDENTE CON RETROCESO

1. Se colocan las reglas de la gramática según un orden preestablecido para cada uno de los no terminales de la que está compuesta.

2. Se comienza la construcción del árbol sintáctico a partir del símbolo inicial, y aplicando las siguientes reglas en forma recursiva. Al nodo en proceso de expansión en un determinado momento se le llamará **nodo activo**.

ALGORITMO DE A. SINTÁCTICO DESCENDENTE CON RETROCESO

3. Cada nodo activo escoge la primera de sus alternativas posibles, por ejemplo, para el *no terminal* A con la regla A → x1x2...xn crea *n* descendientes directos, y el nodo activo en ese momento pasa a *ser el primer descendiente por la izquierd*a. Cuando se deriven todos los descendientes, pasará a ser nodo activo el más inmediato derecho de A susceptible de ser derivado. En el caso de que la regla fuese: A → x1 | x2 |...| xn se elegirá en un principio la alternativa de más a la izquierda.

ALGORITMO DE A. SINTÁCTICO DESCENDENTE CON RETROCESO

4. Si el nodo activo es un **terminal** deberá entonces compararse el símbolo actual de la cadena a analizar con dicho nodo. *Si son iguales* se avanza un token de entrada y el nuevo símbolo actual será el situado más a la derecha del terminal analizado. Si *no son iguales* se **retrocede** hasta un nodo *no terminal* y se reintenta eligiendo la **siguiente alternativa**. Si aún así no existe ninguna alternativa posible se retrocede al no terminal anterior, y así sucesivamente. Si se llega al símbolo inicial la cadena no pertenece al lenguaje.

ALGORITMO DE A. SINTÁCTICO ASCENDENTE CON RETROCESO

Ejemplo de análisis ascendente con retroceso

Dada la gramática,

S → a B c D e [1]

B → B b [2]

B → b [3]

D → d [4]

comprobar si la cadena *abbcde* pertenece al lenguaje que describe.

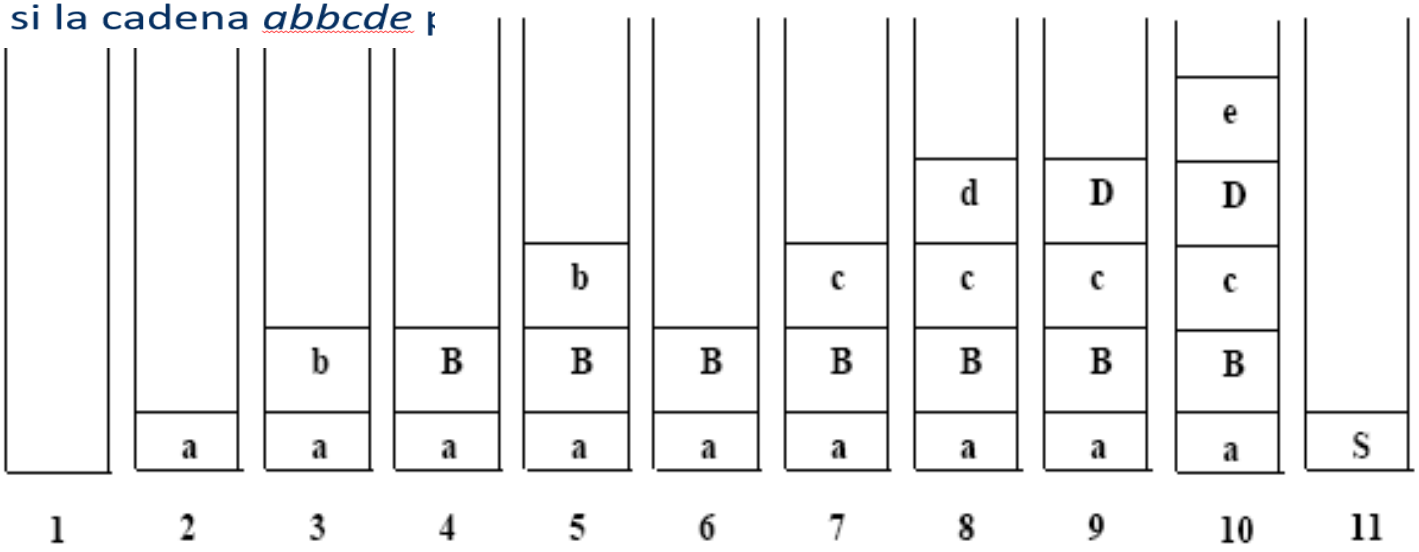
- Los contenidos de la pila en las sucesivas etapas del análisis serán:

ALGORITMO DE A. SINTÁCTICO ASCENDENTE CESO

Dada la gramática,

S → a B c D e [1]
B → B b [2]
B → b [3]
D → d [4]

comprobar si la cadena *abbcde* p



ALGORITMO DE A. SINTÁCTICO ASCENDENTE CON RETROCESO

1. Inicialmente la pila está vacía.

2. Se lee el token a y se carga en la pila.

3. Se lee el token b y se carga en la pila

4. Se reduce b a B utilizando la regla [3] de la gramática.

5. Se lee el token b y se carga en la pila.

6. Se reduce Bb a B utilizando la regla [2].

7. Se carga c en la pila.

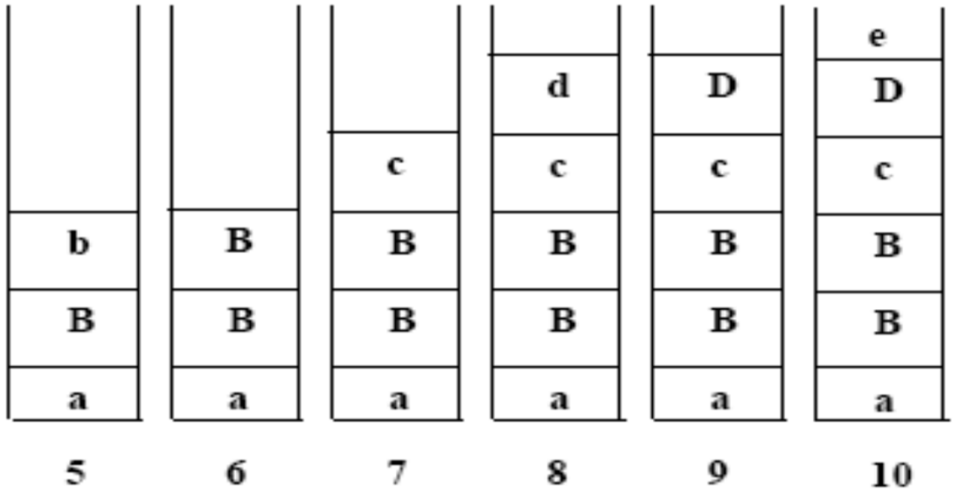
ALGORITMO DE A. SINTÁCTICO ASCENDENTE CON RETROCESO

8. Se carga d en la pila.
9. Se reduce d a D utilizando la regla [4].
10. Se carga e en la pila.
11. Se reduce aBcDe a S utilizando la regla [1].

NOTA: Se ha conseguido reducir la cadena de entrada al símbolo inicial S, por lo tanto, la sentencia pertenece al lenguaje generado por la gramática dada. A la hora de hacer una reducción, hay que tener cuidado al elegir la regla a utilizar.

ALGORITMO DE A. SINTÁCTICO ASCENDENTE CON RETROCESO

- Si en el paso 6 del ejemplo anterior se hubiera reducido utilizando la regla [3] en lugar de la [2] nos habría quedado en la pila:



y no se podría seguir adelante a partir de aquí porque en la gramática no existe ninguna regla que se ajuste a esa combinación de símbolos.

BIBLIOGRAFIA

- SANCHIS F. J., GALAN C. ***Compiladores. Teoría y Construcción.*** 1986. Madrid. Editorial Paraninfo.
- http://www.lcc.uma.es/~galvez/ftp/libros/Compiladores.pdf
- http://di002.edv.uniovi.es/~cueva/publicaciones/monografias/10_Conceptos_Basicos_Procesadores_Lenguaje.pdf

RECURSOS GRAFICOS

- Pixabay
- Pexels
- Icon-Icons