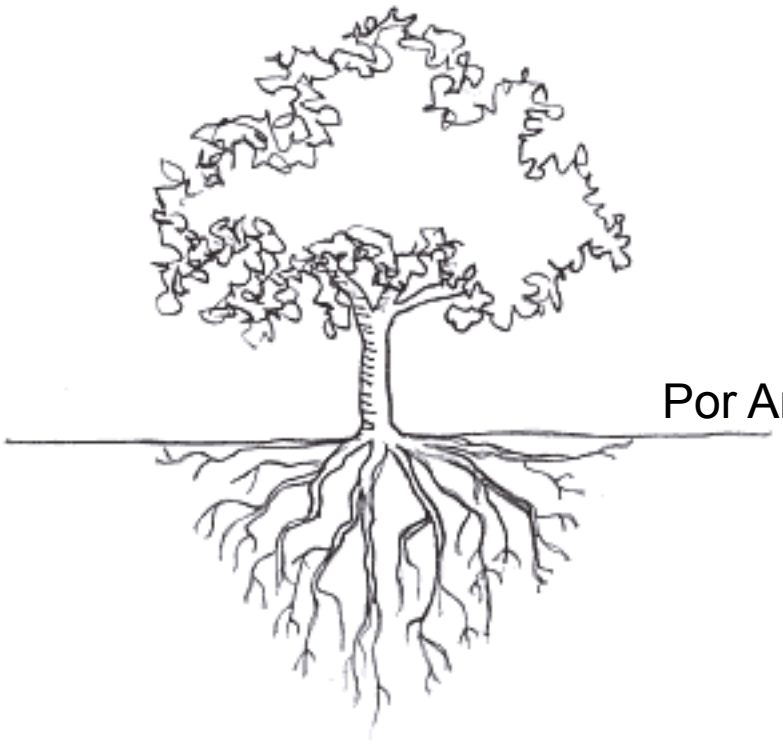
ANALIZADORES DESCENDETES

Procesadores de Lenguajes I



Por Andrea Cimmino Arriaga

INDICE

Introducción

- El presente trabajo pretende presentar de forma resumida y genérica los analizadores ascendentes de gramáticas. Así mismo al final del trabajo, tras haber expuesto las características de dichos reconocedores y haberlas comparado con los descendentes, intentare sacar una conclusión sobre cual de los dos tipos de analizadores considero mejor.

1.0 Gramáticas Independientes del Contexto.

1.1 Descripción de las GIC.

- Las GIC son aquellas en las que cada regla de producción es de la forma:

$V \rightarrow t$

- Donde V no es un símbolo terminal y es una cadena de terminales y/o no terminales.

- Las GIC generan lenguajes formales libres de contexto. Se pueden definir como:

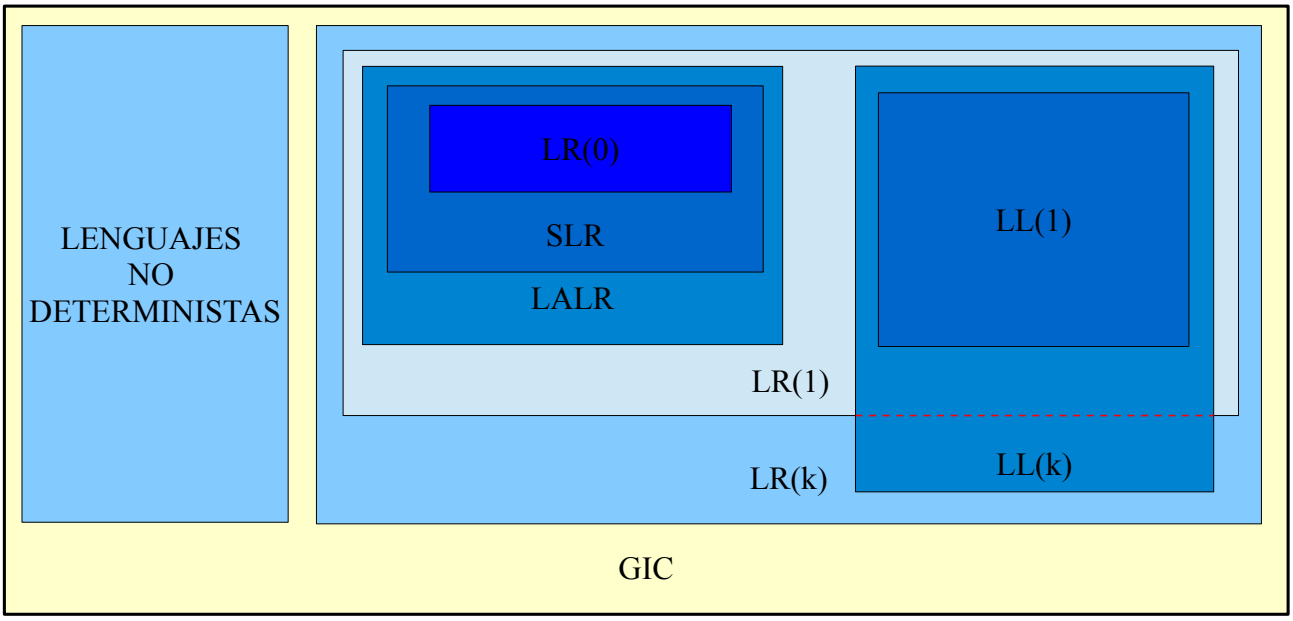
$G = (V_t, V_n, P, S)$

Donde:
- V_t es un conjunto finito de terminales.
- V_n es un conjunto no finito de terminales.
- P es un conjunto finito de producciones.
- $S \in V_t$ el denominado símbolo inicial.

- Los elementos de P son de la forma $V_n \rightarrow (V_t \cup V_n)^*$

- Un dato interesante es que la mayoría de los lenguajes de programación pueden definirse mediante este tipo de gramáticas.
-Los analizadores ascendentes reconocen lenguajes generados a partir de gramáticas LR(k), que están incluidas dentro de las GIC.

1.2 Clasificación de las gramáticas.

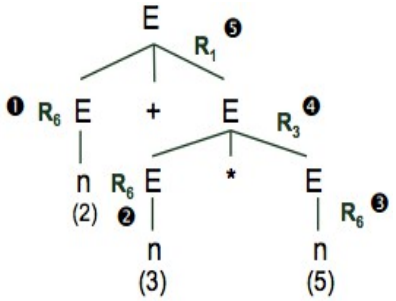


2.0 Analizadores Ascendentes

2.1 Funcionamiento general.

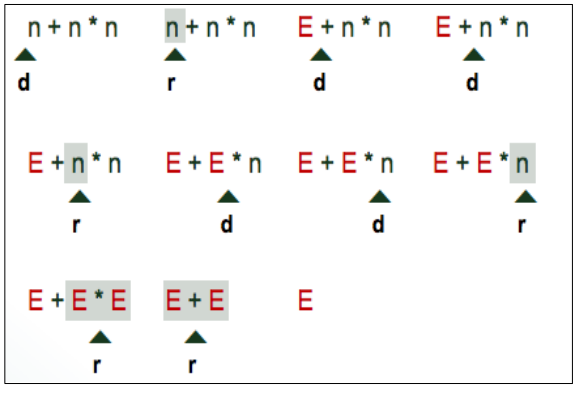
- Los Analizadores ascendentes aplican una técnica especifica del análisis sintáctico que consiste en comprobar si una cadena dada pertenece a un lenguaje definido por una gramática L(G). Para ello, parte de los elementos terminales y aplica las derivaciones a la inversa (por la derecha, Right Most Derivation), procesa la cadena de izquierda a derecha e intenta alcanzar el axioma para obtener el árbol de análisis sintáctico o error en caso de que la cadena no perteneciera a dicho lenguaje.

- Por ejemplo, dada una entrada tal que ' n + n * n':

Cadena de derivación	Arbol de análisis sintáctico
El análisis ascendente va generando una cadena de derivación por la derecha leída en sentido inverso. n + n * n ← E + n * n ← E + E * n ← E + E * E ← E + E ← E	Se parte la cadena de entrada leída de izquierda a derecha y se aplican reglas a la inversa para intentar alcanzar el axioma. 

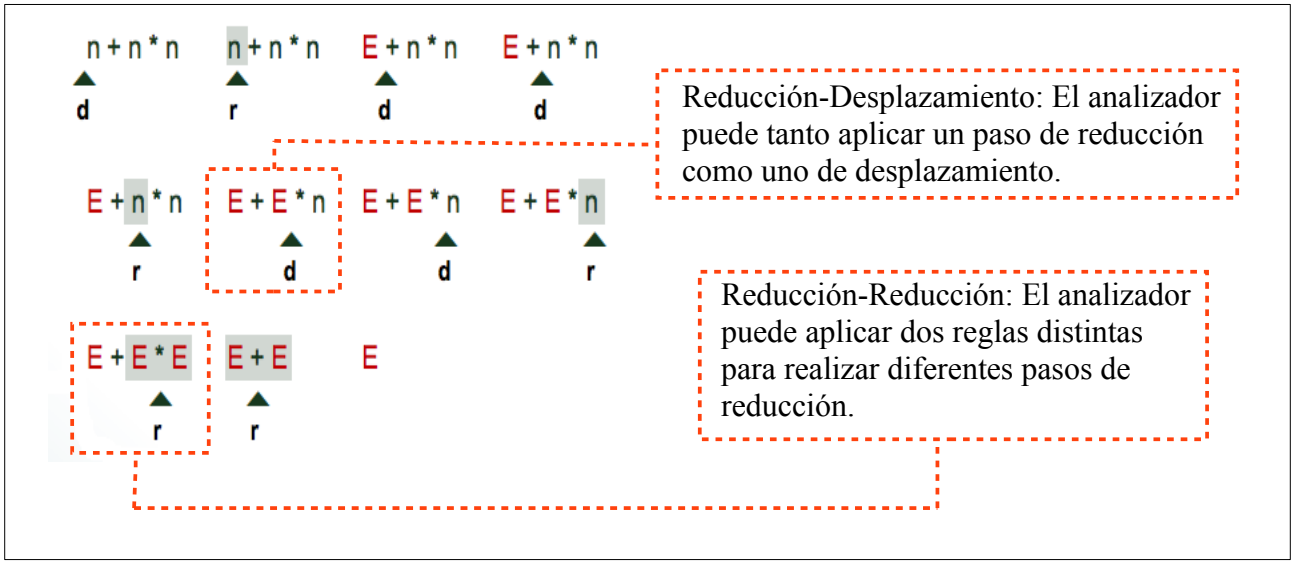
- Los analizadores sintácticos ascendentes son capaces de decidir qué regla de producción aplicar a cada paso en función de los elementos terminales que se encuentran en la cabeza de lectura de la cadena de entrada. Como consecuencia se consigue un proceso de análisis con complejidad lineal O (n) con respecto al tamaño del problema. El conjunto de gramáticas que pueden ser analizadas mediante un análisis ascendente lineal se denomina LR(1) que es mucho más amplio que el de las gramáticas LL(1) vistas en clase. Por ese motivo estos analizadores son llamados genéricamente analizadores LR(k) o analizadores por reducción-desplazamiento.

- A cada paso el analizador decide realizar una de dos operaciones posibles:



- **Desplazar** (d): El analizador debe procesar un número suficiente de terminales desplazando la cabeza lectora antes de aplicar una regla de producción a la inversa. Esta operación se llama desplazar.
- **Reducir** (r): Cuando se reconoce en la forma de frase en curso un fragmento igual a la parte derecha de una regla se sustituye éste por el antecedente. Esta operación se llama reducir

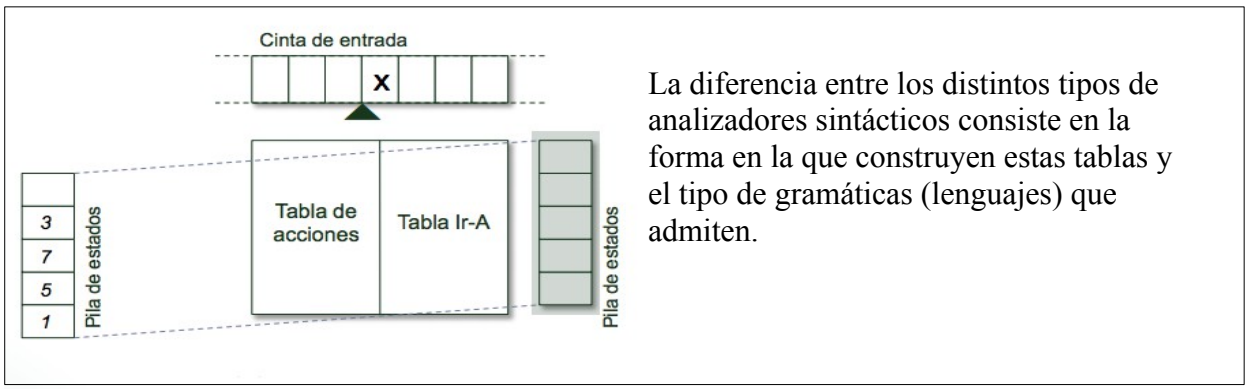
- En este proceso pueden surgir 2 tipos de conflictos que el analizador debe ser capaz de resolver:



2.2 Arquitectura general de los analizadores ascendentes.

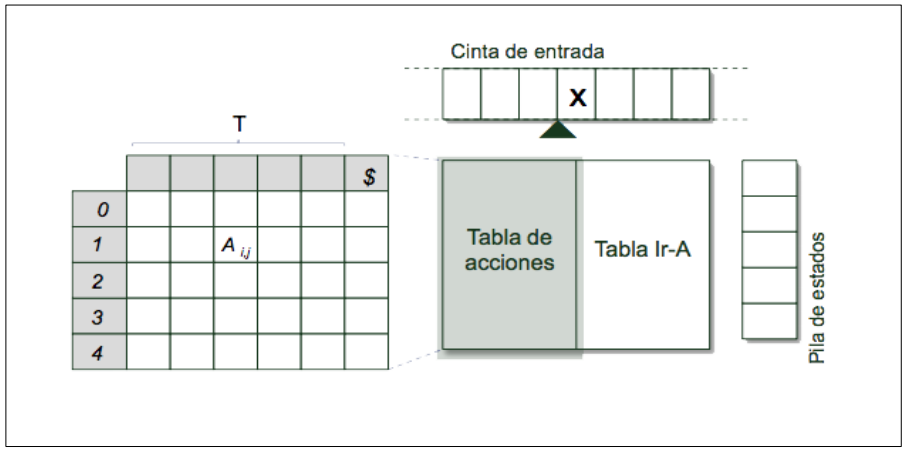
- Todos los tipos de analizadores sintácticos ascendentes tienen un mismo modelo de arquitectura general, esta consta de 3 elementos fundamentales:

- Pila de estados: El analizador sintáctico puede atravesar, a lo largo del proceso de análisis, distintos estadios de compilación que dependen del lenguaje analizado. Estos estadios se identifican con estados codificados numéricamente que se gestionan en una pila. En cada momento el estado en curso aparece en la cima de la pila.

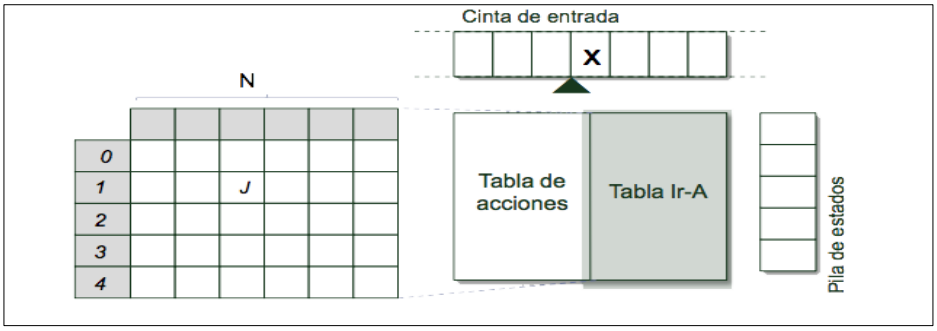


- Tabla de acciones: Para cada estado y cada terminal (o $) a la entrada el analizador tiene información de que acciones debe realizar. Las operaciones posibles son:

  > $\mathbf{d_j}$: Desplazar y apilar el estado j
  > $\mathbf{r_k}$: Reducir la regla r
  > **e** : Emitir un error
  > **ok** : Aceptar una cadena de entrada

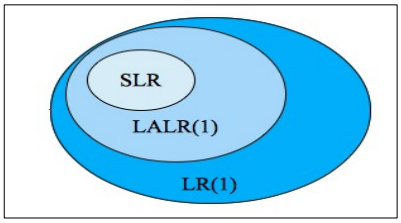


- Tabla Ir-A: En el caso de aplicar reducción, y del estado de compilación en curso, esta tabla indica el estado que hay que apilar en función del antecedente de la regla de producción aplicada.



2.3 Tipos de analizadores ascendentes.

- Tenemos tres tipos de analizadores:



- Analizadores SLR o LR simple: Determinan la regla aplicar consultando el primer terminal a la entrada.

Ventajas	Inconvenientes
> Fácil de entender y construir. > Un estado puede tener varias reducciones. > Utiliza un autómata muy simple (autómata **LR(0)**).	> Estrategias de reducción mas complejas. > Tiene un conjunto reducido de gramáticas.

- Analizadores LR(1) o LR Canónico: Determinan la regla aplicar consultando el primer terminal a la entrada.

Ventajas	Inconvenientes
> Gran conjunto de gramáticas posibles. > Varias reducciones para un estado. > Las predicciones son fiables.	> Más difícil de entender y construir. > Estrategia de reducción compleja. > No existen estados únicos de reducciones. > Utiliza un autómata muy complejo para crear las tablas. No puede usar **LR(0)**.

- Analizadores LALR o LR de examen anticipado: Determinan la regla aplicar consultando los primero K símbolos terminales a la entrada.

Ventajas	Inconvenientes
> Gran conjunto de gramáticas posibles. > Varias reducciones para un estado. > Las predicciones son muy fiables.	> Más difícil de entender y construir. > Estrategia de reducción compleja. > No existen estados únicos de reducciones. > Utiliza un autómata muy complejo para crear las tablas. No puede usar **LR(0)**.

2.4 Comparativa analizadores ascendentes y descendentes.

Ascendentes	Descendentes
Se parte de los terminales y se construye la inversa de una derivación para intentar alcanzar el axioma.	Se parte del axioma y se aplica una cadena de derivaciones para construir un árbol sintáctico.
Disminuye el número de reglas mal aplicadas con respecto al tipo descendente, por la forma en que construye el árbol.	Requiere realizar una serie de operaciones antes de usar las gramáticas: • Eliminar ambigüedad. • Eliminar recursividad por la izquierda. • Factorizar.
Mas complejos de implementar.	Sencillos de implementar.
Mayor numero de gramáticas susceptibles de ser analizadas.	

2.5 Conclusiones.

- Los dos métodos, tanto ascendentes como descendentes tienen una serie de ventajas e inconvenientes que los hacen más útiles en ciertas ocasiones. Lo más remarcable quizás, y haga que los analizadores ascendentes sean mejores, es que debido a su forma de construir el árbol se disminuye el número de reglas mal aplicadas con respecto a los ascendentes y ademas puedan analizar más gramáticas. Es decir, computacionalmente y pensando en eficiencia, aunque complejos de implementar dan mas prestaciones; con lo que si el coste de implementación no es relativamente alto deberían ser mejores que los analizadores descendentes.

3.0 Webgrafía.

- http://sycg.wordpress.com/2010/12/14/analizador-sintactico-ascendente-y-descendente/
- http://es.wikipedia.org/wiki/Analizador_sint%C3%A1ctico_LR
- http://robsitemas.wordpress.com/2010/12/14/analizador-ascendente-y-descendente/
- http://www.escet.urjc.es/~procesal/transp/Tema3-AnalizadorSintactico-LR-SLR.pdf
- http://www.escet.urjc.es/~procesal/transp/Tema3-AnalizadorSintactico-LR-LR1-LALR.pdf