Procesadores de Lenguajes

Tema 4

Análisis sintáctico

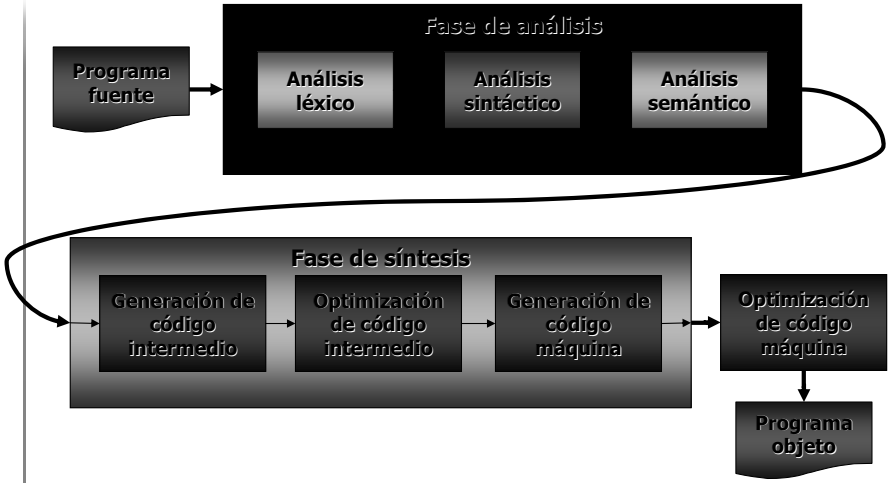
Tema 4 – Análisis Sintáctico

Sesión 1: Conceptos básicos del análisis sintáctico

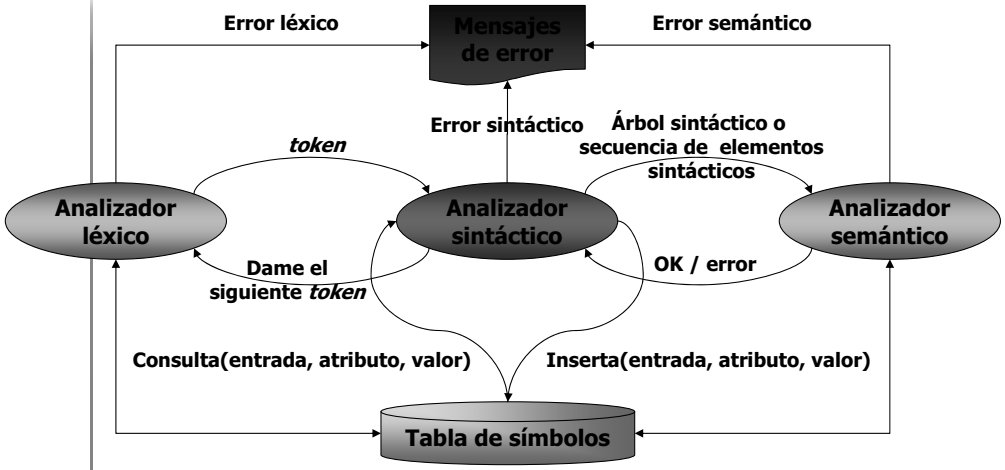
Sesión 1: Conceptos básicos del análisis sintáctico

- Generalidades:
  - El análisis sintáctico en contexto
  - ¿Por qué separar análisis léxico y sintáctico?
  - En qué consiste el análisis sintáctico
  - Tipos de analizadores sintácticos
    - Análisis sintáctico descendente
    - Análisis sintáctico ascendente
  - Árboles de derivación y *parses* (con ejemplo)
  - El analizador sintáctico como traductor
- Diseño de alto nivel:
  - Estrategias para el análisis sintáctico
    - Eficiencia de analizadores sintácticos
  - Tipos de gramáticas para el análisis sintáctico

El análisis sintáctico en contexto (I)



El análisis sintáctico en contexto (II)



¿Por qué separar análisis léxico y sintáctico?

- Se simplifica el diseño del analizador sintáctico:
  - No ha de preocuparse de leer el fichero de entrada, ni procesar blancos, ni comentarios, ni de recibir caracteres inesperados.
- El diseño general del compilador se hace más claro y comprensible.
- Se mejora la eficiencia del compilador en su conjunto:
  - La lectura del programa fuente suele requerir gran parte del tiempo de compilación, que se ve reducido si el analizador lexicográfico incorpora técnicas especiales de lectura, o está realizado en ensamblador.
- Aumenta la portabilidad del compilador:
  - Abstrae al resto del compilador de todas las diferencias que se produzcan en el alfabeto de entrada, o en el dispositivo de almacenamiento (por ejemplo: migrar de programas fuente en ASCII al estándar EBCDIC).

En qué consiste el análisis sintáctico

- Analizar sintácticamente una cadena de *tokens* no es más que encontrar para ella el árbol sintáctico o la cadena de derivación que tiene:
  - como raíz, el axioma de la gramática,
  - como nodos terminales, la sucesión ordenada de símbolos que componen la cadena analizada.
- En caso de no existir este árbol sintáctico (o la cadena de derivaciones), la cadena de tokens no pertenecerá al lenguaje, y el analizador sintáctico ha de emitir el correspondiente mensaje de error.

Tipos de analizadores sintácticos

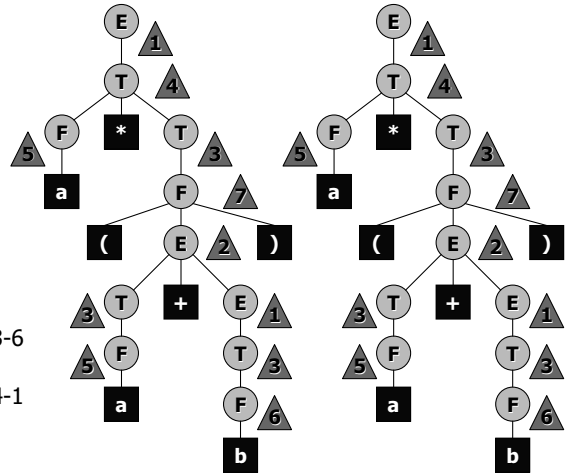
- **Analizadores descendentes:**
  - Partiendo de la raíz del árbol (el axioma inicial de la gramática), construyen el árbol sintáctico de análisis (descendiendo) hacia sus nodos terminales; para ello, se utilizan derivaciones por la izquierda, hasta llegar a reconocer (salvo error) la cadena analizada.
- **Analizadores ascendentes:**
  - Partiendo de sus nodos terminales (la cadena de tokens), construyen el árbol sintáctico de análisis (ascendiendo) hacia su raíz (el axioma inicial de la gramática); para ello, se utilizan derivaciones por la derecha hasta llegar a reconocer (salvo error) la cadena analizada.

Árboles de derivación y *parses*

- Para representar el árbol sintáctico que conduce hasta una cadena, se asigna a cada regla de la gramática un número.
- ***Parse*:**
  - Secuencia ordenada de los números de las reglas aplicadas para construir el árbol de derivación de la cadena de *tokens* de entrada.
  - ***Parse* izquierdo**:
    - Son los números de las reglas de derivación izquierda utilizadas para generar la cadena a partir del axioma – recorrido del árbol en **preorden** (usado en *análisis descendente*).
  - ***Parse* derecho**:
    - Son los números de las reglas de derivación derecha utilizadas para generar la cadena a partir del axioma (recorrido del árbol en **postorden**) en orden inverso. El tomar el orden inverso viene condicionado por ser el *análisis ascendente* el que normalmente utiliza las reglas de derivación derecha, con lo que el orden en el que aparecen al realizar el análisis es **invertido**.

Árboles de derivación y *parses* : ejemplo

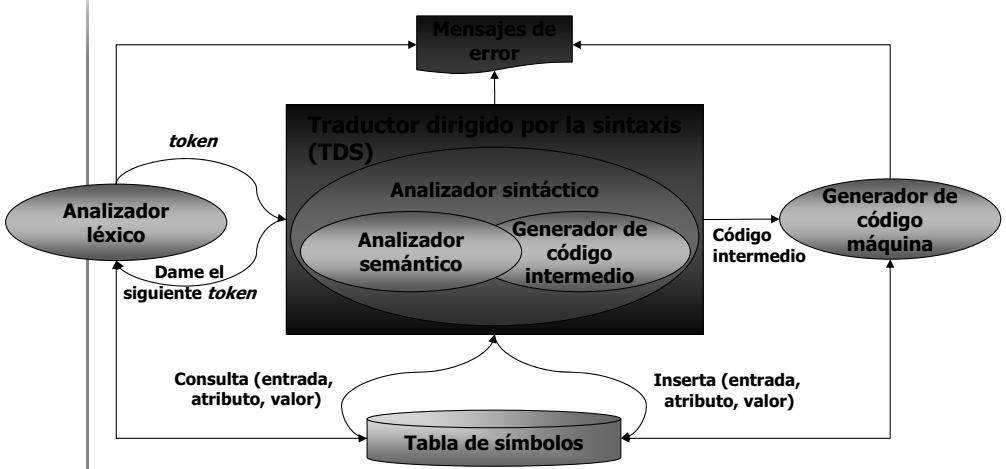
- Dada la gramática:
  1. E → T
  2. E → T+E
  3. T → F
  4. T → F*T
  5. F → a
  6. F → b
  7. F → (E)
- y la sentencia:
  - a*(a+b)
- El parse izquierdo es:
  - 1-4-5-3-7-2-3-5-1-3-6
- y el derecho:
  - 5-5-3-6-3-1-2-7-3-4-1



El analizador sintáctico como traductor (I)

- Traduce la cadena de *tokens* del fichero de entrada (programa fuente) a una estructura que la represente desde un punto de vista sintáctico:
  - Árbol sintáctico (didácticamente, muy útil; en la práctica, poco eficiente).
  - Secuencia de reglas usadas para reconocer (analizar) sintácticamente la cadena de entrada (*parse*).
  - Vertebrando, **mediante su propio código**, el resto de tareas del compilador, excepto la de generación de código final (máquina) – traducción dirigida por la sintaxis (TDS).

El analizador sintáctico como traductor (II)



Estrategias de análisis sintáctico (I)

- Propiedades deseables en un analizador sintáctico:
  - Eficiencia ⇒ minimizar:
    - Su coste:
      - Temporal
      - De memoria, aunque este criterio sea secundario.
    - La dimensión (*k*) de la ventana de *tokens* que se considera en cada momento (preanálisis)
  - Ausencia de retrocesos (*backtracking*)
    - Las acciones semánticas son difíciles de deshacer

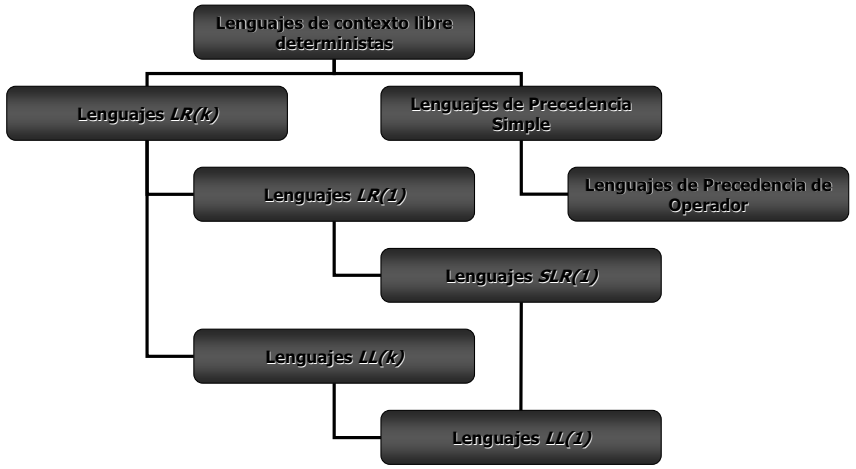
Estrategias de análisis sintáctico (II)

- Eficiencia de analizadores sintácticos:
  - Si *n* es el número de tokens del programa fuente:
    - Para cualquier gramática de contexto libre:
      - Se puede construir un analizador sintáctico de complejidad $O(n^3)$ ⇒ muy costoso
    - Los lenguajes informáticos, en general, se diseñan con reglas de contexto libre que permitan su análisis sintáctico mediante un algoritmo lineal ($O(n)$):
      - Mediante el análisis de izquierda a derecha de la entrada
      - Considerando un único *token* en cada momento

Estrategias de análisis sintáctico (III)

	LL(K)	LR(K)
Dirección de lectura de los *tokens*	***Left-to-right (L)*** **(de izquierda a derecha)**	***Left-to-right (L)*** **(de izquierda a derecha)**
Tipo de derivación realizada	***Leftmost derivation (L)*** **(por la izquierda)**	***Rightmost derivation (R)*** **(por la derecha)**
Dimensión de la ventana de tokens de preanálisis (*look-ahead*)	***k***	***k***
Tipo de analizador	**Descendente**	**Ascendente**

Tipos de lenguajes formales para el análisis sintáctico



Tipos de analizadores sintácticos

- Analizador descendente:
  - Analizador descendente recursivo:
    - Con retroceso
    - Sin retroceso (predictivo)
  - Analizador descendente no recursivo predictivo (≡ tabular):
    - Analizador *LL(K)*
      - Analizador *LL(1)*
- Analizador ascendente:
  - Analizador ascendente con retroceso
  - Analizador de gramáticas de precedencia simple
  - Analizador de gramáticas de precedencia de operador
  - Analizador *LR(K)*
    - Analizadores *LR(1)*
      - Analizadores *SLR(1)*

Bibliografía

- Aho, A. V.; Sethi, R.; Ullman, J. D.: *Compilers: Principles, Techniques and Tools*. Massachusetts: Addison-Wesley Publishing Company, 1986.
- Alfonseca Cubero, E.; Alfonseca Moreno, M.; Moriyón Salomón, R. Teoría de autómatas y lenguajes formales. Madrid: Mc-Graw-Hill/Interamericana de España, S.A.U., 2007.
- Grogono, P. Programación en Pascal. Wilmington, Delaware (EE.UU.):Addison-Wesley Iberoamericana, 1996.
- Sanchís Llorca, F. J. y Galán Pascual, C. Compiladores: Teoría y construcción. Madrid: Editorial Paraninfo, 1986.