Universidad de Huelva

Universidad de Huelva

Grado en Ingeniería Informática

Tema 2. Análisis Léxico

Resumen

Autor: Alberto Fernández Merchán
Asignatura: Procesadores de Lenguajes

1. Introducción

1.1. Características

- Lee caracteres.
- Produce *tokens* (Componentes Léxicos).
- Filtra comentarios y separadores múltiples (blancos).
- Lleva un contador de línea y columna del texto fuente.
- Genera errores en caso de que la entrada no corresponda a ninguna categoría léxica.

1.2. Definiciones

- Categoría Léxica: Símbolo elemental del lenguaje fuente.
- Componente Léxico (token): Elemento que pertenece a una categoría léxica.
- Atributos de un componente: Información del componente necesarias en etapas posteriores del análisis.
- Lexema: Cadena de caracteres correspondiente al componente léxico.

1.3. Categorías Léxicas Habituales

- Palabras Clave: Palabras con significado especial. (Palabras Reservadas).
- Identificadores: Nombre de variables, constantes, funciones, tipos...
- Operadores: Símbolos identificadores de operaciones aritmético-lógicas.
- Constantes Numéricas: Literales que especifican valores numéricos.
- Constantes de Carácter o cadenas: Literales con el valor de un carácter o cadena.
- Símbolos Especiales: Separadores, delimitadores, terminadores...
- Blancos: Caracteres de separación de componentes léxicos.
- Comentarios: Información para el lector.
- Fin de Entada: Simboliza el final de la lectura.

2. Especificación de categorías léxicas

2.1. Gramáticas Regulares

Tipos de Producciones:

- $< A > \rightarrow\ a < B >$
- $< A > \rightarrow\ a$
- $< A > \rightarrow \lambda$

2.2. Expresiones Regulares

- Concatenación de Lenguajes: $LM = \{xy \mid x \in L \wedge y \in M\}$
- Notación: $L^k = LLLL...L \ \ (k-1 \ \ veces)$
- Clausura de L: Conjunto de cadenas que pueden obtenerse concatenando un número arbitrario de cadenas de L. $L^* = \bigcup_{k=0}^{\infty} L^k$
- Lenguaje Vacío: $L_\phi = \phi$
- Lenguaje con la cadena vacía: $L_\lambda = \{\lambda\}$

Abreviaturas de Expresiones Regulares

- Los corchetes representan opciones: [“a”,“f”,“j”] = (“a”| “f”| “j”).
- El guión indica intervalo: [“0-3”] = [“0”, “1”, “2”, “3”].
- El signo + indica clausura positiva. (Todas las cadenas excepto λ).
- El signo ? indica opcionalidad. $r? = (r \mid \lambda)$
- El símbolo $\sim$ significa caracteres excluidos: $\sim$ [“0” − “9”]. (Todos los caracteres excepto los dígitos)
- $\sim$ [ ]: Cualquier carácter.

3. Autómatas Finitos No Deterministas

3.1. Reconocedores de Lenguajes Regulares

Programa que toma una cadena como entrada y devuelve verdadero o falso en función de si la cadena pertenece o no al lenguaje.

Los lenguajes regulares pueden ser reconocidos por autómatas finitos.Existen dos tipos de autómatas finitos: deterministas y no deterministas.

3.2. Definición

Un AFN se define mediante la quíntupla: $AFN = (Q, \Sigma, E, q_0, F)$ donde:

- Q: Conjunto finito de estados.
- Σ: Alfabeto de símbolos.
- E: Conjunto de transiciones entre estados.
- q_0: Estado inicial.
- F: Conjunto de estados iniciales.

3.3. Funcionamiento

La cadena x será aceptada si existe un camino que parta de q_0, acepte la entrada x y termine en, al menos, un estado final.

Se denomina clausura-$\lambda(E)$ al conjunto de estados formado por todos los estados de E más aquellos a los que se puede acceder desde E con transiciones λ.

3.4. Creación de un AFN a partir de una Gramática Regular

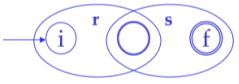
- Cada símbolo no terminal → estado.
- El axioma genera el estado inicial.
- Se añade un estado final F.
- Cada producción tipo $A \rightarrow aB$ genera una transición del estado A al B con el símbolo 'a'.
- Cada producción tipo $A \rightarrow a$ genera una transición del estado A al F con el símbolo 'a'.

3.5. Creación de un AFN a partir de una Expresión Regular

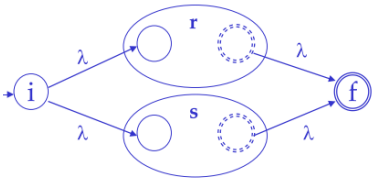
- λ:

- a:

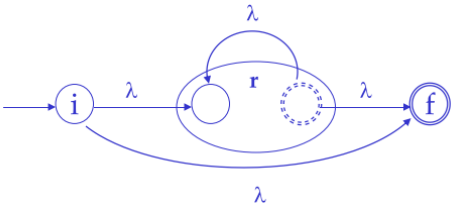
- rs:



- $r \mid s$:



- r^*:



4. Autómatas Finitos Deterministas

4.1. Definición

En un AFD no se admiten transiciones lambda y no existen transiciones que partan del mismo estado con el mismo símbolo.

4.2. Funcionamiento

Una cadena x es aceptada si existe un camino que parta del estado inicial, acepte la entrada x y termine en un estado final.

4.3. Creación de un AFD a partir de un AFN

Los AFD tienen la misma capacidad expresiva que los AFN, cada macroestado del AFN corresponde con un estado del AFD. El algoritmo es el siguiente:

1. Generar el estado inicial del AFD como el macroestado inicial del AFN e incluirlo en una lista de estados por analizar.
2. Analizar el primer estado de la lista por analizar.
   a) Extraer el primer estado de la lista e introducirlo en una lista de estados analizados.
   b) Estudiar las transiciones del estado para cada símbolo del alfabeto.
   c) Si el macroestado correspondiente a una transición no ha aparecido anteriormente, se crea un nuevo estado del AFD correspondiente a ese macroestado.
3. Repetir el paso anterior hasta que no queden estados por analizar.
4. Los estados finales son aquellos que correspondan a macroestados con algún estado final del AFN.

4.4. Creación de un AFD a partir de una Expresión Regular

Podemos generar un AFN a partir de una ER y luego transformarlo a AFD o, por otro lado, generar el AFD directamente a partir de la ER siguiendo el siguiente algoritmo:

1. Se introduce un punto en la expresión regular para indicar la parte reconocida en cada momento.
2. Un estado del autómata está asociado a un conjunto de expresiones regulares con puntos.
3. El estado inicial se obtiene colocando el punto al comienzo de la ER.
4. Las transiciones de cada estado corresponden al consumo de algún símbolo y dan lugar al desplazamiento del punto en la ER.

4.5. Minimización de un AFD

Para minimizar un AFD seguimos el siguiente algoritmo:

1. Crear una partición con dos grupos: estados finales y los no finales.
2. Para cada grupo con varios estados, dividir el grupo en subgrupos tales que dos estados s y t estén en el mismo subgrupo si y solo si para cada símbolo a, existe una transición desde s con el símbolo a hacia un estado de un cierto grupo y desde t hacia ese mismo grupo.
3. Repetir el paso 2 hasta que no se dividan más grupos.
4. Cada grupo es un estado del AFD
5. Eliminar los estados no alcanzables desde el estado inicial y los que no tengan transiciones que puedan conducir a un estado final.

4.6. Comparación AFD - AFN

- Tienen la misma capacidad descriptiva.
- El número de estados de un AFN crece linealmente con el tamaño de la expresión, mientras que, en un AFD crece exponencialmente con el tamaño de esta.
- El tiempo de análisis de un AFD es lineal $O(n)$, mientras que el de un AFN es de $O(n*r)$ donde n es el tamaño de la cadena y r el de la expresión.

5. Implementación de un analizador léxico

5.1. Características

Cada categoría léxica tiene asociada su expresión regula y un conjunto de acciones. Se sigue una estrategia avariciosa, es decir, se intentan reconocer la cadena más larga antes de cambiar de categoría léxica. Para implementar los analizadores léxicos utilizaremos una máquina discriminadora determinista

5.2. Máquina Discriminadora Determinista

Es similar a un AFD y tiene asignadas acciones a los estados finales. Para crear una se sigue los siguientes pasos:

1. Añadir un símbolo especial a cada expresión regular.
2. Unir las expresiones en una.
3. Construir el AFD de la expresión.
4. Eliminar estados finales y arcos de los símbolos especiales.

categoría	Expresión regular
entero	[0-9][0-9]*#$_{entero}$
real	[0-9][0-9]*\.[0-9][0-9]*#$_{real}$
identificador	[a-zA-Z][a-zA-Z0-9]*#$_{identificador}$
asignación	:=#$_{asignación}$
rango	\.\.#$_{rango}$
blanco	[\t\n][\t\n]*#$_{blanco}$
eof	eof#$_{eof}$

Figura 1: Paso 1

Expresión regular
[0-9][0-9]*#$_{entero}$ | [0-9][0-9]*\.[0-9][0-9]*#$_{real}$ | [a-zA-Z][a-zA-Z0-9]*#$_{identificador}$ | :=#$_{asignación}$ | \.\.#$_{rango}$ | [\t\n][\t\n]*#$_{blanco}$ | eof#$_{eof}$

Figura 2: Paso 2

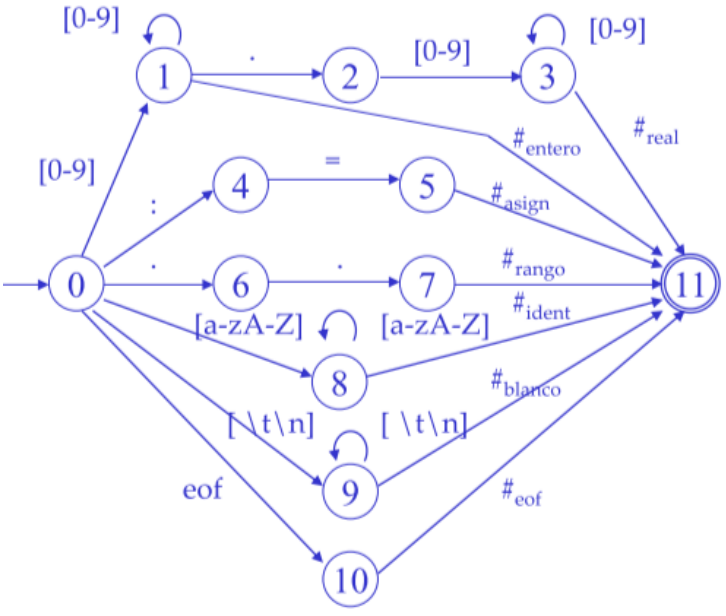


Figura 3: Paso 3

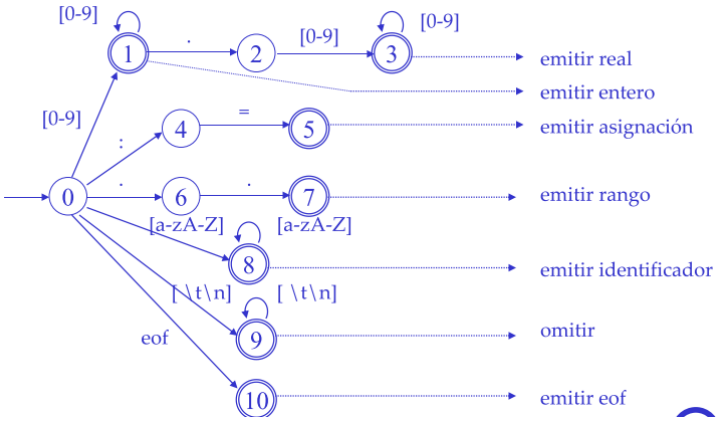


Figura 4: Paso 4-6

5. Convertir en estados finales los estados origen de los arcos de símbolos especiales.

6. Asociar a estos estados las acciones correspondientes.

Inicialmente la máquina se encuentra en el estado 0. Evoluciona como un AFD hasta encontrar un carácter no reconocido, retrocede hasta el último estado final encontrado y ejecuta las acciones asociadas al estado final. Por último, vuelve al estado inicial.

5.3. Tratamiento de Errores

Cuando no se ha alcanzado ningún estado final siguiendo el comportamiento anterior, se dice que se ha detectado un error.

Para tratarlos, se genera un error léxico, se devuelven los caracteres leídos, se elimina el primero y se continúa el análisis.

5.4. Tratamiento de Palabras Reservadas

Se pueden confundir con identificadores, para solucionar el problema, se deben tratar como identificadores y almacenar todas las palabras reservadas en una tabla. Antes de emitir el identificador, buscar en la tabla esa cadena. Si aparece, emitirá el token de palabra reservada, si no, el de identificador.

Otra opción es utilizar máquinas discriminadoras no deterministas.

5.5. Máquina Discriminadora No Determinista

Son similares a las MDD y están basadas en AFN. Para crearlas se siguen los siguientes pasos:

1. Ordenar las diferentes expresiones regulares

2. Unir todas las expresiones regulares

3. Generar el AFN

4. Añadir acciones a los estados finales.

Inicialmente la máquina se encuentra en el estado 0. Evoluciona como un AFN hasta encontrar un carácter no reconocido, retrocede a la última transición que contuviera, al menos, un estado final. Si encuentra varios estados finales, selecciona el primero en el orden establecido en el paso 1. Ejecuta las acciones asociadas a ese estado final y, por último, vuelve al estado inicial.

categoría	Expresión regular
class	class
int	int
long	long
double	double
identificador	[a-zA-Z][a-zA-Z0-9]*
blanco	[\t\n][\t\n]*
eof	eof

Figura 5: Paso 1

Expresión regular
class | int | long | double | [a-zA-Z][a-zA-Z0-9]* | [\t\n][\t\n]* | eof

Figura 6: Paso 2

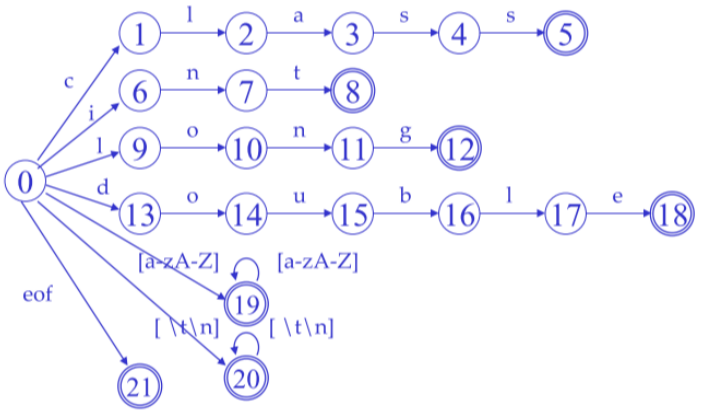


Figura 7: Paso 3

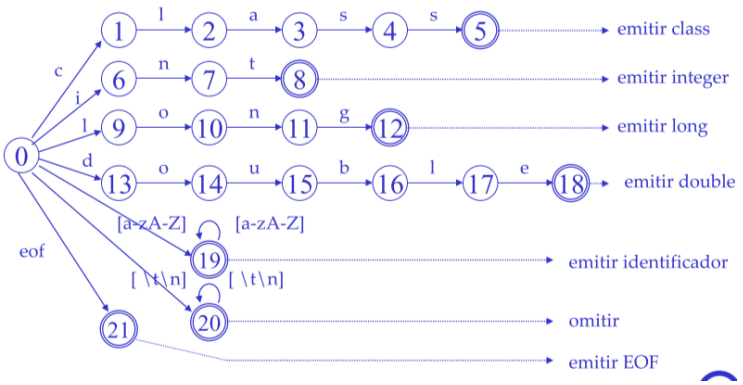


Figura 8: Paso 4