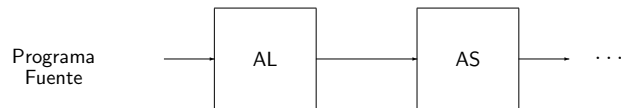



2. Análisis Léxico

- Introducción al problema del Análisis Léxico
- Formalismo de especificación léxica de los Lenguajes de Programación
- Construcción de un AL
- Generación automática de un AL



Especificación léxica de un Lenguaje Natural (Castellano)

Lema	Definición	Instanciación
árbol	(Del lat. <i>arbor</i> , - <i>ōris</i>) 1. m. Planta perenne, de tronco leñoso y elevado, que se ramifica a cierta altura del suelo.	

Especificación léxica de un Lenguaje de Programación

- **Lema** ⇒ Categoría sintáctica (o símbolo) del LP
- **Definición** ⇒ Patrones (definición) de los símbolos del LP
- **Lexema** ⇒ Cadena de caracteres que es una posible instanciación de un símbolo o categoría sintáctica

Ejemplo de especificación léxica de un LP

Símbolos	Patrones de los símbolos	Lexemas
identificador	Cadena alfanumérica, con el primer carácter alfabético.	x25
constante entera	Cadena de uno o más dígitos	127
operador relacional	<, <=, >, >=, ==, !=	>
operador de asignación	=	=
...

Objetivo del AL

Detectar e identificar las componentes léxicas presentes en la cadena (programa fuente) de entrada

Componente léxica (o "token")

- Código del símbolo (o categoría sintáctica)
- Atributo(s) del símbolo (o categoría semántica)

Ejemplo

$m = x > 2 \Rightarrow (\text{id}, \text{ind}_{(m)}) (\text{opasig}) (\text{id}, \text{ind}_{(x)}) (\text{oprel}, \text{cod}_{(>)}) (\text{cte}, \text{val}_{(2)})$

Expresiones Regulares (Especificación)

- Dado un alfabeto Σ , una Expresión Regular (ER) sobre Σ es:
 - \emptyset es una ER y define el lenguaje \emptyset ,
 - ϵ es una ER y define el lenguaje $\{\epsilon\}$,
 - $a \in \Sigma$ es una ER y define el lenguaje $\{a\}$.
- Si r y s son ER, siendo L_r y L_s sus lenguajes respectivos, entonces se cumple:
 - $r \cdot s$ es una ER y define el lenguaje $L_r \cdot L_s$,
 - $r \mid s$ es una ER y define el lenguaje $L_r \cup L_s$,
 - r^+ es una ER y define el lenguaje L_r^+ ,
 - r^* es una ER y define el lenguaje L_r^* .

Ejemplo

letra	$a \mid b \mid \dots \mid z$
dígito	$0 \mid 1 \mid \dots \mid 9$
identificador	$\text{letra} \cdot (\text{letra} \mid \text{dígito})^*$
cte. entera	$\text{dígito} \cdot (\text{dígito})^*$
op. de asignación	$=$
op. relacional	$> \mid >= \mid < \mid <= \mid == \mid !=$

Autómatas de Estados Finitos Determinista (Análisis)

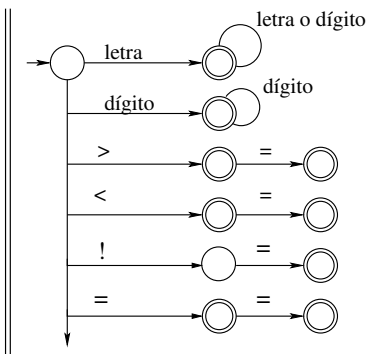
$$AEF = (\Sigma, Q, q_0, F, \delta) \quad \text{Donde: } F \subseteq Q; \quad q_0 \in Q; \quad \delta : Q \times \Sigma \rightarrow Q$$

Equivalencia $ER \Leftrightarrow AEF$

$ER \Leftrightarrow AEFND$ con transiciones $\epsilon \Leftrightarrow AEFND \Leftrightarrow AEFD \Leftrightarrow AEFD$ mínimo

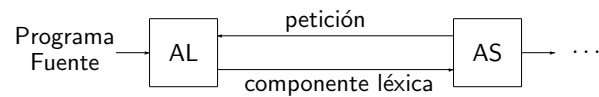
Ejemplo

letra	$a \mid b \mid \dots \mid z$
dígito	$0 \mid 1 \mid \dots \mid 9$
identificador	$\text{letra} \cdot (\text{letra} \mid \text{dígito})^*$
cte. entera	$\text{dígito} \cdot (\text{dígito})^*$
op. de asignación	$=$
op. relacional	$> \mid >= \mid < \mid <= \mid == \mid !=$



CONSTRUCCIÓN DE UN AL

1. AL como una rutina del AS



2. Detección de los símbolos del lenguaje

2.1 No es un problema tan fácil

- Ejemplo FORTRAN.- No existen palabras reservadas y el espacio en blanco no es un separador:

```
DO 10 I = 1,27          DO 10 I = 1.27
```

- Ejemplo PL/I.- No existen palabras reservadas:

```
if then then then = else; else else = then
```

CONSTRUCCIÓN DE UN AL

2.2 Detección de las palabras reservadas

- Palabras reservadas como expresiones regulares.
- Palabras reservadas como identificadores especiales (tabla de palabras reservadas).

3. Acciones asociadas a la detección de un símbolo

3.1 Emitir las componentes léxicas ("tokens") detectadas

3.2 Tratamiento de errores léxicos

3.3 Manipulación de la Tabla de Símbolos

- Gestión de la Tabla de Símbolos (Lenguajes sencillos, p.ej. FORTRAN)
- Gestión de la Tabla de Nombres (Lenguajes complejos con estructura de bloques)

CONSTRUCCIÓN DE UN AL

4. Otras funciones del AL

- eliminación de cadenas inútiles: comentarios, tabuladores, saltos de línea, etc.,
- lectura eficiente del fichero de entrada,
- relación de los mensajes de error con las líneas del programa fuente.

5. Estrategias de implementación de un AL: de AEFD a programas

- Implementación manual directa del programa
- Implementación manual basada en la matriz de transiciones del AEFD
- Implementación basada en un generador automático de AL

GENERADORES AUTOMÁTICOS DE AL: FLEX

FLEX es una herramienta de código abierto que convierte una descripción léxica de un LP, basada en expresiones regulares, en un programa C (o C++). El énfasis se pone en el rendimiento.

```
< declaraciones >
%%
< reglas de traducción >
%%
< procedimientos auxiliares >
```

- **Declaraciones.**- Definiciones de expresiones regulares auxiliares.
- **Reglas de traducción.**- Tienen la siguiente forma:

$$p_i \{ \text{acción}_i \} \quad i : 1 \dots n.$$

Donde p_i es la expresión regular que define un símbolo y acción_i es el segmento de programa con las acciones asociadas a la detección del símbolo.

- **Procedimientos auxiliares.**- Segmentos de programa auxiliares.

```
letra      [a-zA-Z]
digito     [0-9]

%%
"<"        { return(MENOR_);      }
"<="       { return(MENORIG_);    }
">"        { return(MAYOR_);      }
">="       { return(MAYORIG_);    }
"=="       { return(IGUAL_);      }
"!="       { return(DISTINTO_);   }
"="        { return(ASIG_);       }
{letra}({letra}|{digito})* { return(ID_); ind(); }
{digito}({digito})*      { return(CTE_); val(); }
%%
void ind()
/* Busca un nombre en la Tabla de Nombres, si no lo encuentra
   lo crea. Devuelve la posición en la Tabla de Nombres. */
    ...

void val()
/* Devuelve el valor numérico asociado al lexema. */
    ...
```