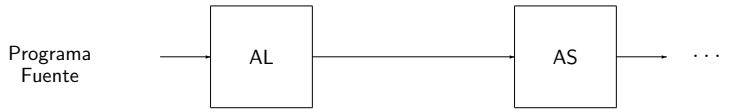
2. Análisis Léxico

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



Especificación léxica del `Castellano`

Lema	Definición	Instanciación
árbol	`(Del lat. arbor, -ŏris)` 1. m. Planta perenne, de tronco leñoso y elevado, que se ramifica a cierta altura del suelo.	encina, roble, etc.

Especificación Léxica		**Análisis Léxico**		
		Entrada	Salida	
Símbolo	**Definición de los símbolos**	**Lexema**	**Token**	
			cod-símb	*atributo(s)*
identificador	Cadena alfanumérica, con el primer carácter alfabético.	x25	id	$\text{ind}_{(x25)}$
const. entera	Cadena de uno o más dígitos	127	cte	$\text{val}_{(127)}$
op. relacional	$<$, $<=$, $>$, $>=$, $==$, $!=$	$>$	oprel	$\text{cod}_{(>)}$
op. asignación	$=$	$=$	opasig	
...	...	...	...	...

Token $\equiv$ (código del símbolo, atributo(s) del símbolo)

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

Expresiones Regulares (Especificación)

- Dado un alfabeto T, una Expresión Regular (ER) sobre T es:
  - $\emptyset$ es una ER y define el lenguaje $\emptyset$,
  - ϵ es una ER y define el lenguaje $\{\ \epsilon\ \}$,
  - $a \in T$ 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:
  - $r \mid s$ es una ER y define el lenguaje $L_r \cup L_s$,
  - $r \cdot s$ es una ER y define el lenguaje $L_r \cdot L_s$,
  - r^* es una ER y define el lenguaje L_r^*.
  - r^+ es una ER y define el lenguaje L_r^+.

Autómatas de Estados Finitos (Análisis)

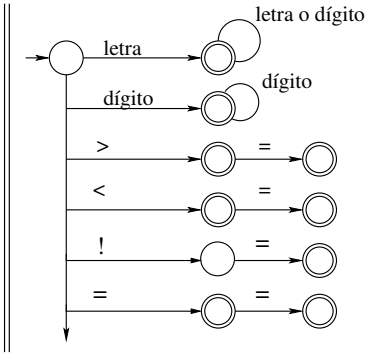
$AEF = (Q, T, \delta, q_0, F)$ Donde: $F \subseteq Q$; $q_0 \in Q$; $\delta : Q \times T \to \wp(Q)$

Equivalencia ER ⇔ AEF

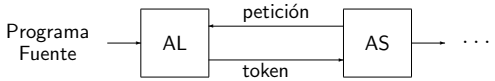
ER ⇔ AEFND con transiciones ϵ ⇔ AEFND ⇔ AEFD ⇔ AEFD mínimo

Ejemplo

letra	$a \mid b \mid \ldots \mid z$
dígito	$0 \mid 1 \mid \ldots \mid 9$
identificador	letra · (letra | dígito)*
cte. entera	dígito · (dígito)*
op. relacional	>|>=|<|<=|==|! =
op. de asignación	=



AL en el seno de un compilador



Funciones de un AL

1. Detección de los símbolos del lenguaje

   ➢ 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
   ```


   ➢ Detección de las palabras reservadas:
   - Palabras reservadas como expresiones regulares.
   - Palabras reservadas como identificadores especiales (tabla de palabras reservadas).

2. Realización de las acciones asociadas a la detección de un símbolo
   ➢ manipulación de la tabla de nombres,
   ➢ tratamiento de errores léxicos,
3. Emisión de los tokens
4. Otras
   ➢ 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.
   ➢ reconocimiento y ejecución de las directivas de compilación.

Generadores automáticos de AL: FLEX

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

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

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

  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.

Generadores automáticos de AL: LEX

```
letra            [a-zA-Z]
digito           [0-9]
identificador    {letra}({letra}|{digito})*
constante        {digito}({digito})*
%%
"<"              { return(MENOR_);      }
"<="             { return(MENORIG_);    }
">"              { return(MAYOR_);      }
">="             { return(MAYORIG_);    }
"=="             { return(IGUAL_);      }
"!="             { return(DISTINTO_);   }
"="              { return(ASIG_);       }
{identificador}  { ind(); return(ID_);  }
{constante}      { val(); return(CTE_); }
%%
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.              */
               {     ...    }
```