ICI 4242 - Autómatas y compiladores

Análisis Léxico

Rodrigo Olivares
Mg. en Ingeniería Informática
rodrigo.olivares@uv.cl

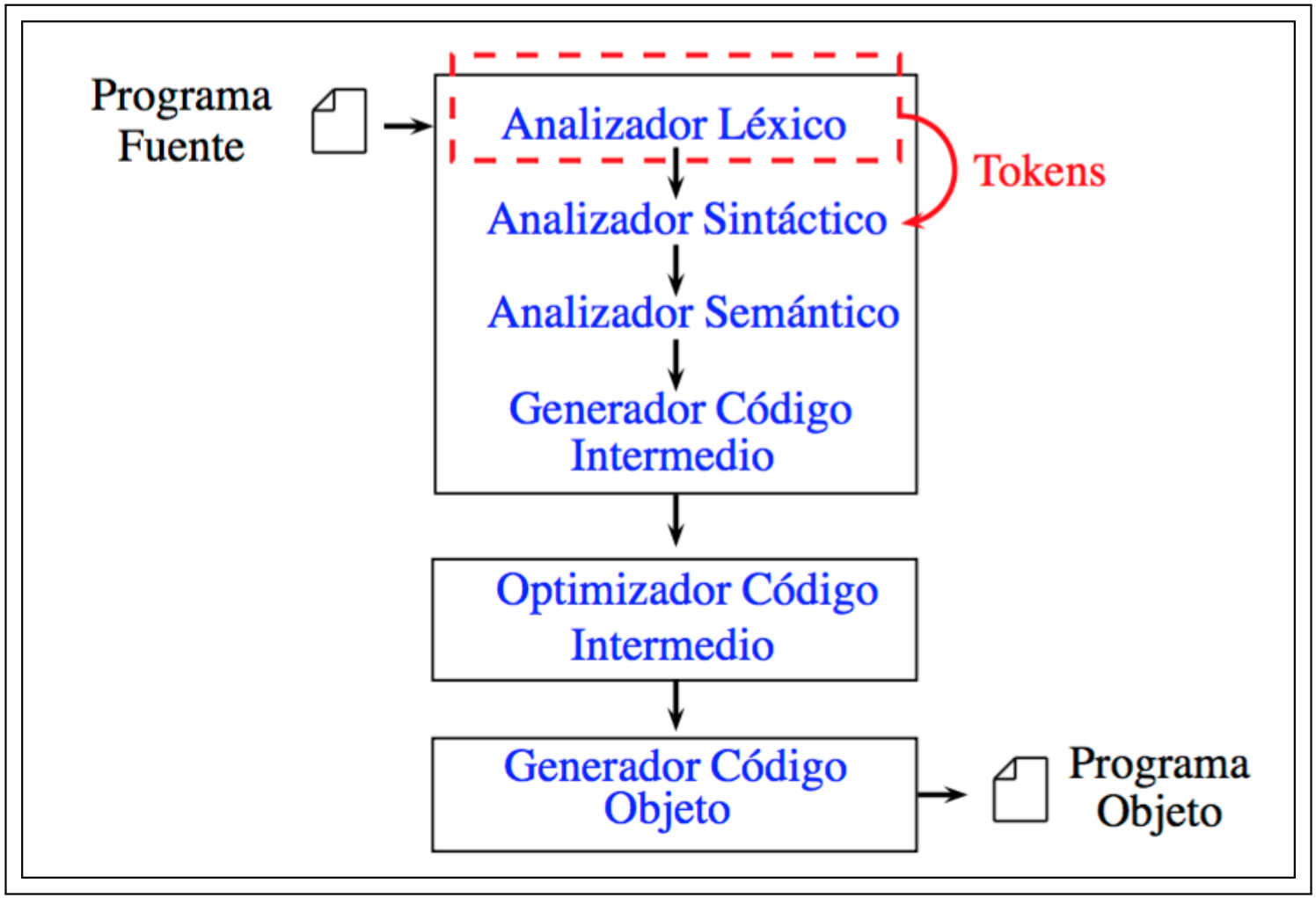
1er Semestre

Funciones del analizador léxico

Definición

Analizador léxico: *Corresponde a la primera fase de un compilador*. Es la encargada de recibir el programa fuente, reconocer caracter a caracter cada componente y luego agruparlos, para formar unidades con significado propio, los *componentes léxicos* (**tokens**).

Funciones del analizador léxico



Funciones del analizador léxico

Estos componentes léxicos representan:

- → Palabras reservadas: **if**, **while**, **do**, ...
- → Identificadores: asociados a variables, nombres de funciones, tipos de datos definidos por el usuario, etiquetas, ... Por ejemplo: **posicion**, **velocidad**, **tiempo**, ...
- → Operadores: $=$ $*$ $+$ $-$ $/$ $==$ $>$ $<$ $\&$ $!=$ ...
- → Símbolos especiales: ; ( ) [ ] # ...
- → Constantes numéricas: literales que representan valores enteros, punto flotante, etc. Por ejemplo: 982; $0xF678$; $-83{,}2E^{+2}$, ...
- → Constantes de caracteres: literales que representan cadenas concretas de caracteres. Por ejemplo: **"hola mundo"**,...

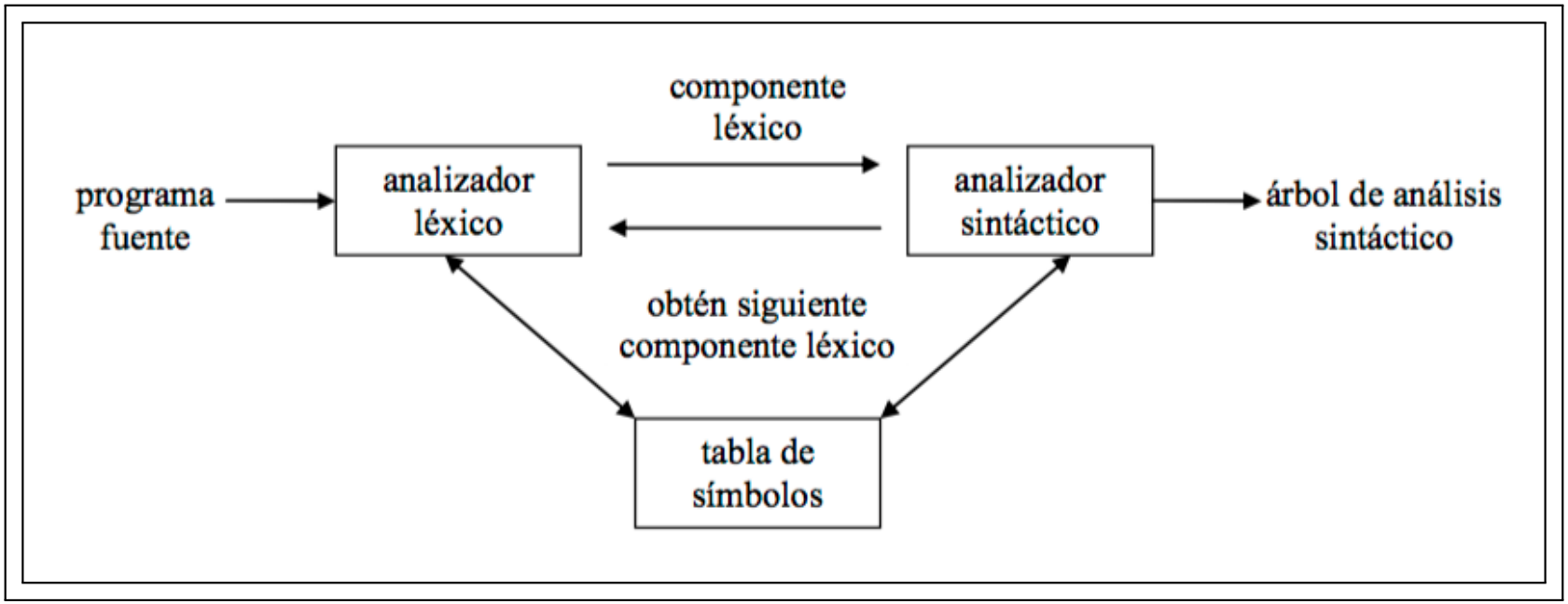
Funciones del analizador léxico

El analizador léxico opera bajo petición del analizador sintáctico, devolviendo un componente léxico conforme el analizador sintáctico lo va necesitando para avanzar en la gramática.

Los componentes léxicos son los símbolos terminales de la gramática. Suele implementarse como una subrutina del analizador sintáctico.

Cuando recibe la orden **obtén el siguiente componente léxico**, el analizador léxico lee los caracteres de entrada hasta identificar el siguiente componente léxico.

Funciones del analizador léxico



Funciones del analizador léxico

Otras funciones secundarias:

- → Manejo del archivo de entrada del programa fuente: abrirlo, leer sus caracteres, cerrarlo y gestionar posibles errores de lectura.
- → Eliminar comentarios, espacios en blanco, tabuladores y saltos de línea (caracteres no válidos para formar un **token**).
- → Inclusión de ficheros: *#include*, *import*, *required*, etc
- → Contabilizar el número de líneas y columnas para emitir mensajes de error.
- → Reconocimiento y ejecución de las directivas de compilación (por ejemplo, para depurar u optimizar el código fuente).

Funciones del analizador léxico

Patrón:

Es una regla que genera la secuencia de caracteres que puede representar a un determinado componente léxico (una expresión regular).

Lexema:

Cadena de caracteres que concuerda con un patrón que describe un componente léxico. Un componente léxico puede tener uno o infinitos lexemas. Por ejemplo: palabras reservadas tienen un único lexema. Los números y los identificadores tienen infinitos lexemas.

Funciones del analizador léxico

Componente léxico	**Lexema**	**Patrón**
identificador	indice, a, temp	letra seguida de letras o dígitos
num_entero	1492, 1, 2	dígito seguido de más dígitos
if	if	letra i seguida de letra f
do	do	letra d seguida de o
op_div	/	caracter /
op_asig	=	caracter =

Funciones del analizador léxico

Los componentes léxicos se suelen definir como un tipo enumerado. Se codifican como enteros. También se suele almacenar la cadena de caracteres que se acaba de reconocer (el lexema), que se usará posteriomenete para el análisis semántico.

```
typedef enum {
        TKN_IF,
        TKN_THEN,
        TKN_NUM,
        TKN_ID,
        TKN_OPADD,
        ⋮
} TokenType;
```

Funciones del analizador léxico

Es importante conocer el lexema (para construir la tabla de símbolos). Los componentes léxicos se representan mediante una estructura registro con tipo de token y lexema:

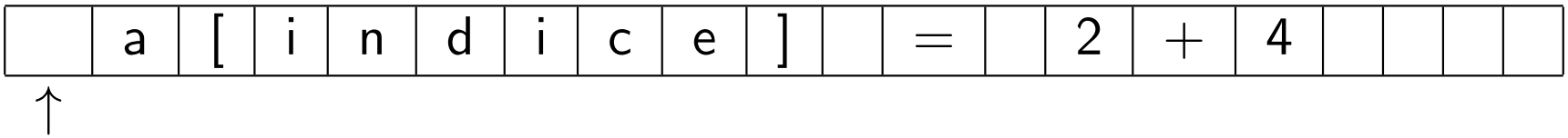
```
typedef struct {
    TokenType token;
    char *lexema; // se reserva memoria dinámicamente
} TokenRecord;
TokenRecord getToken(void);
```

Funciones del analizador léxico

Ejemplo

$a[indice] = 2 + 4$

buffer de entrada



Funciones del analizador léxico

Ejemplo: Cada componente léxico va acompañado de su lexema:

<TKN_ID, a>
<TKN_CORC_APER, [>
<TKN_ID, indice>
<TKN_CORC_CIER,] >
<TKN_NUM, 2>
<TKN_OP_ADD, + >
<TKN_NUM, 4 >

Especificación de los componentes léxicos.

Expresiones regulares

Los componentes léxicos se especifican haciendo uso de expresiones regulares. Además de las tres operaciones básicas: **concatenación**, **repetición** $(^*)$ y **alternativas/unión** $(|)$, se usarán los siguientes metasímbolos:

- **Una o más repeticiones** $+$
  - α^+ indica una o más repeticiones de α
  - $(0|1)^+ = (0|1)(0|1)^* \Leftrightarrow \alpha^+ = \alpha.\alpha^*$
- **Cualquier caracter .**
  - $.^*b.^*$ indica cualquier cadena que contiene una letra b
- **Cualquier carácter excepto un conjunto dado ~**
  - ~(a|b) indica cualquier caracter que no sea una a ó b
- **Opcionalidad ?**
  - α? indica que la expresión α puede aparecer o no. En el caso de que aparezca, sólo lo hará una vez.

Especificación de los componentes léxicos.

Expresiones regulares

Los componentes léxicos se especifican haciendo uso de expresiones regulares. Además de las tres operaciones básicas: **concatenación**, **repetición** (*) y **alternativas/unión** ($|$), se usarán los siguientes metasímbolos:

- **Un rango de caracteres** [ ] (**clase**)
  - [a-z] indica cualquier caracter entre la a y z **minúsculas**.
  - [a-zA-Z] indica cualquier letra del abecedario minúscula o mayúscula.
  - [0-9] indica cualquier dígito de 0 a 9.
  - [abc] indica $a|b|c$.

Uso

Cuando queremos usar estos símbolos con su significado tenemos que usar la barra de escape, [a\-z] significaría cualquier letra que sea **a**, **guión** ó **z**.

Especificación de los componentes léxicos.

Expresiones regulares

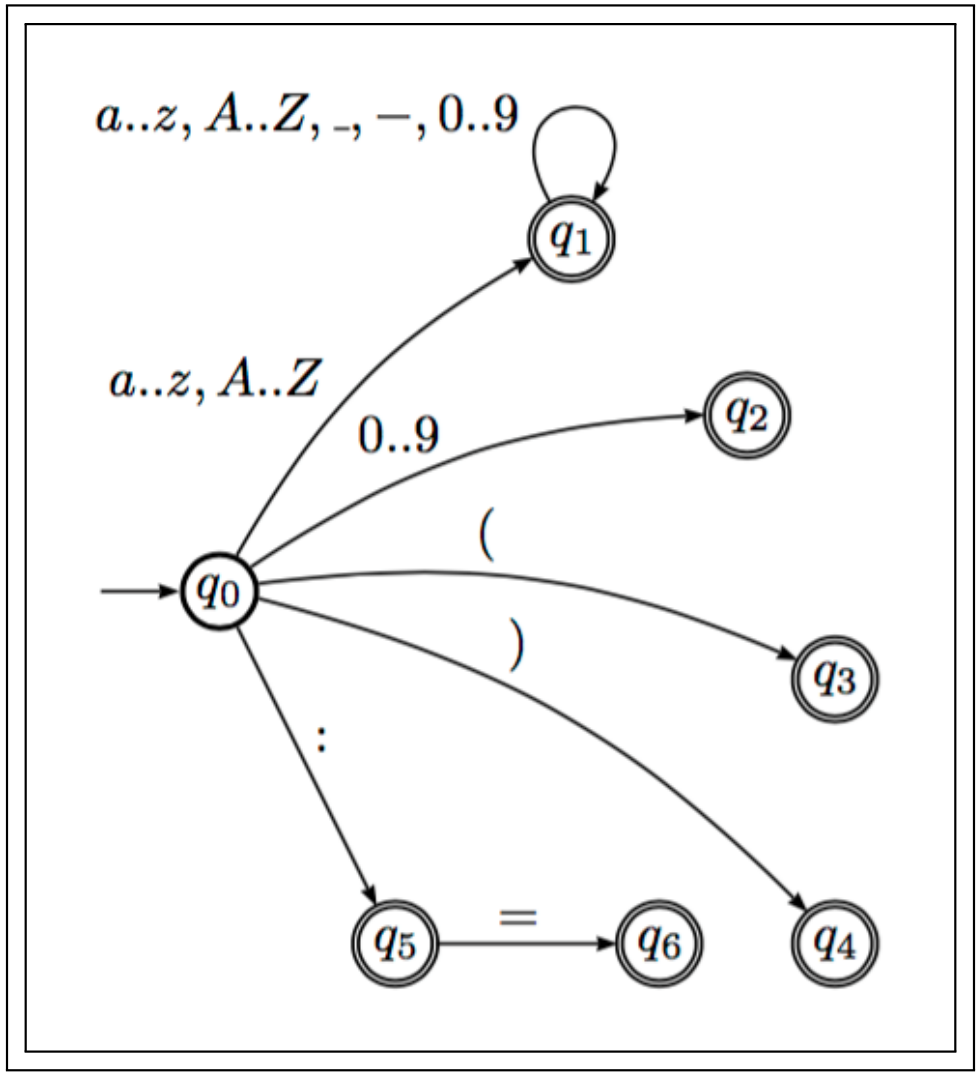
Ejemplos

- Números
  - nat $= [0-9]^+$
  - signedNat $= (+|-)?$ nat
  - number $=$ signedNat(“.”nat)? (E signedNat)?
- Identificadores
  - letter $=$ [a-zA-Z]
  - digit $=$ [0-9]
  - identifier $=$ letter (letter $|$ digit)*

  Palabras Reservadas
  - TKN_IF $=$ “if”
  - TKN_WHILE $=$ “while”
  - TKN_DO $=$ “do”

Especificación de los componentes léxicos.

Autómata Finito Determinista



Preguntas

Preguntas ?