



PROCESADORES DE LENGUAJE

Ingeniería Informática
Especialidad de computación
Tercer curso, segundo cuatrimestre



Departamento de Informática y Análisis Numérico
Escuela Politécnica Superior de Córdoba
Universidad de Córdoba

Curso académico 2018 - 2019

Hoja de ejercicios nº 1.- ANÁLISIS LÉXICO

Alfabetos, palabras y lenguajes formales

1. Obtén las palabras de longitudes 1, 2 y 3 de los siguientes alfabetos:
 $\Sigma_1 = \{a\}$, $\Sigma_2 = \{0, 1\}$, $\Sigma_3 = \{if, then, else\}$, $\Sigma_4 = \{a, b, c, d\}$
2. Indica el lenguaje universal Σ^* que genera el alfabeto $\Sigma = \{a, b, 1, 2\}$
3. Dado el alfabeto $\Sigma = \{a, b, c\}$:
 - Se definen las siguientes palabras
 - $x = ab$, $y = bba$, $z = cb$
 - Realiza las siguientes operaciones
 - xy , yx , $x(yz)$, $x \varepsilon z$, $z y^0 x$, $x^2 y$, z^2 , $(x y)^2$
4. Dado el alfabeto $\Sigma = \{a, b, c\}$:
 - Se definen los siguientes lenguajes formales
 - $L_1 = \{a, aa, aaa\}$, $L_2 = \{a, b, ba, bc\}$ y $L_3 = \{\varepsilon, a, b, c\}$
 - Realiza las siguientes operaciones
 - $L_1 \cup L_2$, $L_1 \cap L_2$, $L_1 L_2$, $L_1 - L_2$, $L_2 - L_1$
 - $(L_1)^2$
 - L_1^*
 - L_3^+
 - $L_1 (L_2 \cup L_3)$
 - $(L_3 - L_1) \cap (L_1 L_3)$
5. Dado el alfabeto $\Sigma = \{a, b\}$
 - Indica dos lenguajes L_1 , L_2 que verifiquen que
 - $(L_1 \cup L_2)^* \neq L_1^* \cup L_2^*$
 - Indica dos lenguajes L_1 , L_2 que verifiquen que
 - $L_1 \not\subset L_2$, $L_2 \not\subset L_1$ y $(L_1 \cup L_2)^* = L_1^* \cup L_2^*$
 - Indica tres lenguajes L_1 , L_2 y L_3 de forma que
 - $L_3 (L_2 - L_1) \neq L_3 L_2 - L_3 L_1$
6. Sea $\Sigma = \{a, b\}$ y $L \subseteq \Sigma^*$ es un lenguaje definido recursivamente de la siguiente forma:

- a) $\varepsilon \in L$
 - b) Si $x \in L$, entonces $a x b, b x a \in L$
 - c) Si $x, y \in L$, entonces $x y \in L$
 - d) No hay nada más en L
 - Demuestra que:
 - $L = \{ w \mid w \in \Sigma^* \wedge w \text{ contiene el mismo número de } a \text{es que de } b \text{es} \}$
 - Si $b, \varepsilon \in L$ ¿Qué más palabras hay en L ?
 - Da una definición recursiva para que $L' \subseteq \Sigma^*$ contenga todas las palabras que posean doble número de a es que de b es.
7. Si $\text{cardinal}(L)$ nos indica cuantas palabras posee un lenguaje, comprueba si es cierta o falsa la siguiente afirmación:
- $\text{cardinal}(L_1 L_2) = \text{cardinal}(L_1) \text{cardinal}(L_2)$.
- Si se cree que es falsa, póngase un contraejemplo; si se cree que es verdadera, demuéstrese.

Expresiones regulares

8. Indica algunas expresiones regulares que se puedan definir sobre el siguiente alfabeto $\Sigma = \{a, b, 1, 2\}$
9. Indica cuál es el lenguaje denotado por las siguientes expresiones regulares definidas sobre $\Sigma = \{a, b\}$:
- a) $a a^* b b^*$
 - b) $a (a^* + b^*) b$
 - c) $a (a + b)^* b$
10. Dado el alfabeto $\Sigma = \{-, ., _, a, b, c, \dots, z, A, B, \dots, Z, 0, 1, \dots, 9\}$ y las siguientes definiciones regulares
- $\text{dígito} = 1 + 2 + \dots + 9$
 - $\text{cero} = 0$
 - $\text{numero} = \text{cero} + \text{dígito}$
 - $\text{punto} = .$
 - $\text{guión} = -$
 - $\text{subrayado} = _$
 - $\text{letra} = a + b + \dots + z + A + B + \dots + Z$
 - Indica cuál es el lenguaje denotado por cada una de las siguientes expresiones regulares:
 - a) $\text{numero numero}^* \text{ punto numero}^*$
 - b) $\text{numero}^* (\text{ punto } + \varepsilon)$
 - c) $(\text{cero} + \text{dígito numero}^*) \text{ punto } (\text{cero} + \text{numero}^* \text{ dígito})$
 - d) $\text{letra} (\text{letra} + \text{numero})^* \text{ numero}$
11. Define expresiones regulares que denoten los siguientes lenguajes formales definidos sobre el alfabeto $\Sigma = \{0, 1\}$:
- a) $L_1 = \{ x \mid x \in \Sigma^* \wedge x \text{ sólo contiene dos ceros y un número indefinido de unos} \}$
 - b) $L_2 = \{ x \mid x \in \Sigma^* \wedge x \text{ contiene al menos dos ceros consecutivos} \}$

- c) $L_3 = \{ x \mid x \in \Sigma^* \wedge x \text{ contiene un número impar de ceros y un número indefinido de unos} \}$
- d) $L_4 = \{ x \mid x \in \Sigma^* \wedge \text{ cada cero de } x \text{ es seguido inmediatamente por } 11 \}$
12. Escribe expresiones regulares que denoten los siguientes lenguajes definidos sobre $\Sigma = \{a, b, c\}$
- Palabras que comienzan y finalizan con la letra "a".
 - Palabras que comienzan o finalizan con la letra "a" (o ambas posibilidades).
 - Palabras en las que la "a", si aparece, siempre precede a la "b".
 - Palabras que tengan un número impar de a's.
13. Define expresiones regulares que denoten los siguientes lenguajes definidos recursivamente:
- $\varepsilon \in L$. Si $x \in L$, entonces **aax** y **xb** son palabras de L. Sólo están en L las palabras obtenidas mediante las premisas anteriores.
 - $\varepsilon \in L$. Si $x \in L$, entonces **abx**, **bax**, **aax** y **bbx** son palabras de L. Sólo están en L las palabras obtenidas mediante las premisas anteriores.
14. Escribe expresiones regulares que denoten los siguientes lenguajes:
- Números naturales que no contengan dos o más ceros al principio: 0, 10, 121,...
 - Números pares.
 - Números impares.
 - Números reales con formato de punto fijo o con formato de punto flotante pero que no tengan ceros superfluos, es decir,
 - son permitidos los números del tipo 0.0, 132.0, 0.526, 1203.0494,
 - pero no son permitidos los números de la forma 00.12, 124.000, 001.727, 52.700.
15. Considera el siguiente código escrito en el lenguaje C que implementa el método de ordenación de Shell:
- ```
#include <stdio.h>
#include <malloc.h>
#include "macros.h"
/* Longitud maxima -1 de los nombres */
#define NUMERO_CARACTERES 20

void shell (struct ficha_persona *dato, int n)
{
 int d,i, bandera;
 struct ficha_persona auxiliar;
 d = n;
 do {
 d = d / 2;
 do{
 bandera = 0;
 i = 0;
 do {
 if (dato[i].edad > dato[i+d].edad)
 {
 strcpy(auxiliar.nombre, (dato+i)->nombre);
```

```

 auxiliar.edad = (dato+i)->edad;
 strcpy((dato+i)->nombre, (dato+i+d)->nombre);
 (dato+i)->edad = (dato+i+d)->edad;
 strcpy((dato+i+d)->nombre, auxiliar.nombre);
 (dato+i+d)->edad = auxiliar.edad;
 bandera = 1;
 }
 i++;
} while (i+d <= n-1);
} while (bandera !=0);
} while (d!=1);
}

```

- Indica los diferentes tipos de componentes léxicos o “tokens” que generaría el analizador léxico.
- Define las expresiones regulares que denotan los tipos de componentes léxicos indicados en el apartado anterior.

16. Considera el siguiente código escrito en el lenguaje FORTRAN

```

INTEGER I, J
REAL Vector(10), Matriz(5,5)

PRINT *, 'INTRODUCE LAS COMPONENTES IMPARES DEL VECTOR'
DO 10 I = 1, 9, 2
 PRINT *, 'COMPONENTE ', I, ' -->'
 READ *, Vector (I)
10 CONTINUE

PRINT *, 'INTRODUCE LAS COMPONENTES DE LA DIAGONAL PRINCIPAL'
DO 20 I = 1, 5
 PRINT *, 'COMPONENTE (', I, ', ', I, ') -->'
 READ *, Matriz (I,I)
20 CONTINUE

```

- Indica los diferentes tipos de componentes léxicos que reconocería el analizador léxico.
- Escribe las expresiones regulares correspondientes a dichos tipos de componentes léxicos.

17. Indica las expresiones regulares que denoten los componentes léxicos de un lenguaje de programación en pseudocódigo:

- **Identificadores:**
  - Podrán estar compuestos por letras, números y el símbolo “\_”.
  - Podrán comenzar por una letra o el símbolo “\_”
  - El símbolo “\_” no podrá aparecer al final.
- **Números:**
  - Se podrán definir números enteros (19), reales de punto fijo (19.75) o con notación exponencial (0.19e+2).
- **Cadenas de caracteres:**
  - Estarán compuestas por cualquier carácter excepto las comillas simples de apertura (‘) y cierre (’), que deberán aparecer al principio y al final, respectivamente.
  - Se utilizará la barra invertida \ para poder introducir las comillas simples dentro de las cadenas.
- **Palabras reservadas:**

- Se podrán escribir con letras mayúsculas o minúsculas o ambas
- Deberán comenzar y terminar por el símbolo de subrayado “\_”.
- Por ejemplo: mientras
- **Operadores:**
  - Asignación:
    - se utilizará el operador de Pascal ( := )
  - Lógicos:
    - Estarán delimitados por dos símbolos de subrayado
    - Por ejemplo: no
  - Aritméticos:
    - Se utilizarán como operadores las tres “primeras letras” de cada una de las operaciones aritméticas (suma, resta, multiplicación, división y potencia) pero delimitadas por el símbolo “\_”
    - Por ejemplo: pot
  - Relacionales:
    - Se utilizarán los símbolos empleados por el lenguaje C, excepto los operadores “igual” y “distinto” que se utilizarán los empleados por Pascal (=, <>).

### Autómatas finitos

18. Dados los siguientes autómatas finitos deterministas:

- Dibuja la representación gráfica de cada uno.
- Comprueba si reconocen o no las palabras que se indican en cada caso, mostrando las transiciones que se vayan produciendo “paso a paso”.
- Indica de manera informal cómo es el lenguaje que reconoce cada autómata.
- Define una expresión regular que denote el lenguaje que reconocería cada autómata.

a) AFD1

| $\delta$          | <i>a</i> | <i>b</i> | <i>c</i> |
|-------------------|----------|----------|----------|
| $\rightarrow q_0$ | $q_1$    | $q_3$    | $q_3$    |
| $q_1$             | $q_1$    | $q_2$    | $q_2$    |
| $\leftarrow q_2$  | $q_3$    | $q_2$    | $q_2$    |
| $q_3$             | $q_3$    | $q_3$    | $q_3$    |

- $x = aabcc$
- $y = abca$

b) AFD2

| $\delta$          | <i>d</i> | <i>o</i> | <i>p</i> |
|-------------------|----------|----------|----------|
| $\rightarrow q_0$ | $q_1$    | $q_2$    | -        |
| $\leftarrow q_1$  | $q_1$    | $q_1$    | $q_3$    |
| $\leftarrow q_2$  | -        | -        | $q_3$    |
| $q_3$             | $q_4$    | $q_5$    | -        |
| $\leftarrow q_4$  | $q_4$    | $q_6$    | -        |
| $\leftarrow q_5$  | $q_4$    | $q_6$    | -        |
| $q_6$             | $q_4$    | $q_6$    | -        |

donde  $d \in \{1, 2, 3, 4, 5, 6, 7, 8, 9\}$  y  $p$  es el punto decimal ".".

- $x_1 = 0.79$
- $x_2 = 012.0$
- $x_3 = 12.0$
- $x_4 = 10203$
- $x_5 = 45.600$
- $x_6 = 0.0$

19. Dados los siguientes autómatas finitos **no** deterministas:

- **Dibuja** la representación gráfica de cada uno.
- Comprueba si reconocen o no las cadenas que se indican en cada caso, mostrando las transiciones que se vayan produciendo "paso a paso".

a) AFN sin transiciones épsilon no triviales

| $\delta$          | $a$            | $b$            | $c$            |
|-------------------|----------------|----------------|----------------|
| $\rightarrow q_0$ | $\{q_1, q_2\}$ | $\phi$         | $\phi$         |
| $q_1$             | $\phi$         | $\{q_1, q_2\}$ | $\{q_3, q_4\}$ |
| $q_2$             | $\phi$         | $\{q_1, q_2\}$ | $\{q_3, q_4\}$ |
| $\leftarrow q_3$  | $\phi$         | $\phi$         | $\{q_3, q_4\}$ |
| $\leftarrow q_4$  | $\phi$         | $\phi$         | $\{q_3, q_4\}$ |

$x = abbcc$ ,  $y = abcb$

b) AFN con transiciones épsilon no triviales

| $\delta$          | $a$       | $b$            | $c$                 | $\epsilon$          |
|-------------------|-----------|----------------|---------------------|---------------------|
| $\rightarrow q_0$ | $\{q_1\}$ | $\{q_2, q_3\}$ | $\phi$              | $\{q_0, q_2\}$      |
| $q_1$             | $\phi$    | $\{q_3, q_4\}$ | $\{q_1, q_2, q_3\}$ | $\{q_2\}$           |
| $\leftarrow q_2$  | $\{q_3\}$ | $\{q_1, q_2\}$ | $\phi$              | $\{q_3\}$           |
| $q_3$             | $\{q_0\}$ | $\{q_2\}$      | $\{q_4\}$           | $\{q_4\}$           |
| $\leftarrow q_4$  | $\phi$    | $\{q_2, q_3\}$ | $\{q_1, q_4\}$      | $\{q_1, q_2, q_4\}$ |

$x = aabc$ ,  $y = bbacab$

20. Algoritmo de "construcción de subconjuntos"

- a) Aplica el algoritmo de "construcción de subconjuntos" a los autómatas finitos no deterministas del ejercicio anterior.
- b) Comprueba si los autómatas finitos deterministas construidos reconocen las cadenas propuestas en dicho ejercicio.

21. Dadas las siguientes expresiones regulares

- *letra* (*letra* + *dígito*) \*
- (*letra* + *subrayado*) (*letra* + *subrayado* + *dígito*)\*
- *letra* (*letra* + *dígito* + *guion* (*letra* + *dígito*))\*
- *comillas* (*letra* + *dígito* + *barra comillas*)\* *comillas*

donde *letra*  $\in \{a, \dots, z, A, \dots, Z\}$ , *dígito*  $\in \{0, 1, \dots, 9\}$ , *subrayado* es el símbolo '\_', *guion* es el símbolo '-', *comillas* es el símbolo '"' y *barra* es símbolo '\'.

- Utiliza el algoritmo de "construcción de Thompson" para construir los autómatas finitos **no** deterministas equivalentes.
- Utiliza el algoritmo de "construcción de subconjuntos" para construir los autómatas finitos **deterministas** equivalentes a los obtenidos en el apartado anterior.
- Minimiza** los autómatas finitos deterministas.
- Comprueba si los autómatas finitos construidos en los apartados anteriores reconocen, respectivamente, las siguientes cadenas:
  - $x = \text{dato}, y = \text{dato1}, z = 1\text{dato}$
  - $x = \text{dato}, y = \text{dato\_1}, z = \_\text{dato\_}1$
  - $x = \text{dato-1}, y = \text{dato--1}, z = \text{dato1-1}$
  - $x = \text{"ejemplo de \"cadena\" "}$

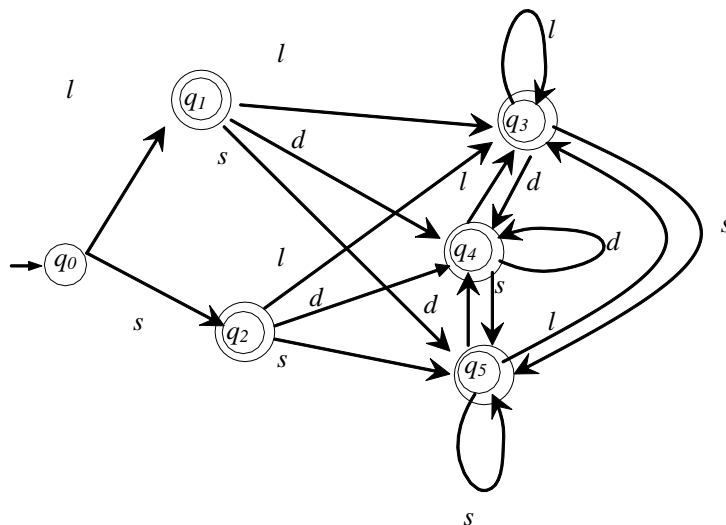
22. Dado el siguiente autómata finito determinista

| $\delta$         | <i>l</i>  | <i>d</i>  | <i>g</i>  |
|------------------|-----------|-----------|-----------|
| $\rightarrow q0$ | <i>q1</i> | -         | -         |
| $\leftarrow q1$  | <i>q2</i> | <i>q3</i> | <i>q4</i> |
| $\leftarrow q2$  | <i>q2</i> | <i>q3</i> | <i>q4</i> |
| $\leftarrow q3$  | <i>q2</i> | <i>q3</i> | <i>q4</i> |
| $\leftarrow q4$  | <i>q2</i> | <i>q3</i> | -         |

donde el significado de *l*, *d* y *g* es el siguiente: *l* = letra, *d* = dígito y *g* = guion

- Dibuja su representación gráfica.
- Minimiza** el autómata mediante la obtención del autómata cociente.

23. Dado el siguiente autómata finito determinista:



donde el significado de **l**: letra, **d**: dígito y **s**: subrayado.

- Comprueba si reconoce o no la cadena  $x = \text{ldsl}$
- Minimiza el autómata finito determinista.
- Comprueba si el autómata minimizado reconoce o no la cadena  $x = \text{ldsl}$

### Errores léxicos

24. El siguiente código escrito en lenguaje **C** calcula el factorial de un número, pero tiene “diez errores léxicos”.

```

[1] int factorial (int n
[2]
[3] Int i;
[4] int re$ultado;
[5]
[6] nif ((n==0) || (n==1))
[7] return (1.0.0);
[8] else {
[9] resultado := 1;
[10] for (i===n ; i > 1 ; i--)
[11] resultado **= i;
[12]
[13] return (resultado)
[14] }
[15] }
```

- Indica los errores léxicos que se **pueden detectar** durante el análisis léxico
- Indica los errores léxicos que se **pueden detectar** durante el análisis sintáctico

- Observación: se indican los números de las líneas para facilitar la identificación de cada error



25. Considera el siguiente fragmento de código erróneo escrito en C

```
[1] fooor (i = N ; > i 1.0.0 ; i--)
[2] {
[3] factorial = factorial * $n;
[4] }
[5] printf " Factorial = %d " /n, $factorial);
```

- a) Indica los errores que **puede** detectar “el analizador léxico” y los que **no** puede detectar y “**por qué**”.
- b) Indica los componentes léxicos que reconocería el “analizador léxico”.
- c) Escribe las expresiones regulares correspondientes a los diferentes tipos de componentes léxicos reconocidos en el fragmento anterior.
  - Observación: se indican los números de las líneas para facilitar la identificación de cada error