



# UNIVERSIDAD DE GRANADA

**Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación**

## Prácticas 1 y 2

Asignatura: Modelos de Computación

Año: 2022/2023

Grupo: MC\_B3

A 12 de enero de 2023

Linqi Zhu

Lorena Gómez Gómez

# 1. Relación de problemas de práctica 1

a.  $\{a^n \in \{a,b\}^* \text{ con } n \geq 0\} \cup \{a^n b^n \in \{a,b\}^* \text{ con } n \geq 0\}$

Este lenguaje generará cadenas como: a, ab, aab, aaaabbbb... y rechazará otras como: b, abb, bbb...

La gramática puede ser:

$S \rightarrow AB$

$A \rightarrow aA$

$B \rightarrow aBb$

$B \rightarrow \epsilon$

$A \rightarrow \epsilon$

Comprobación de la gramática con JFlap:

The screenshot shows the JFlap software interface. At the top, there are buttons for 'Start', 'Pause', and 'Step', and a dropdown menu set to 'Noninverted Tree'. Below this, the 'Input' field contains 'aab', and a message states 'String accepted! 18 nodes generated.' and 'Input Field Text Size (For optimization, move one of the window size adjusters arou...'. A 'Table Text Size' slider is visible. The main area displays a parse tree for the input 'aab'. The root node is 'S', which has two children: 'A' and 'B'. Node 'A' has two children: 'a' and 'A'. The 'A' child of 'A' has a single child 'λ'. Node 'B' has three children: 'a', 'B', and 'b'. The 'B' child of 'B' has a single child 'λ'. To the left of the tree is a table with LHS and RHS columns, showing the grammar rules: S → AB, A → aA, B → aBb, B → λ, and A → λ. At the bottom, a status bar says 'Derived λ from B. Derivations complete.' On the right side, there is a 'Table Text Size' slider and a table with 'Input' and 'Result' columns. The table contains the following data:

Input	Result
aaaabbbb	Accept
aaabbb	Accept
aaaabbbb	Accept
aab	Accept
ab	Accept
a	Accept
b	Reject
abb	Reject
aaabbbb	Reject

At the bottom right, there are buttons for 'Load Inputs', 'Run Inputs', 'Clear', and 'Enter Lambda'.

b.  $\{ucv \in \{a, b, c\}^* \text{ tales que } u \text{ y } v \text{ tienen la misma longitud}\}$

Este lenguaje generará cadenas como: acb, abcab, bbcaa..... y rechazará otras como: a, abb, aaa, abcc.....

La gramática puede ser:

$S \rightarrow c \quad S \rightarrow ASA \quad A \rightarrow a \quad A \rightarrow b \quad A \rightarrow c$

Comprobación de la gramática con JFlap:

The screenshot shows the JFlap software interface for testing a grammar. The window title is "Noninverted Tree".

**Control Bar:** Start, Pause, Step, Noninverted Tree (dropdown), and a blue arrow icon.

**Input Section:**

- Input: abccab
- String accepted! 57 nodes generated.
- Input Field Text Size (For optimization, move one of the window size adjustors around this window afte...)

**Table Text Size:** A slider control.

**Grammar Rules Table:**

LHS	RHS
S	→ c
S	→ ASA
A	→ a
A	→ b
A	→ c

**Tree Diagram:** A noninverted tree structure. The root node is S (green). It has three children: A (green), S (green), and A (green). The leftmost A has a single child 'a' (yellow). The middle S has three children: A (green), S (green), and A (green). The leftmost A has a single child 'b' (yellow). The middle S has a single child 'c' (yellow). The rightmost A has a single child 'a' (yellow). The rightmost A has a single child 'b' (yellow).

**Results Table:**

Input	Result
acb	Accept
abcab	Accept
bbcaa	Accept
a	Reject
abb	Reject
abcc	Reject

**Footer:** Derived b from A. Derivations complete. Load Inputs, Run Inputs, Clear, Enter Lambda

c.  $\{uv \in \{0,1\}^* \text{ tales que } u^{-1} \text{ es un prefijo de } v\}$

Esta gramática generará cadenas que tengan un palíndromo al principio de ellas.

Por ejemplo: 011010, 0011, 11110... etc

Rechazará cadenas del tipo: 1011, 01, 11101

Las reglas de producción de la gramática son muy parecidas a las que hicimos en clase para el ejemplo de los palíndromos, solo que al final se le añade una variable para que la cadena contenga más caracteres.

$S \rightarrow 0S0A$      $S \rightarrow 1S1A$      $A \rightarrow 0AA \rightarrow 1AS \rightarrow \lambda$   
 $A \rightarrow \lambda$

Ejecución con JFLAP

Start Pause Step Noninverted Tree

Input: 011010  
String accepted! 33 nodes generated.  
 Input Field Text Size (For optimization, move one of the window size adjusters around this window...)

Table Text Size

LHS	RHS
S	$\rightarrow \lambda$
S	$\rightarrow 1S1A$
S	$\rightarrow 0S0A$
A	$\rightarrow 0A$
A	$\rightarrow 1A$
A	$\rightarrow \lambda$

Table Text Size

	S
S $\rightarrow$ 0S0A	0S0A
A $\rightarrow$ 1A	0S01A
S $\rightarrow$ 1S1A	01S1A01A
A $\rightarrow$ 0A	01S1A010A
S $\rightarrow$ lambda	011A010A
A $\rightarrow$ lambda	011010A
A $\rightarrow$ lambda	011010

Table Text Size

Input	Result
011010	Accept
1001110	Accept
0101101	Reject
1001	Accept
011	Reject
111	Accept

d.  $\{u1^n \in \{0, 1\}^* \text{ donde } |u| = n\}$

Este lenguaje generará cadenas como: 01, 0011, 1011, 1111 y rechazará otras como: 111, 1100, 11011.

La gramática puede ser:

$S \rightarrow \lambda$        $S \rightarrow A1$     $A \rightarrow 0B$     $A \rightarrow 1B$        $B \rightarrow \lambda$        $B \rightarrow A1$

Comprobación de la gramática con JFlap:

Start Pause Step Noninverted Tree

Input

Input Field Text Size (For optimization, move ...)

Table Text Size

Input	Result
01	Accept
11	Accept
1111	Accept
0011	Accept
1011	Accept
0000011111	Accept
111111111111	Accept
111111111110	Reject
111	Reject
1100	Reject
11011	Reject
111011	Reject

LHS	RHS
S	$\rightarrow \lambda$
S	$\rightarrow A1$
A	$\rightarrow 0B$
A	$\rightarrow 1B$
B	$\rightarrow \lambda$
B	$\rightarrow A1$

Input 1011

String accepted! 7 nodes generated.

Input Field Text Size (For optimization, move one of the window size adjusters around this window after resizing t...)

Table Text Size

LHS	RHS
S	$\rightarrow \lambda$
S	$\rightarrow A1$
A	$\rightarrow 0B$
A	$\rightarrow 1B$
B	$\rightarrow \lambda$
B	$\rightarrow A1$

Derived λ from B. Derivations complete.

- e. Palabras con 0's y 1's que no contengan dos 1's consecutivos y que empiecen por un 1 y terminen por dos 0's.

Esta gramática genera cadenas como por ejemplo: 10100100, 100, 1010100

Sin embargo rechazará otras como por ejemplo: 01, 110100, 111

Unas posibles reglas de producción para esta gramática podrían ser:

$S \rightarrow 1A$        $A \rightarrow 0B$   $B \rightarrow 1A$        $B \rightarrow 0B$        $B \rightarrow \lambda$

La primera regla de producción garantiza que la cadena empiece por 1.

La segunda regla en combinación con la primera y la tercera, garantizan que no pueda haber dos 1's seguidos.

Por último las dos últimas se encargan de que sea posible acabar con dos 0's.

Ejecución en JFLAP:

Input: 100100  
String accepted! 11 nodes generated.

Input Field Text Size (For optimization, move one of the window size adjusters around this window after resizing t...)

LHS	RHS
S	$\rightarrow 1A$
A	$\rightarrow 0B$
B	$\rightarrow 1A$
B	$\rightarrow 0B$
B	$\rightarrow \lambda$

Table Text Size

	S
S $\rightarrow$ 1A	1A
A $\rightarrow$ 0B	10B
B $\rightarrow$ 0B	100B
B $\rightarrow$ 1A	1001A
A $\rightarrow$ 0B	10010B
B $\rightarrow$ 0B	100100B
B $\rightarrow$ $\lambda$	100100

Derived  $\lambda$  from B. Derivations complete.

Table Text Size	
Input	Result
100110	Reject
0100100	Reject
10100100	Accept
1000	Accept
101000	Accept

## 2. Analizador léxico práctica 2

Descripción del analizador léxico realizado:

El analizador léxico que hemos realizado consiste en un programa que analiza un archivo html de la página web : <https://www.eltiempo.es/granada.html>.

(Donde pone *granada*, podría ir cualquier ciudad de España).

Obtiene información sobre el tiempo de la ciudad de la cual hemos descargado dicho archivo html, por ejemplo muestra la temperatura actual, probabilidad de lluvia, tiempo actual y viento actual.

Además recopila información tanto de las temperaturas máximas y mínimas como de la previsión de viento de los próximos 15 días.

Para ello hemos elaborado dos ficheros diferentes:

descarga.py: para descargar el archivo html ( hecho en python)

tiempo.lex: analizador léxico que trabaja sobre el fichero html descargado (hecho en lenguaje C)

Para obtener la información de los ficheros de entrada y almacenarla en las estructuras de datos para luego operar con los datos hacemos uso de las siguientes expresiones regulares:

C/C++

```
letra [a-zA-Z]
digito [0-9]
numero {digito}*
simboloT [°C]
medidaViento [km/h]
Ciudad ("niv4:")+" "+({letra}+" "*)+" "
Provincia ("province:")+" "+({letra}+" "*)+" "
TemperaturaActual ("currentTemperature:")+" "+{numero}+{simboloT}+" "
TiempoActual ("weatherForecast:")+" "+({letra}+" "*)+" "
ProbPrecipitacionActual ("precipitationProbability:")+" "+{numero}+" "
VientoActual ("windSpeed:")+" "+{numero}+" "+{medidaViento}+" "
exp_temp_max_tag ("<span class=\"m_table_weather_day_max_temp\">")
exp_temp_min_tag ("<span class=\"m_table_weather_day_min_temp\">")
exp_temp_tag ("<span data-temp=\"\"{numero}+\" \" data-temp-include-units=\"\">")
exp_tag_viento ("<span data-wind=\"\"{numero}+\" \"
data-wind-include-units=\"1\">")
exp_tag_lluvia ("<span>\"{numero}+\".\"?\", \"?{numero}+\" mm")
```

## Ejemplo de uso:

```
PROBLEMS  OUTPUT  TERMINAL  DEBUG CONSOLE  COMMENTS
z07@cvi103220 entrega % ls
descargar.py tiempo.lex
z07@cvi103220 entrega % python3 descargar.py
Introducir la ciudad:granada
Descarga con éxito. Nombre de archivo: granada .html
z07@cvi103220 entrega % ls
descargar.py granada.html tiempo.lex
z07@cvi103220 entrega % flex tiempo.lex
z07@cvi103220 entrega % ls
descargar.py granada.html lex.yy.c tiempo.lex
z07@cvi103220 entrega % gcc lex.yy.c
z07@cvi103220 entrega % ls
a.out descargar.py granada.html lex.yy.c tiempo.lex
z07@cvi103220 entrega % ./a.out < granada.html
Información obtenida de la web:
Ciudad: Granada
Provincia: Granada
Temperatura actual: 17°C
Probabilidad de lluvia actual: 0 %
Tiempo actual: Poco nublado
Viento: 6 kmh

-----
Prevision quincenal de temperaturas (max y min)
15 oC 14 oC 15 oC 14 oC 10 oC 9 oC 8 oC 7 oC 9 oC 10 oC 11 oC 10 oC 11 oC 11 oC
4 oC 3 oC 3 oC 4 oC 1 oC 6 oC 3 oC 2 oC 2 oC 3 oC 3 oC 3 oC
-----
Prevision quincenal de viento
5 km/h 7 km/h 5 km/h 4 km/h 4 km/h 3 km/h 2 km/h 2 km/h 8 km/h 8 km/h 8 km/h 6 km/h 4 km/h 7 km/h
z07@cvi103220 entrega %
```

## Código del problema:

### descargahtml.py

Unset

```
import urllib.request

def getHtml(url):
    html = urllib.request.urlopen(url).read()
    return html

def saveHtml(file_name, file_content):
    # Preste atención a los caracteres prohibidos en los nombres
    # de archivos de Windows, como /
    with open(file_name.replace('/', '_') + ".html", "wb") as f:
        # Escribir archivos con bytes en lugar de str, por lo
        # que se requiere transcodificación
        f.write(file_content)

s=input('Introducir la ciudad:')
s = s.replace(" ", "-")
s = s.replace("ñ", "n")

aurl = "https://www.eltiempo.es/" + s + ".html"
html = getHtml(aurl)
saveHtml(s,html)

print("Descarga con éxito. Nombre de archivo:",s,".html")
```



## tiempo.lex

C/C++

```
%{  
  
#include <stdio.h>  
#include<string.h>  
#include <ctype.h>  
#include <stdbool.h>  
  
char ciudadActual[256];  
char TemperaturaA[256];  
char tiempoAct[256];  
char provinciaAct[256];  
char precipitacionProbAct[256];  
char vientoAct[256];  
char temperaturas_tag[12500];  
char viento_tag[5000];  
  
int temperaturas_semana_max[15];  
int temperaturas_semana_min[15];  
int viento_semanal[15];  
  
int contador_max = 0;  
int contador_min = 0;  
int cont_viento = 0;  
  
void formatea(char *array){  
    size_t tam = strlen(array);  
    for(int i = 0; i<tam; i++){  
        while(array[i] == 39 || ispunct(array[i])){  
            memmove(&array[i], &array[i + 1], tam - i);  
        }  
    }  
}  
  
void quita_substring(char *substring, char *array){  
    size_t tam = strlen(array);  
    size_t tam_substring = strlen(substring);  
    bool continua = true;  
  
    for(int i = 0; i<tam && continua; i++){  
        if(array[i] == ':')  
            continua = false;  
        for(int j = 0; j<tam_substring; j++){  
            if(array[i] == substring[j]){
```

```

        memmove(&array[i], &array[i + 1], tam
- i);
    }
}

}

}%

letra [a-zA-Z]
digito [0-9]
numero {digito}*
simboloT [°C]
medidaViento [km/h]
Ciudad ("niv4:")+" "+({letra}+"*"-")+" "
Provincia ("province:")+" "+({letra}+"*"-")+" "

TemperaturaActual
("'currentTemperature:')+" "+{numero}+{simboloT}+" "
TiempoActual ("weatherForecast:')+" "+({letra}+"*")+""
ProbPrecipitacionActual
("'precipitationProbability:')+" "+{numero}+" "
VientoActual ("windSpeed:')+" "+{numero}+"
"*{medidaViento}+" "

exp_temp_max_tag ("<span
class=\"m_table_weather_day_max_temp\">")
exp_temp_min_tag ("<span
class=\"m_table_weather_day_min_temp\">")
exp_temp_tag ("<span data-temp=\"\"{numero}+\" \"
data-temp-include-units=\"\">")
exp_tag_viento ("<span data-wind=\"\"{numero}+\" \"
data-wind-include-units=\"1\">")
exp_tag_lluvia ("<span>\"{numero}+\".\"?\", \"?{numero}+\" mm")

%%
{Ciudad} {
    strcpy(ciudadActual, yytext);
    quita_substring("niv4",ciudadActual);
    formatea(ciudadActual);
    ciudadActual[0] = toupper(ciudadActual[0]);
}

{Provincia} {
    strcpy(provinciaAct, yytext);
    quita_substring("province",provinciaAct);

```

```

        formatea(provinciaAct);
        provinciaAct[0] = toupper(provinciaAct[0]);
    }
    {TemperaturaActual} {
        strcpy(TemperaturaA, yytext);

        quita_substring("currentTemperature", TemperaturaA);
        formatea(TemperaturaA);
    }
    {ProbPrecipitacionActual} {

        strcpy(precipitacionProbAct, yytext);

        quita_substring("precipitationProbability", precipitacionProbAct);

        formatea(precipitacionProbAct);
    }
    {TiempoActual} {
        strcpy(tiempoAct, yytext);

        quita_substring("weatherForecast", tiempoAct);
        formatea(tiempoAct);
    }
    {VientoActual} {
        strcpy(vientoAct, yytext);

        quita_substring("windSpeed", vientoAct);
        formatea(vientoAct);
    }
    {exp_temp_max_tag}{exp_temp_tag} {
        strcpy(temperaturas_tag,
yytext);

        quita_substring("_ span
classmtableweatherdaymaxtempspan
datatempincludeunits", temperaturas_tag);
        formatea(temperaturas_tag);
        int temp =
atoi(temperaturas_tag);

        temperaturas_semana_max[contador_max] = temp;
        contador_max++;
    }
    {exp_temp_min_tag}{exp_temp_tag} {

```

```

        strcpy(temperaturas_tag,
yytext);
        quita_substring("_ span
classmtableweatherdaymintempspan
datatempincludeunits",temperaturas_tag);
        formatea(temperaturas_tag);
        int temp =
atoi(temperaturas_tag);

temperaturas_semana_min[contador_min] = temp;
        contador_min++;
    }
{exp_tag_viento} {
        strcpy(viento_tag, yytext);
        quita_substring("span
datawind includeunits 1",viento_tag);
        formatea(viento_tag);
        int viento =
atoi(viento_tag);
        viento_semanal[cont_viento]
= viento;
        cont_viento++;
    }

%%

int yywrap(){
    printf("Informacion obtenida de la web: \n");
    printf ("\tCiudad: %s\n",ciudadActual);
    printf ("\tProvincia: %s\n",provinciaAct);
    printf ("\tTemperatura actual: %s\n",TemperaturaA);
    printf ("\tProbabilidad de lluvia actual: %s %%
\n",precipitacionProbAct);

    printf ("\tTiempo actual: ");
    if(strcmp(tiempoAct, "Clear") == 0)
        printf ("Despejado\n");
    if(strcmp(tiempoAct, "Mostly clear") == 0)
        printf ("Poco nuboso\n");
    if(strcmp(tiempoAct, "Partly cloudy") == 0)
        printf ("Intervalos nubosos\n");
    if(strcmp(tiempoAct, "Cloudy") == 0){
        printf ("Nuboso\n");
    }
    if(strcmp(tiempoAct, "Overcast") == 0){

```

```

        int prob = atoi(precipitacionProbAct);
        if(prob<80)
            printf ("Cubierto\n");
        if(prob>= 80 && prob<90)
            printf ("Cubierto, llovizna\n");
        if(prob>= 90)
            printf ("Cubierto, lluvia\n");
    }
    printf ("\tViento: %s\n",vientoAct);
    printf
    ("-----\n");
    printf ("Prevision quincenal de temperaturas (max y
min)\n");

    for(int i=0; i<contador_max; i++){
        printf("\t%d oC ", temperaturas_semana_max[i]);
    }
    printf ("\n");
    for(int i=0; i<contador_min; i++){
        printf("\t%d oC ", temperaturas_semana_min[i]);
    }

    printf ("\n");
    printf
    ("-----\n");
    printf ("Prevision quincenal de viento\n");
    for(int i=0; i<cont_viento; i++){
        printf("\t%d km/h ", viento_semanal[i]);
    }

    return 1;
}
int main(){
    yyout=fopen("testout.txt","w");
    yylex();
    return 1;
}

```