

examenes-an.pdf



Anónimo



Procesadores de Lenguajes



4º Doble Grado en Ingeniería Informática y Administración y Dirección de Empresas

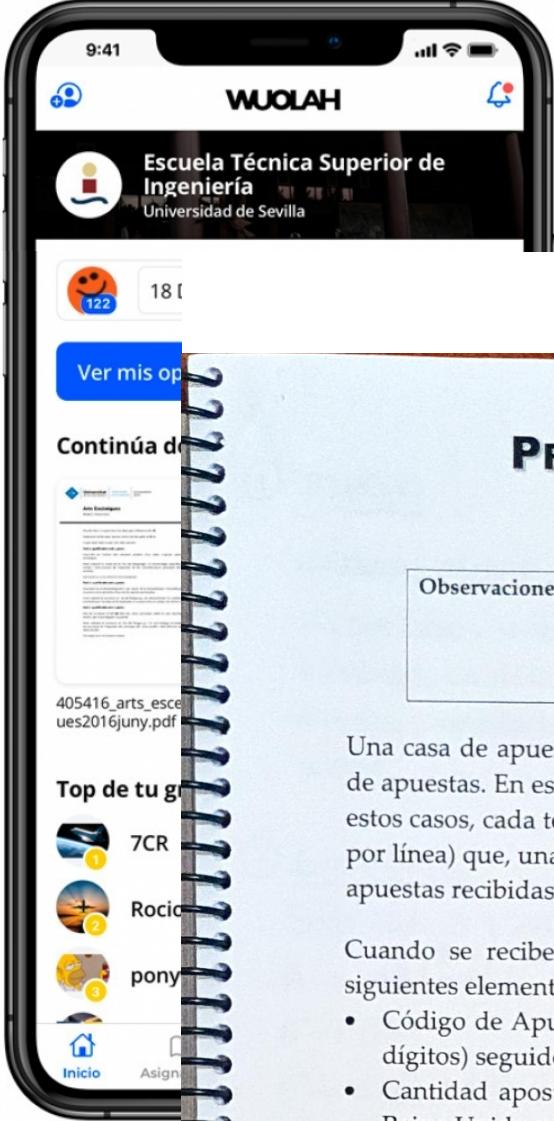


Escuela Técnica Superior de Ingenieros Informáticos
Universidad Politécnica de Madrid



Descarga la APP de Wuolah.
Ya disponible para el móvil y la tablet.





Descarga la APP de Wuolah.

Ya disponible para el móvil y la tablet.

Available on the
App Store

GET IT ON
Google Play

Una casa de apuestas está en la última fase del proyecto de automatización de la gestión de apuestas. En esta etapa, se abordará la gestión de las apuestas recibidas por teléfono. En estos casos, cada teleoperador escribe cada apuesta en un fichero de texto (con una apuesta por línea) que, una vez unificados por día, se vuelcan a la Base de Datos donde ya están las apuestas recibidas por la web y aplicaciones móviles.

Cuando se recibe una llamada telefónica, los teleoperadores anotan en el fichero los siguientes elementos en cualquier orden, pudiendo separarlos por blancos o tabuladores:

- Código de Apuesta: consta del año de la apuesta (un año del siglo actual escrito con 4 dígitos) seguido de un guion y otra serie de dígitos (al menos uno). Por ej.: 2018-123.
- Cantidad apostada: el servicio de teleoperador está disponible en la Unión Europea, Reino Unido y Estados Unidos, por lo que el teleoperador anota la moneda en la que se hace la apuesta (obligatoriamente con parte entera y dos decimales). Por ej.: 7.99€, 28.05£, 45.90\$.
- Código de Usuario: empieza por la letra 'U' y va seguida de 6 dígitos. Por ej.: U123456.
- Hora de la Apuesta: se escribe con horas y minutos en el formato dd:dd. Hay que tener en cuenta que las apuestas por teleoperador solo pueden realizarse desde las 08:00 hasta las 21:59. Por ej.: 14:21, 09:00.

La Base de Datos que almacena la información tiene los siguientes campos:

- Código de Apuesta. Por ej.: 2018-123.
- Cantidad apostada. Se almacena siempre el importe en euros. Por ej.: 7.99, 31.57, 39.55.
- Código de Usuario: Por ej.: U123456.
- Hora de la Apuesta: se almacena como el minuto del día en que se ha recibido la apuesta. Por ej.: 861, 540.

Ejemplo de fichero de apuestas correcto:

2019-34	37.00€	U435678	12:35
5442.90\$	2018-85233852	21:59	U564896

Ejemplo de fichero de apuestas con todos los elementos incorrectos:

2209-34	37.0€	U435	22:35
.90¥	201885233852	08:67	V564896

Se pide realizar el Diseño del Analizador Léxico, que ayude en el proceso de volcar los ficheros de texto a la Base de Datos, en todas sus fases: Tokens, Gramática Regular, Autómata Finito, Acciones Semánticas y Errores.

Nota: Se dispone de acceso al servicio del Banco de España para la tasa de cambio de monedas, con el formato *cambio-diario* (*importe, moneda*), que devuelve la tasa de cambio en euros en formato de número real. Por ej.: *cambio-diario* (45.90, \$) → 39.55.

3 de octubre 2018

① Tokens

- < CODIGO_ARISTA, lexema > → 4 dígitos - dígitos
- < CANTIDAD, valor_en_euros > → Parte entera 4 + 2 decimales
- < CODIGO_USUARIO, lexema > → Letra U 6 dígitos
- < HORA, valor > → 2 num : 2 num
- < eol, - >

② Gramática

1 S → del S | dA | eD | eol ✓

2 A → dA' | -A' | , B | : C ✓

3 A' → dA" ✓

4 A" → dA" | del ✓

5 B → dB' ✓

6 B' → dB" ✓

7 B" → \$ | £ | € ✓

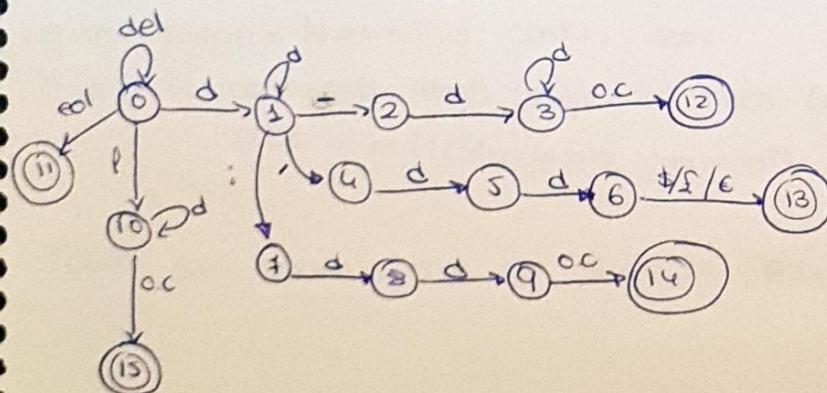
8 C → dc' ✓

9 C' → dc" ✓

10 C" → del ✓

11 D → dD | del ✓

③ Autómata



del: borrar, tabulador

l: letra

d: dígito

oc: otro carácter



**KEEP
CALM
AND
ESTUDIA
UN POQUITO**

④ Acciones semantics y errores

0-0: Leer;

0-11: Gen-Token(ed, .); Leer;

0-2: lexema=d; cont=3; valor=d; Leer

1-1: lexema=lexema \oplus d; cont++; valor=valor \times 10 + valorA(d); Leer

1-2: lexema=lexema \oplus ..; Leer

2-3: lexema=lexema \oplus d; Leer

3-3: lexema=lexema \oplus d; Leer

3-12: If cont <> 4 then error ("Código apuesta debe tener 4 dígitos antes del -")
Else Gen-Token(CODIGO_APUESTA, lexema); Leer

4-4: valor=valor \oplus ,; Leer

4-5: valor=valor \times 10 + valorA(d); Leer

5-6: valor=valor \times 10 + valorA(d); Leer \rightarrow xq hay 2 decimales

6-13: valor=valor/100; valor=cambio_dario(valor, car) //car= €, \$, £
Gen-Token(CANTIDAD, valor_en_euros); Leer

1-f: valor=valor \oplus ; Leer

7-8: valorMin=d; Leer

8-9: valorMin=valorMin \times 10 + valorA(d); Leer

9-14: If cont == 2 AND 8 <= valor < 22 AND valorMin <= 59
Then Gen-Token(HORA, valor); Leer
ELSE error ("Hora incorrecta")

0-10: lexema=l; cont2=0; car=l; Leer

10-10: lexema=lexema \oplus d; cont++; Leer

10-15: If cont == 6 AND car == 'U' Then Gen-Token(CODIGO_USUARIO, lexema); Leer
ELSE error ("Código usuario incorrecto")

TODAS las transiciones no indicadas PRODUCEN error.

PROCESADORES DE LENGUAJES Y COMPILEDORES

30 de junio de 2017

Observaciones:

1. Las calificaciones del primer parcial se publicarán hacia el 13 de julio y la revisión será hacia el 17 de julio. Las fechas exactas se avisarán en la web de la asignatura.
2. La duración de este examen es de 2 horas.
3. Cada ejercicio debe entregarse en hojas separadas.

1. Un lenguaje tiene los siguientes elementos:

- Operadores aritméticos: +, -, *, **
- Operadores lógicos: .AND., .OR., .NOT.
- Identificadores: comienzan por una letra y van seguidos de un máximo de 15 letras o dígitos
- Palabras reservadas en mayúsculas, como: IF, THEN, OUT, LOOP, ENDLOOP, NAND...
- Números reales según la expresión regular: $d^* \cdot d^*$

Teniendo en cuenta que el lenguaje distingue mayúsculas de minúsculas y que los elementos pueden ir separados por delimitadores, se pide construir un **Analizador Léxico** para este lenguaje (*tokens*, gramática, autómata finito determinista, acciones semánticas y errores), que introduzca toda la información posible en la Tabla de Símbolos.

2. Dada la siguiente gramática:

$$\begin{aligned} P &\rightarrow D\ P \mid I\ P \mid \lambda \\ D &\rightarrow T\ id \\ T &\rightarrow \text{int} \mid \text{real} \\ I &\rightarrow \text{inicia}\ id \ V \\ V &\rightarrow \text{cte-real}\ V \mid \text{cte-int}\ V \mid id \end{aligned}$$

Se pide:

- a. Demostrar si la gramática es LL(1).
- b. Diseñar la tabla del **Analizador Sintáctico LL(1)**.
- c. Construir el estado inicial del Autómata reconocedor de prefijos viables de un **Analizador Sintáctico Ascendente LR**, así como todos los demás estados que se deriven de él en una sola transición. Determinar si existen conflictos en estos estados y, en su caso, ¿cuáles son?
- d. En el supuesto de que se introdujera al lenguaje generado por esta gramática la restricción de que en la lista de valores (V) de la sentencia inicia (I) siempre debe existir un mayor número de constantes reales que enteras, ¿qué cambios habría que introducir a los Analizadores Sintácticos del apartado b y del apartado c?

3. Dado el siguiente fragmento de gramática:

$$\begin{aligned} E &\rightarrow E_1 \text{ out } \{ E_2, E_3 \} \mid E_1 + E_2 \mid E_1 \text{ nand } E_2 \mid id \\ S &\rightarrow \text{if } E \text{ then } S_1 \mid \text{loop } E\ S_1 \text{ endloop } \mid S_1 ; S_2 \end{aligned}$$

El lenguaje tiene las siguientes características:

- El lenguaje tiene los tipos entero, lógico y cadena
- El lenguaje no tiene conversión automática de tipos
- Todas las variables se han de declarar antes de ser usadas
- El operador lógico out devuelve falso si el número E_1 es mayor que E_2 y menor que E_3
- El operador lógico nand representa la operación *not and*
- El operador + realiza una suma entre enteros y una concatenación entre cadenas
- La sentencia loop funciona de la siguiente manera: se evalúa la expresión E; si es cierta o si es un número mayor que cero, se ejecuta el cuerpo del loop y se vuelve a evaluar la expresión; si es falsa o vale cero o menos, se sale del loop
- La sentencia if se ejecuta si la expresión es cierta

Se pide diseñar el **Analizador Semántico** mediante una **Definición Dirigida por la Sintaxis**, explicando brevemente las funciones y atributos utilizados.



[Ver mis op](#)

Continúa d

405416_arts_escuela2016juni.pdf

Top de tu g

7CR
Rocio
pony

Inicio Asigna

Descarga la APP de Wuolah.

Ya disponible para el móvil y la tablet.

Available on the
App Store

GET IT ON
Google Play

30 junio 2017

① Tokens

<MAS,>

<MENOS,>

<POR,>

<PORPOR,>

<OP_Logico, n>

n=1 : AND.

n=2 : OR.

n=3 : NOT.

<ID, posTS> → empieza x letra, seguidas de un nº de ls letras o dígitos

<PALRES, posTPRS> → mayuscula

<REAL, valores>

② Geométrica

S → delS | + | - | * A | . B | lc | aDlde

| A → * | x

| B → aB | . B'

| B' → λ

| C → lc | dc | λ

| D → aDlx

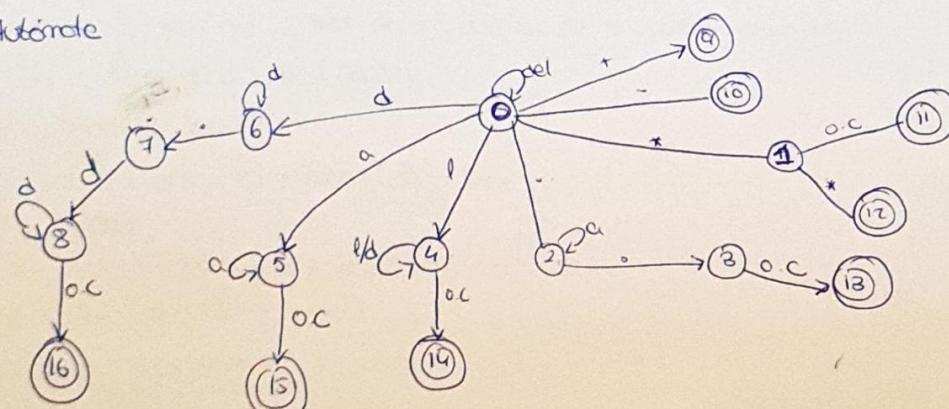
| E → dE' | . E'

| E' → dE"

| E" → dE" | x

del: delimitador
a: carácter en mayúsculas
l: letra en minúsculas
d: dígito

③ Autómata



④ Acciones semánticas y errores

0-0: Leer
 0-1: Leer;
 1-11: Gen-Token(POR,)
 1-12: Gen-Token(PORPOR,); Leer
 0-2: Lexema = .; Leer
 2-2: Lexema := Lexema ⊕ a; Leer
 2-3: Lexema := Lexema ⊕ .; Leer
 3-13: p = buscar_Operador_ARIT(lexema)
 If p == null then error ("No existe ese operador aritmético")
 Else if (p == 'AND') then Gen-Token(OP_LOGICO, 1);
 Else if (p == 'OR') then Gen-Token(OP_LOGICO, 2);
 Else Gen-Token(OP_ARITMETICO, 3);
 0-4: Lexema := l; cont = 1; Leer
 4-4: Lexema := Lexema ⊕ l/d; cont++; Leer
 4-14: p = buscar_TS(Lexema)
 If cont <= 15 Then
 If p == null then Gen-Token(ID, posTS); Insertar_TS(Lexema);
 Else error ("El ID ya se encuentra en la TS")
 Else error ("Debe tener meno de 15 caracteres")
 4-5: Lexema = a; Leer
 5-5: Lexema := Lexema ⊕ a; Leer
 5-15: p = buscar_TPR(Lexema)
 If p == null then error ("No existe la palabra reservada")
 Else Gen-Token(PALRES,);
 0-6: valor = d; Leer
 6-6: valor := valor * 10 + valorA(d); Leer
 6-7: Leer,
 7-8: valor := valor * 10 + valorA(d)/10; Leer
 8-16: Gen-Token(leer, valor)
 0-10: Gen-Token(menos,); Leer
 0-9: Gen-Token(mas,); Leer

TODAS LAS TRANSICIONES NO ESPECIFICADAS PRODUCEN ERROR

PROCESADORES DE LENGUAJES Y COMPILADORES

12 de enero de 2017

Observaciones:

1. Las calificaciones se publicarán hacia el 23 de enero.
2. La revisión será hacia el 25 de enero.
3. En la web se avisará de las fechas exactas.
4. La duración de este examen es de 40 minutos por pregunta.
5. Todos los ejercicios tienen la misma puntuación.

1. Una sentencia simplificada de un lenguaje tiene la siguiente estructura en EBNF:

```

S ::= SELECT [DISTINCT] columna {, columna} FROM tabla {, tabla} [WHERE Cond] [ORDER BY columna]
Cond ::= E OpRel E
E ::= núm_entero | núm_real | id(E) | id | (S)
OpRel ::= > | < | <= | >= | <>
  
```

Donde:

- id, columna y tabla son identificadores que comienzan por una letra que puede ir seguida de letras, dígitos o dólares (\$), pero no puede haber dos o más dólares seguidos.
- El lenguaje diferencia entre minúsculas y mayúsculas.
- Las palabras reservadas del lenguaje van en mayúsculas.
- Los elementos del lenguaje pueden ir separados por blancos, tabuladores y saltos de línea
- Los números pueden llevar parte decimal (798.37) o no (873); en caso de llevar parte decimal, ésta va separada con un punto. En un número con parte decimal se puede omitir la parte entera (.578), pero no la decimal (88. es incorrecto).
- En la notación EBNF, los corchetes ([]) y las llaves ({}) no forman parte del lenguaje, e indican, respectivamente, opcionalidad y que el contenido puede aparecer de 0 a varias veces.

Se pide diseñar un **Analizador Léxico** para este fragmento de lenguaje (indicando una gramática regular, tokens completos, autómata finito determinista, acciones semánticas y errores), que introduzca toda la información posible en la **Tabla de Símbolos**.

2. Dada la siguiente G (y sin modificarla):

```

S → A + B ; C
A → x A | C
B → B y | λ
C → (A) | λ
  
```

Se pide:

- Determinar (razonadamente) qué tipo de **analizador sintáctico** puede ser **válido** para esta gramática
- Construir la **tabla** de dicho **analizador sintáctico**.

Enero de 2017

① Tokens

$\langle \text{ID}, \text{posTS} \rangle \rightarrow$ id comienza x letra, seguido de l/d/ \$ → NO más de 3 seguidos

$\langle \text{PALRES}, \text{posTPR} \rangle \rightarrow$ mayúscula

$\langle \text{ENT, valor} \rangle \rightarrow$ 79832
273 . 573
88 → ID

$\langle \text{Real, valor} \rangle$

$\langle \text{eol, } \rangle$

$\langle \text{OP_REL, n} \rangle$ atbs no con significado \rightarrow 1: > , 2: >= , 3: < , 4: <= , 5: <>

$\langle \text{PARENTESIS, n} \rangle$ 1 paréntesis cerrado, 2 paréntesis abierto

$\langle \text{coma, } \rangle$

② Gramática

0 $S \rightarrow \text{delS} \mid \ell A \mid d B \mid . B' \mid \dots \mid 1 \ell \mid \ell \mid > D \mid < E \mid a F$

1 $A \rightarrow \ell A \mid d A \mid \$ A'$

2 $A' \rightarrow \ell A \mid d A \mid \lambda$

3 $B \rightarrow d B \mid \lambda \mid . B'$ atbs que no
tienen alc.

4 $B' \rightarrow d B'' \mid \lambda$ atbs que no
tienen alc.

5 $D \rightarrow = \mid \times$

6 $E \rightarrow = \mid \times \mid \lambda$

7 $F \rightarrow a F \mid \text{dah}$

del: blanco, tabulador, eol

l: letra minúscula

d: dígito

a: letra mayúscula

λ : otro carácter Pueden ir separados
(NO obligatoriamente)



Ver mis op

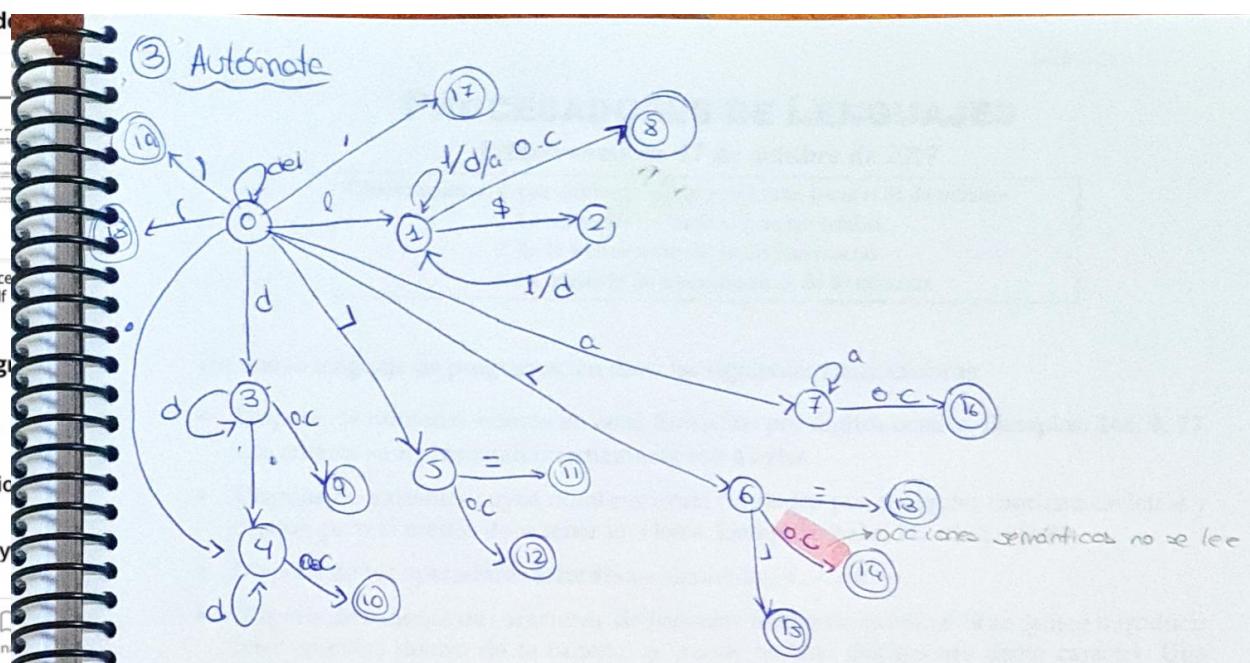
Continúa d

405416_arts_escuela2016juni.pdf
Auto Escritor

Top de tu g

7CR
Rocio
pony
Inicio

Asignac



(4) Acciones semánticas y errores

0-0: Leer

0-19: Gen-Token(PARENTESIS, 1); Leer

0-18: Gen-Token(PARENTESIS, 2); Leer

0-17: Gen-Token(Coma,); Leer

0-1: Lexema = l; Leer

1-1: Lexema = Lexema ⊕ l, d, ; Leer

1-2: Lexema = Lexema ⊕ \$; Leer

2-1: Lexema = Lexema ⊕ l, d; Leer

1-8

p = buscar_TS(Lexema);

If (p=null) then p = Inserta_TS(Lexema); Gen-Token(ID, p.octs); Leer
Else error("Identificador ya declarado")

1-7: Lexema = a; Leer

1-7: Lexema = Lexema ⊕ a; Leer

1-16: p = buscar_TPR(Lexema);

If (p=null) Then error("Palabra reservada incorrecta");

Else Gen-Token(PALRES, p.octs); Leer

0-6: Leer;

6-13: Gen-Token(OP_REL, 4); Leer

6-14: Gen-Token(OP_REL, 3); Leer

6-15: Gen-Token(OP_REL, 5); Leer

0-5: Leer;

5-11: Gen-Token(OP_REL, 2); Leer

5-12: Gen-Token(OP_REL, 3)

0-3: valor = d; Leer

3-3: valor = valor * 10 + valorA(d); Leer

3-9: Gen-Token(ENT, valor);

3-4: Leer;

0-4: Leer;

4-4: valor = valor * 10 + valorA(d)/10; Leer

4-10: Gen-Token(Real, valor)

PROCESADORES DE LENGUAJES

Primer examen. 17 de octubre de 2017

- Observaciones:**
1. Las calificaciones se publicarán hacia el 31 de octubre.
 2. La revisión será hacia el 3 de noviembre.
 3. En la web se avisarán las fechas exactas.
 4. La duración de este examen es de 40 minutos.

Un nuevo lenguaje de programación tiene las siguientes características:

- Dispone de números enteros en octal formados por dígitos octales. Ejemplos: 246, 0, 77. Los enteros se representan internamente con 4 bytes.
- Dispone de variables cuyos nombres están formados por cualquier cantidad de letras y dígitos, pero al menos debe tener una letra. Ejemplos: hola, 3E, c123, 98w89.
- Dispone de los operadores aritméticos siguientes: +, -, ++, --.
- Dispone de cadenas de caracteres, delimitadas mediante comillas. Si se quiere introducir unas comillas dentro de la cadena, se puede realizar duplicando dicho carácter. Una cadena no puede tener más de 80 caracteres.

Teniendo en cuenta que los distintos elementos del lenguaje pueden ir separados por blancos, tabuladores o saltos de línea y que no hay distinción entre mayúsculas y minúsculas, se pide diseñar un **Analizador Léxico** para este lenguaje (*Tokens, Gramática, Autómata y Acciones Semánticas*), que introduzca toda la información posible en la Tabla de Símbolos.

Ejemplo de un fragmento de fichero correcto en este lenguaje:

```
1ab3cd+ 3d 1 -- 0
"holá"++
+ "Adiós"
-3141 13579L
Holá -- "3,1416 (""pi""")
-Begin 333-333+ 0110 ++ 3D
```

17 octubre 2017

① Tokens

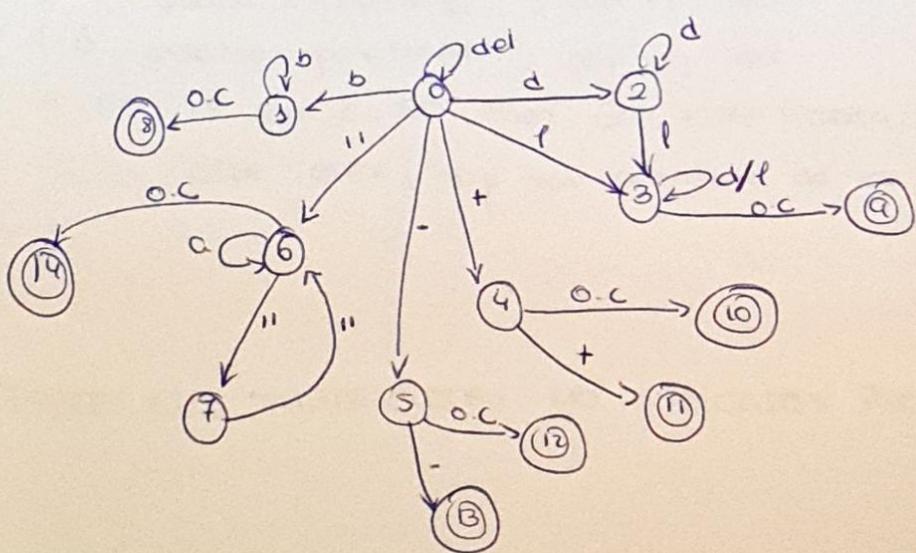
- $\langle \text{ENTERO}, \text{valor} \rangle \rightarrow \text{digito octal}$
- $\langle \text{ID}, \text{posTS} \rangle \rightarrow \text{Nºs, letras, al menos 1 letra}$
- $\langle \text{OP_ARITmetico}, n \rangle \rightarrow$ $n=1: +$ $n=3: ++$
 $n=2: -$ $n=4: --$
- $\langle \text{CADENA, lexemas} \rangle \rightarrow \text{entre " "}$

② Gramática

- 1 $S \rightarrow \text{del } S \mid bA \mid dB \mid +D \mid -E \mid "F"$
- 2 $A \rightarrow bA \mid \lambda$
- 3 $B \rightarrow dB \mid \epsilon C$
- 4 $C \rightarrow dC \mid \epsilon C \mid \lambda$
- 5 $D \rightarrow + \mid \lambda$
- 6 $E \rightarrow - \mid \lambda$
- 7 $F \rightarrow \alpha F \mid "G" \mid \lambda$
- 8 $G \rightarrow "F"$

b: digito octal
 a: cualquier carácter menos "
 d: digito
 l: letra
 o.c.: otro carácter
 del: delimitador

③ Autómata



4. Acciones semánticas y errores

0-0: Leer

0-1: valor = b; Leer

1-1: valor = valor * 10 + valorA(b); Leer

1-8: Gen-Token (entero, valor)

0-2: cadena = d; Leer

2-2: cadena = cadena ⊕ d; Leer

2-3: cadena = cadena ⊕ l; Leer

0-3: cadena = ~~kkkkkkkk~~ ⊕ l; Leer

3-3: cadena = cadena ⊕ l,d; Leer

3-4: Gen-Token (D, posTS) 39:

$p = \text{buff_TS}(\text{cadena})$,
if $p == \text{null}$ then insert- $\text{TS}(\text{cadena})$,
Gen-Token (D, posTS)
else error ("ID ya insertado")

0-4: Leer

4-10: Gen-Token (OP_ARITMETICO, 1)

4-11: Gen-Token (OP_ARITMETICO, 2)

0-5: Leer

5-12: Gen-Token (OP_ARITMETICO, 3)

5-13: Gen-Token (OP_ARITMETICO, 4)

0-6: palabra = \emptyset ; cont = 0; Leer

6-6: palabra = palabra ⊕ a; cont++; Leer

6-7: palabra = palabra ⊕ " "; cont++; Leer

7-6: palabra = palabra ⊕ " "; cont++; Leer

6-14: IF cont <= 80 then Gen-Token (cadena, lexema)

Else error ("Hay más caracteres de los permitidos")

TODAS LAS TRANSICIONES NO INDICADAS PRODUCEN ERROR

SEPTIEMBRE 2016

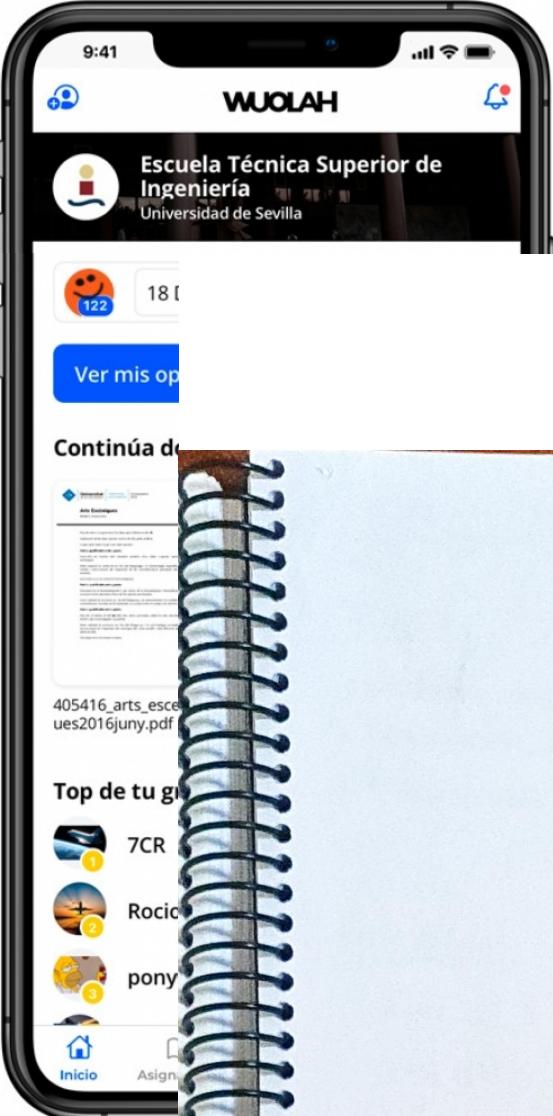
Léxico

Descarga la APP de Wuolah.

Ya disponible para el móvil y la tablet.

Available on the
App Store

GET IT ON
Google Play



Teniendo en cuenta que los distintos elementos del fichero fuente tienen que ir separados obligatoriamente por uno o más blancos o saltos de línea y que la tienda quiere tener todos sus precios en euros, se pide diseñar un Analizador Léxico para este lenguaje (tokens completos, gramática regular, autómata finito determinista y acciones semánticas).

Moneda	Código	Cambio oficial	Ejemplos
Euro	EUR	1	20,21EUR
Dólar estadounidense	USD	0,88948	24,48USD
Corona sueca	SEK	0,10513	19,95GBP
Libra esterlina	GBP	1,18290	13,95SEK
...	55555COP

Un sistema de gestión de una tienda lee de un fichero de texto los nuevos productos que entran en el almacén. Dicho fichero tiene la siguiente estructura:

- Número de unidades de cada producto: entero.
- Descripción, que puede incluir el nombre, la marca y el modelo del producto: una o varias cadenas de caracteres encerradas entre comillas.
- Precio: número (con dos, uno o ningún decimal) seguido inmediatamente de la unidad monetaria según el formato internacional de moneda de tres letras mayúsculas definido por el estándar ISO 4217. Para comprobar los 178 códigos válidos, así como el cambio oficial del día, se tiene una tabla similar a la siguiente (se muestran también ejemplos de precios válidos):

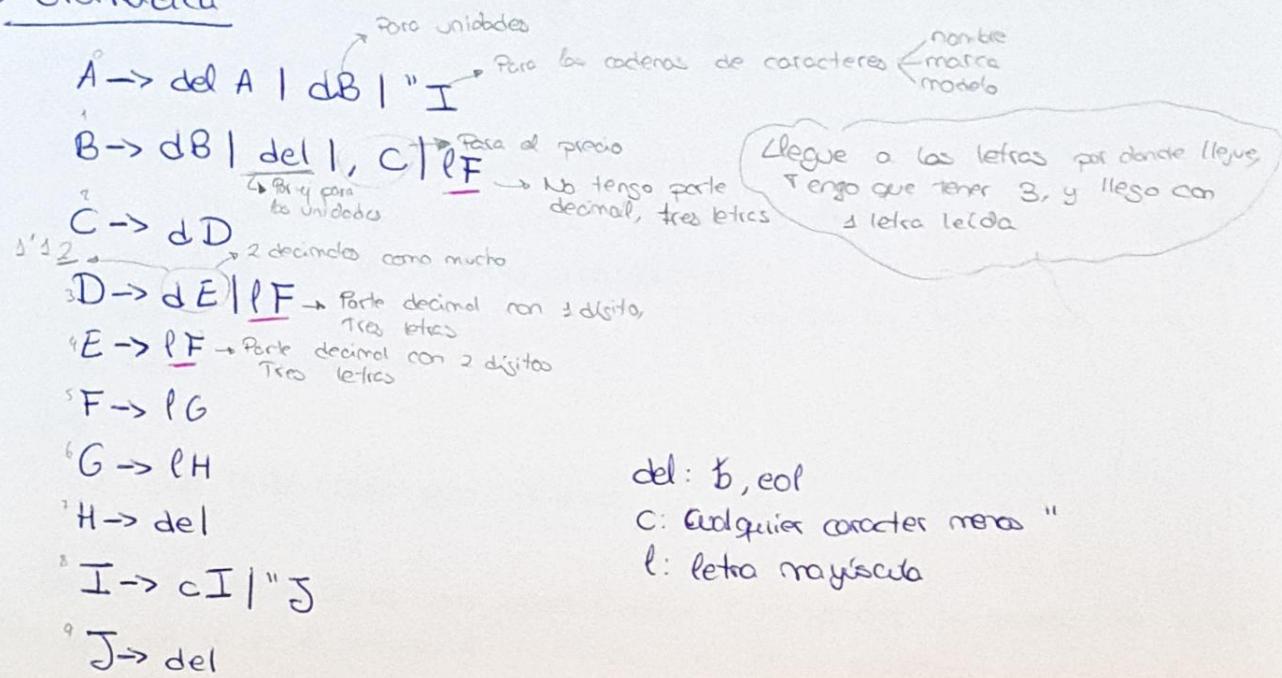
WUOLAH

Tienda → Septiembre 2016

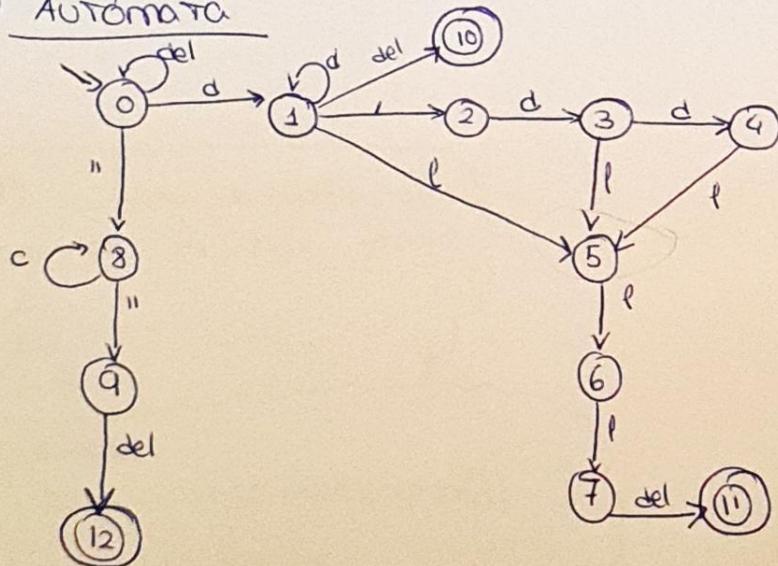
① Tokens

- < Unidades, valores >
- < Descripción, lexemas >
- < Precio, valor en euros >

② Gramática



③ AUTÓMATA



④ Acciones semánticas y errores

Leer en todas las transiciones, al final

0-3: lexema = \emptyset , num = valorA(d) → para leer la moneda

1-3: num = num * 10 + valorA(d)

3-10: Gen-Token (Unidades, num)

1-2: - → No Hago nada

2-3: num = num * 10 + valorA(d) / 10

3-4: num = num * 10 + valorA(d) / 100

1-5, 3-5, 4-5: lexema = lexema \oplus l // Podrá haber puesto lexema = l si no lo hubiese inicializado arriba

5-6: lexema = lexema \oplus l

6-7: lexema = lexema \oplus l

7-11: cambio := busca_Tabb_Gambio (lexema)

If cambio = null then error ("Moneda no válida")

Else Gen-Token (Precio, num/cambio);

0-8: lexema = \emptyset → para que me de el valor en €

8-8: lexema = lexema \oplus l

8-9: -

9-12: Gen-Token (Descripción, lexema)

Todos las transiciones no especificadas corresponden a casos de error

Por ejemplo, si en el estado 7 recibimos una letra, podemos ejecutar la

llamada al módulo error: error ("La moneda se define con 3 letras")

① concatenar la cadena de caracteres que vas generando.

E

EU

EUR

Con números:

num = num * 10 + valorNuevo(d) x es dígito

Número 32 → num = 3

num = 30 + 2 = 32

Para que no haga 312

ANÁLISIS LÉXICO Y TABLA DE SÍMBOLOS

Primer examen. 26 de octubre de 2016

- Observaciones:**
1. Las calificaciones se publicarán hacia el 15 de noviembre.
 2. La revisión será hacia el 17 de noviembre.
 3. En la web se avisará de las fechas exactas.
 4. La duración de este examen es de 40 minutos.

Se tiene un lenguaje de programación con las siguientes características:

- Existen declaraciones de funciones y de variables (globales). Sus nombres deben empezar por letra y pueden ir seguidos por cualquier cantidad de letras y dígitos. Los nombres pueden llevar también dólares (\$) en su interior (pero no pueden ni empezar ni finalizar por dólar).
- Una función se declara poniendo la palabra reservada **function**, el tipo que retorna, su nombre y los tipos y nombres de los parámetros formales encerrados entre paréntesis y separados por comas. Los parámetros siempre se pasan por valor. A continuación vienen las declaraciones de variables (locales) y las sentencias encerradas entre las palabras reservadas **begin** y **end**.
- Una variable se declara poniendo la palabra reservada **var**, su nombre y su tipo (representado mediante las palabras reservadas **int** o **real**). El tipo entero ocupa 2 bytes y el real 4.
- Las sentencias son asignaciones (mediante **:=**) de expresiones a variables.
- Los operandos de las expresiones pueden ser variables, llamadas a funciones u otras expresiones. Las expresiones pueden llevar paréntesis.
- Se pueden realizar operaciones aritméticas sobre datos del mismo tipo. Los operadores disponibles son la suma (+), resta (-) y menos unario (-).
- Las llamadas a funciones se escriben poniendo los parámetros actuales entre paréntesis y separándolos por comas.

Teniendo en cuenta que los distintos elementos del lenguaje pueden ir separados por blancos, tabuladores o saltos de línea y que no hay distinción entre mayúsculas y minúsculas, se pide:

- a. Construir las **Tablas de Símbolos** que generaría un Compilador para el ejemplo.
- b. Diseñar un **Analizador Léxico** para este lenguaje (*Tokens, Gramática Regular, Autómata Finito Determinista y Acciones Semánticas*), que introduzca toda la información posible en la Tabla de Símbolos.

Ejemplo de un fragmento de programa en este lenguaje:

```

Var ab$cd    Int
Var      a Real
Var z3       Int
Function real f (int q, real w$1$$2)
Begin
  ab$cd:=-z3-ab$cd+q
  a := w$1$$2
End
Function int funcion (real q) var z3 real
begin z3 := - (q - f(-ab$cd, z3 + z3)) end
  
```



Ver mis op.

Continúa de

405416_arts_escuela2016juni.pdf

Top de tu grupo

7CR
Rocio
pony
Inicio Asign.

Descarga la APP de Wuolah.

Ya disponible para el móvil y la tablet.

Available on the App Store

GET IT ON Google Play

26 octubre 2016

① Tokens

$\langle ID, posTS \rangle \rightarrow \text{Id/o, l/d } \$ \rightarrow \text{No terminar en } \$$

$\langle PAIRS, posPR \rangle$

$\langle PAB, \rangle / \text{Paréntesis abierto}$

$\langle PCE, \rangle / \text{Paréntesis cerrado}$

$\langle coma, \rangle$

$\langle ASIGNACION, \rangle$

$\langle OP_ARITMETICO, n \rangle \begin{matrix} n=1: + \\ n=2: - \end{matrix}$

② Gramática

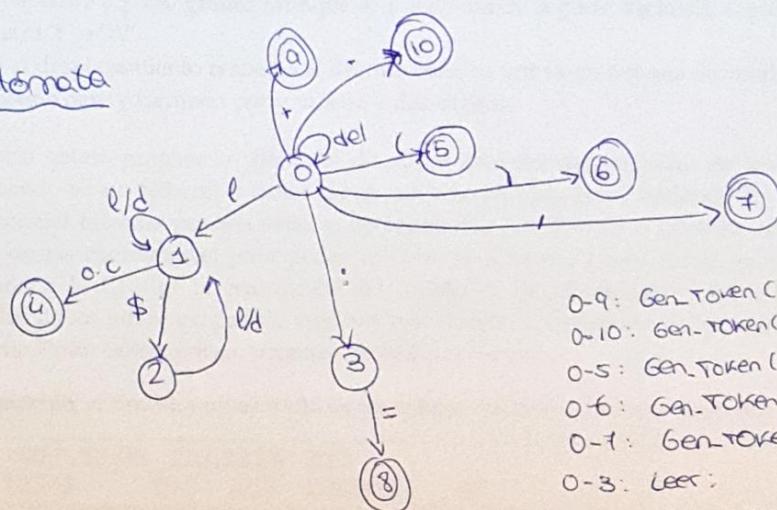
$S \rightarrow \text{del S} | eA \quad 1(1) \quad 1, 1; B \quad 1+1-$

$A \rightarrow eA \quad dA \quad 1\$ \quad A' \quad 1$

$A' \rightarrow eA \quad dA$

$B \rightarrow =$

③ Autómata



0-0: Leer
0-1: lexero=l; Leer
1-1: lexero:=lexero ⊕ l/d; Leer
1-2: lexero:=lexero ⊕ \$; Leer
2-2: lexero:=lexero ⊕ l, d; Leer
1-4: p=buscar_TPR(lexero);
IF p=NULL THEN q=buscar_TS(lexero); Gen-Token(ID, posTS)
ELSE error ("ID ya esté en TS")
ELSE Gen-Token(PAIRS, posPR)

TODAS LAS TRANSICIONES NO ESPECIFICADAS PRODUCEN ERROR

ANÁLISIS LÉXICO Y TABLA DE SÍMBOLOS

Primer examen. 23 de octubre de 2014

- Observaciones:**
1. Las calificaciones se publicarán hacia el 18 de noviembre.
 2. La revisión será hacia el 21 de noviembre.
 3. En la web se avisará de las fechas exactas.
 4. La duración de este examen es de 40 minutos.

Una agencia internacional de espías ha decidido introducir un nuevo sistema para que los espías informen de sus movimientos. Para ello, el espía deberá emitir su código clave y las coordenadas geográficas de su destino base, junto a un desplazamiento opcional respecto a dicha base. Si el espía está trabajando en equipo junto a otros espías, deberá enviar en su informe el código clave de todos los espías del equipo.

- El código clave de espía está formado por una secuencia de entre 3 y 5 dígitos y solo son válidos los códigos de espías almacenados en la tabla de espías de la agencia.
- Las coordenadas geográficas son un sistema de referencia que utiliza las dos coordenadas angulares, latitud y longitud, y sirve para determinar la posición exacta de un punto sobre la superficie terrestre:
 - La latitud mide el ángulo entre cualquier punto y el ecuador, de tal manera que será latitud Norte (N) si está al norte del ecuador, y latitud Sur (S) si está al sur. El valor de la latitud será un número real entre 0 y 90 grados (aunque se puede omitir la parte decimal), seguido inmediatamente de la letra 'N' o 'S'.
 - La longitud mide el ángulo a lo largo del Ecuador desde cualquier punto de la Tierra al meridiano de Greenwich. Se acepta que Greenwich en Londres es la longitud 0, por lo que longitud Este (E) se sitúa al este de Greenwich y longitud Oeste (W) al oeste de Greenwich. La longitud será un número real entre 0 y 180 grados (aunque se puede omitir la parte decimal), seguido inmediatamente de la letra 'E' o 'W'.
- Para el desplazamiento respecto al destino base, se utiliza un sistema de cuadrícula secreta centrada en el destino base y formado por una letra y dos dígitos.

La agencia quiere procesar al final del día todos los informes recibidos de los espías y que se han ido almacenando en un fichero, a razón de un informe en cada línea, teniendo en cuenta que los diferentes elementos del informe pueden venir o no separados por blancos o tabuladores. La agencia necesita los valores de las coordenadas para poder calcular la distancia básica entre los espías. Para todo ello, ha encargado a la ETSIInf la realización del diseño de un **Analizador Léxico** que reconozca todos los elementos de los informes, para lo cual hay que detallar la definición de los *tokens*, la Gramática Tipo 3, el Autómata Finito Determinista, acciones semánticas y errores.

Seguidamente, se muestra un ejemplo de un fichero con cinco informes correctos:

```

007 33.0N 123.321W Z23
12345   013 A00 00090S   0E
179.9999999999999999W975 00007 0007 P02 09876 0.00000000000000000001N
                   88.88S   180E 008
33S33.33EP22900
  
```

Seguidamente, se muestra un ejemplo de un fichero en el que todos los elementos son incorrectos (asumiendo que el espía 'doble-cero tres' no pertenece a la agencia):

```

07 33.0 183.321W 23Z
123456   0A3 A0 91SE
179.9999999999999999N 9T5 003 P2 99 0.0000000000000000000000.1W
                   H8H8 .8S -888E 008
33 N 33°W # KAOS
33.S 13,33E P-22 7
  
```

① Tokens

$\langle \text{ESPIA}, \text{pos} \rangle$ → ab: posición en la tabla de agentes

$\langle \text{Norte}, \text{valor} \rangle$ → lexema: dígitos formando número real entre 0 y 90 grados

$\langle \text{Sur}, \text{valor} \rangle$ → lexema: dígitos formando número real entre 0 y 90 grados (siempre se puede omitir la parte decimal) seguido de la letra N o S

$\langle \text{Este}, \text{valor} \rangle$ → lexema: dígitos formando número real entre 0 y 180 grados (siempre se puede omitir la parte decimal) seguido de W o E

$\langle \text{Oeste}, \text{valor} \rangle$ → lexema: dígitos formando número real entre 0 y 180 grados (siempre se puede omitir la parte decimal) seguido de W o E

zlatitud, valor Atb valor + Norte, valor - Sur

$\langle \text{Longitud}, \text{valor} \rangle$ → Atb valor + Este, - Oeste

$\langle \text{Desplazamiento}, \text{lexema} \rangle$ → letra seguida de 2 dígitos

ab: lexema de la cuadrícula

$\langle \text{CR}, \rightarrow \rangle$ → fin de linea

②

$A \rightarrow dB | l | E | \text{del A} | cr$

$B \rightarrow dB | x | l | C | ln | ls | el | w$

$C \rightarrow dD$

$D \rightarrow dD | ln | ls | el | w | x$

$E \rightarrow dF$

$F \rightarrow d$

1-12: Gen_Token(cr, p); Leer

0-0: Leer

0-1: num:=d; cont=1; p++; Leer

4-13: num = num * 10 + valorA(d); cont++; Leer

5-13: IF cont < 3 or cont > 5 THEN error ("...")

Else q := buscarT(pal)

IF (p == NUEV) THEN error ("...")

Else Gen_Token (ESPIA, p)

1-2: dec := 0

2-3: dec++; num = num * 10 + valorA(d); Leer

3-3: dec++; num = num * 10, valorA(d); Leer

7-14, 3-14: IF (num/30^{dec} > 90) THEN error ("...")

Else Gen_Token (<Norte, num/30^{dec}>); Leer

7-15, 3-15: IF (num/10^{dec} > 90) THEN error ("...")

Else Gen_Token (<Sur, num/10^{dec}>); Leer

1-16, 3-16: IF (num/30^{dec} > 180) THEN error ("...")

Else Gen_Token (<Este, num/30^{dec}>); Leer

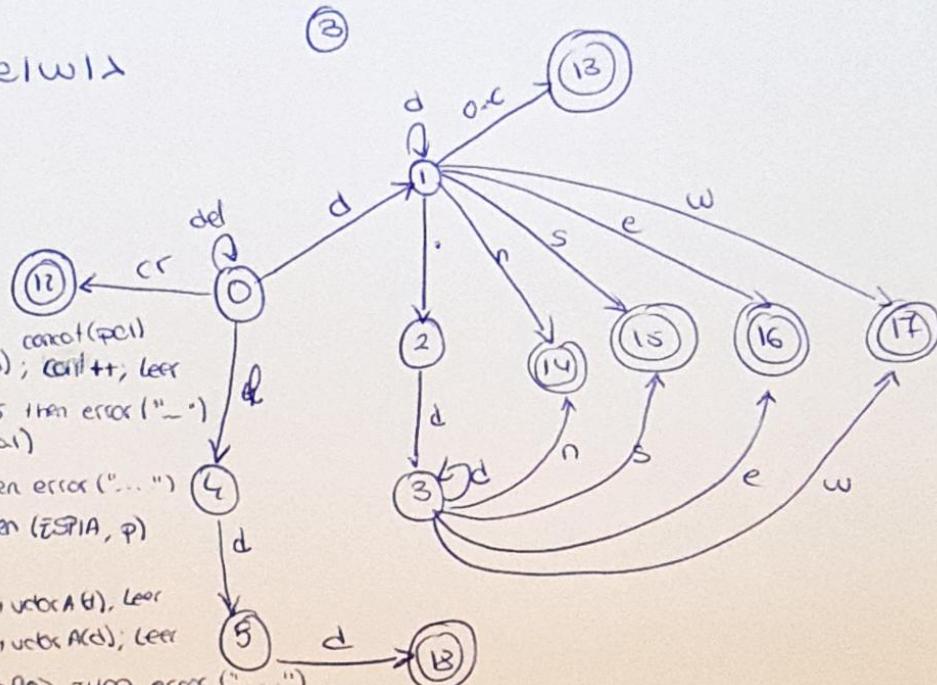
1-17, 3-17: IF (num/30^{dec} > 180) THEN error ("...")

Else Gen_Token (<Oeste, num/10^{dec}>); Leer

4: pal := l; Leer

5: pdl := pdl ⊕ d; Leer

5-18: pdl = pdl ⊕ d; Gen_Token (<Desplazamiento, pdl>); Leer



ANÁLISIS LÉXICO Y TABLA DE SÍMBOLOS

Primer examen. 22 de octubre de 2012

- Observaciones:**
1. Las calificaciones se publicarán hacia el 14 de noviembre.
 2. La revisión será hacia el 16 de noviembre.
 3. En la web se avisará de las fechas exactas.
 4. La duración de este examen es de 40 minutos.

Se tiene un lenguaje para representar los datos de alumnos en un fichero. Para cada alumno, el fichero contiene una línea con la siguiente información, llevando cada campo obligatoriamente al menos un blanco al final:

- Número de DNI (compuesto obligatoriamente por 8 dígitos seguidos de una letra)
- Número de matrícula. Existen tres formatos para representar el número de matrícula:
 - 6 dígitos
 - Una letra seguida de 4 dígitos
 - 2 dígitos, la letra 'M' y 3 dígitos
- Nombre y apellidos del alumno (formados únicamente con letras)
- La palabra 'ECTS:' seguida inmediatamente del número de créditos en los que está matriculado (número entero positivo mayor que cero y menor que 61). Si el alumno no está matriculado, este campo no aparecerá.

Se muestra a continuación como ejemplo un fragmento de cómo podría ser este fichero:

```
ECTS:33 04422185K J0499 MARÍA JOSÉ SÁNCHEZ DE MORA Y GÓMEZ DEL POSTIGO
87654321X 010088 ARNOLD ALOIS SCHWARZENEGGER
11M001 FÉLIX PI ECTS:15 60613579T
005780285 101234 MARLENE JOSEPHINE GÓMEZ JUAN ECTS:3
```

Se desea construir un procesador para este lenguaje con el objetivo de pasar la información de dicho fichero a una base de datos en la que se tendrán que almacenar: el nombre y apellidos, DNI y matrícula de cada alumno, así como el número de créditos en los que está matriculado (con el fin de poder contabilizar cuál es el número total de créditos de todos los alumnos).

Para ello, se pide diseñar un **Analizador Léxico** para este lenguaje (*tokens* completos, gramática regular, autómata finito determinista y acciones semánticas).



[Ver mis op](#)

Continúa de

405416_arts_escuela2016juni.pdf

Top de tu g

7CR

Rocio

pony

Inicio Asign.

Descarga la APP de Wuolah.

Ya disponible para el móvil y la tablet.

Available on the
App Store

GET IT ON
Google Play

Octubre 2012

① TOKENS

<cl, ->

< DNI, lexemas

→ el resto del token no puede ser un valor

< NMed, lexemas

< Nombre-Apellido, lexemas

< Créditos, valores → cuando los ECTS 32, 2016 no interesa el valor 32

Si quiero poner ++

A → +B

B → +

② Gramática

S → del S | <cl> | dA | P | B

A → dA | <A> | B

A' → dA'' | B

F → <A> | B → dA'' | B → cuando estoy en S1045 (Nº de matr.) interesa poner los num
de 6 dígitos
B → dE | <E> | B → cuando tengo nombres, si quisiera BaaB, podría tener AAAARRIA y no puedo tener
números entre letras

E → dE | B
F → dF | B
G → dG | B

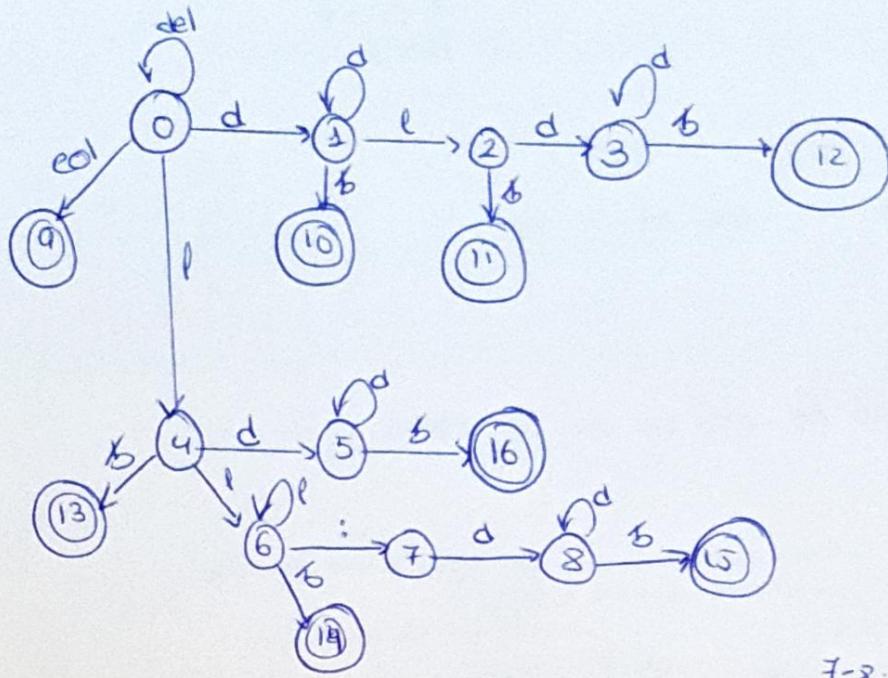
fin de linea → cosas que empiezan por dígito Luum matrícula
empiezan x letras <Nombre>
espacio en blanco <Nombre>

l: letras
d: dígitos

④ lo mejor es crear un controlador para hacer A → dA'
A' → dA'' y así
funciona

→ tengo que comprobar que he leído 4 letras antes de poner :
y que esas letras sean ECTS. Si tuviese AAAA:12, como compruebo
semánticamente que esas letras sean ECTS, pues NO funciona

3) Automata



4) Acciones semánticas:

0-0: Leer

0-1: lexema = d; cont = 1; Leer;

1-1: lexema = lexema \oplus d; cont ++; Leer

1-10: If contador <> 6 then error ("El n° de matrícula debe tener 6 dígitos")
Else Gen-Token (NMat, lexema); Leer;

1-2: letra = l; lexema = lexema \oplus l; Leer;

2-11: If cont <> 8 then error ("El DNI debe tener 8 dígitos")
Else Gen-Token (DNI, lexema); Leer;

2-3: lexema = lexema \oplus d; cont2 = 1; Leer;

3-3: lexema = lexema \oplus d; cont2 ++; Leer

3-12: If cont <> 2 OR cont2 <> 3 then error ("El n° de dígitos del n° mat es erróneo")
Else if letra <> "H" then error ("La letra del n° matrícula es errónea")
Else Gen-Token (NMat, lexema); Leer

0-9: Gen-Token (eol, -); Leer

0-4: lexema = l; Leer

4-5: lexema = lexema \oplus d; cont = 1; Leer

5-5: lexema = lexema \oplus d; cont ++; Leer

5-16: If cont <> 4 then error ("El n° de matrícula debe tener 4 dígitos")
Else Gen-Token (NMat, lexema); Leer

4-13: Gen-Token (Palabra, lexema); Leer

4-6: lexema = lexema \oplus l; Leer

6-6: lexema = lexema \oplus l; Leer

6-7: Leer

7-8: numC = valorA(d); Leer

8-8: numC = numC * 10 + valorA(d); Leer

8-15: If lexema <> "ECTS" then error ("Fallo de lectura")

Else if numC = 0 OR numC > 61 then
error ("Nº créditos incorrecto")

6-14: Gen-Token (Creditos, NumC); Leer

lexema = lexema \oplus l; Leer

lexema = lexema \oplus l; Leer