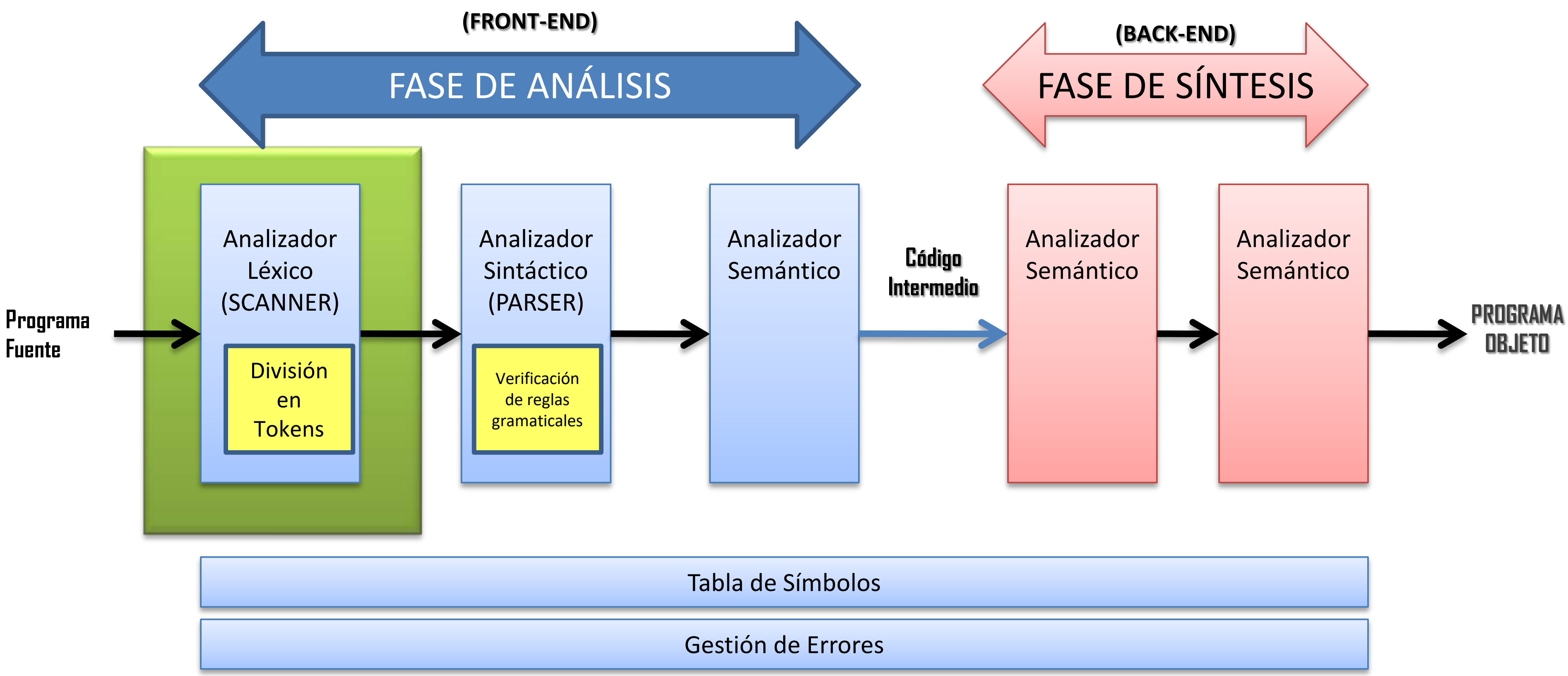
Sesión 2
Analizador Léxico
Lenguajes y Autómatas

Antonio Moratilla Ocaña

Resumen del tema

- Objetivo:
  - Establecer dónde encajan las Expresiones Regulares dentro de los Analizadores Léxicos
  - Introducir qué es un lenguaje formal y cómo influye en el analizador léxico.
  - Introducir el concepto de Autómata Finito como alternativa para la interpretación de cadenas.

Posición en el diagrama



Retomando el hilo...

- Decíamos ayer...
  - Que los analizadores léxicos cogen una entrada y la dividen en TOKENS.
  - Que los TOKENS responden a algún lenguaje o esquema.
  - Ya sabemos qué es una expresión regular.
  - Sabemos que con una expresión regular (patrón) somos capaces de decir si una palabra (lexema) es válida o no para la expresión regular, generando un token.
  - Y dejamos ”colgada” la definición de Lenguaje como un conjunto de palabras formadas con un alfabeto... (que tenía toda la pinta de ser eso para lo que utilizamos los tokens)....
- Y ahora, la continuación...

Conceptos básicos

- **Lenguajes** Un lenguaje es cualquier subconjunto del universo sobre algún alfabeto, es decir, $L \subset W(\Sigma)$ o también $L \subset W^*$.
  - Lenguajes triviales
    - $L = \varnothing$ es el lenguaje vacio (que no contiene ninguna palabra), $|L| = 0$
    - $L = \{\varepsilon\}$ es el lenguaje que solamente contiene la palabra vacio, $|L| = 1$

    son independientes del alfabeto y por eso son lenguajes sobre cualquier alfabeto.
  - Sea = {a, b }
    - $L_1 = \{\varepsilon, a, b\}$
    - $L_{ab} = \{a^n b^n \mid n \;\; IN\}$ es decir, el lenguaje que contiene todas las palabras con un número de a_s seguidos por el mismo número de b_s.
    - $L_{pal} = \{ww^R \mid w \in W^*\}$ es decir, palíndromos

Si $|L| < \infty$ para un lenguaje $L \in W^*$, entonces se denomina L como lenguaje finito.

Conceptos básicos

- **Lenguajes (II)**

  Operaciones sobre/con lenguajes

  Sean L; L_1; L_2; $L_3 \subset W^*$ lenguajes (igual para W(Σ)) las operaciones que se pueden realizar sobre estos lenguajes son:

  – Unión
  – Intersección
  – Complemento
  – Diferencia
  – Concatenación
  – Potencia
  – Clausura positiva
  – Clausura (de Kleene)
  – Reflexión (o inverso)
  – Homomorfismo

Conceptos básicos

Producciones y Derivaciones .

Definimos algunas notaciones para describir reglas de sustitución, es decir, como derivar una palabra con las producciones de la gramática.

- Una producción p es una dupla (pareja) de un conjunto cartesiano sobre dos universos (que pueden ser el mismo), es decir, $p = (A,B) \in W^*_1 \times W^*_2$
- Sea (A,B) una producción, en vez de duplas también escribimos: $A \rightarrow B$.
- Un conjunto de producciones se llama sistema de producciones (o sistema de reglas).
- Una derivación directa $v \rightarrow w$ es una conversión de una palabra en otra aplicando una producción, es decir,
  - Sea $v = aAb$ una palabra, y sea $A \rightarrow B$ una producción,
  - se puede derivar la palabra w = aBb directamente desde v , sustituyendo la subpalabra A por la palabra B como indica la producción.
- Una derivación $v \rightarrow * w$ es una secuencia de derivaciones directa aplicando sucesiva-mente producciones de un sistema. La longitud de una derivación es el número de producciones aplicadas.

Conceptos básicos

Producciones y Derivaciones . Ejemplos.

1. Sean 000 → 010 y 10 → 01 dos producciones.
   - Desde v = 1000 se puede derivar
     - w_1 = 1010 aplicando la primera producción, y
     - w_2 = 0100 aplicando la segunda.

2. Sean 000 → 010 y 10 → 01 dos producciones.
   - Desde v = 1000 se puede derivar
     - w_1 = 0011, es decir, v →* w_1 aplicando v = 1000 → 1010 → 0110 → 0101 → 0011 = w_1
     - w_2 = 0001 aplicando v = 1000 → 0100 → 0010 → 0001 = w_2

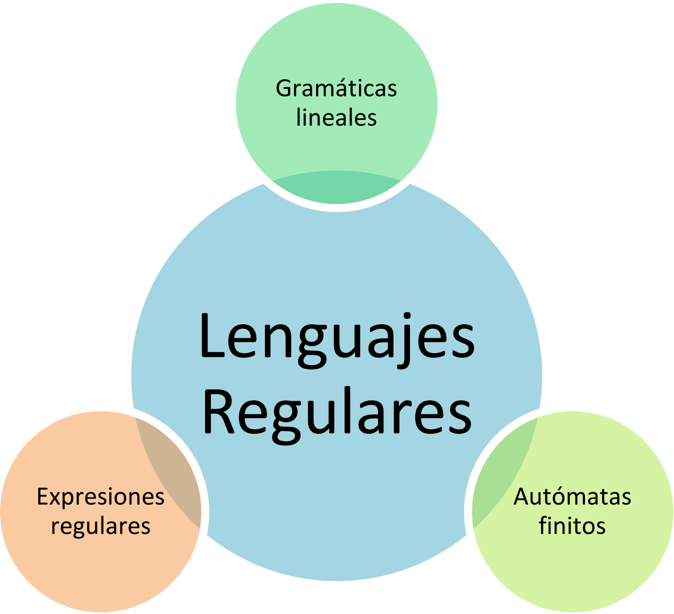
   En el primer caso la longitud de la derivación es 4, en el segundo caso 3.

Dado un sistema de producciones, si sustituimos siempre la primera posibilidad a la izquierda de la palabra de partida, se llama una <u>derivación más a la izquierda</u>, e igual, si sustituimos siempre la primera posibilidad a la derecha de la palabra de partida, se llama una <u>derivación más a la derecha</u>.

Definición de Lenguaje Regular

Un lenguaje regular describe una familia de tokens que se ajustan aun determinado patrón léxico.

Además de la declaración extensiva (sólo viable en los lenguajes finitos) existen 3 diferentes maneras de definir formalmente un lenguaje regular.



Especificación de Analizadores Léxicos

- **Definiciones Regulares.** Nombre que se da a una expresión regular por conveniencia, definiéndola como si fuera un símbolo:
  - Letra →[a-zA-Z]
  - Dígito →0|1|2|3|4|5|6|7|8|9
  - Dígito →[0-9]
  - Identificador →Letra(Letra|Digito)*
  - Dígitos →Digito+
  - Fracción →(.Dígitos)?
  - Exponente →(E(+|-)?Dígitos)?
  - NúmeroReal →Digito+(.Digito+)?
  - NúmeroReal → Dígitos Fracción Exponente

Pensemos...

- Las definiciones regulares hacen uso de las expresiones regulares con nombre.
- Las definiciones regulares permiten algo importante: Reutilización, composición y recursividad del uso de expresiones regulares
- El problema: Las expresiones regulares están bien definidas de forma atómica, pero ¿cómo definimos algo más complejo como las definiciones regulares? ¿Cuándo pasamos de un patrón a otro? ¿cómo? ¿podemos refinar la definición de patrones de paso?
- La alternativa: **Autómatas Finitos**

Especificación de Analizadores Léxicos

- **Un Autómata Finito** reconoce una cadena de entrada si consumidos todos los caracteres de la misma acaba en un estado final de aceptación.
- Autómata finito determinista
  - Un autómata finito determinista es un conjunto

    AFD = (T, Q, f, q, F) donde:

    → T es un alfabeto de símbolos terminales de entrada

    → Q es un conjunto de estados finito no vacío

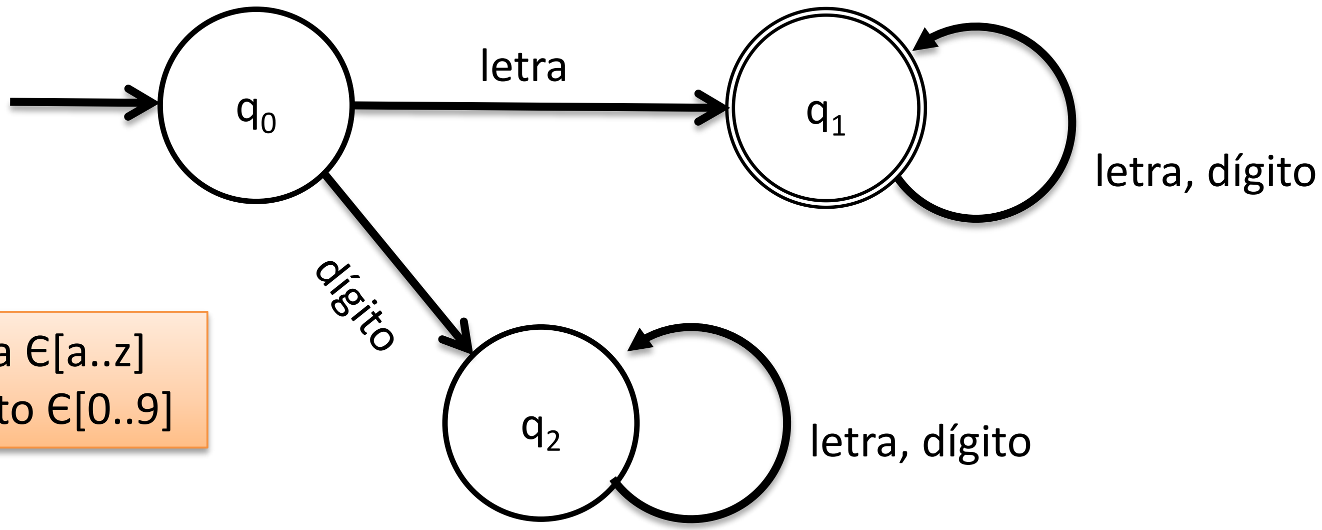
    → f: Q x T → Q es una función de transición

    → q Є Q es un estado inicial

    → F C Q es un subconjunto de estados finales de aceptación

Autómata Finito Determinista

- Representación gráfica
  - Gráficamente, un autómata finito determinista es una colección de estados unidos por arcos, donde para cada estado existe un arco etiquetado con cada elemento de T. El estado inicial tiene una flecha entrante y los estados finales tienen un doble borde



letra Є[a..z]
dígito Є[0..9]

Extensión de los Autómata Finito Determinista

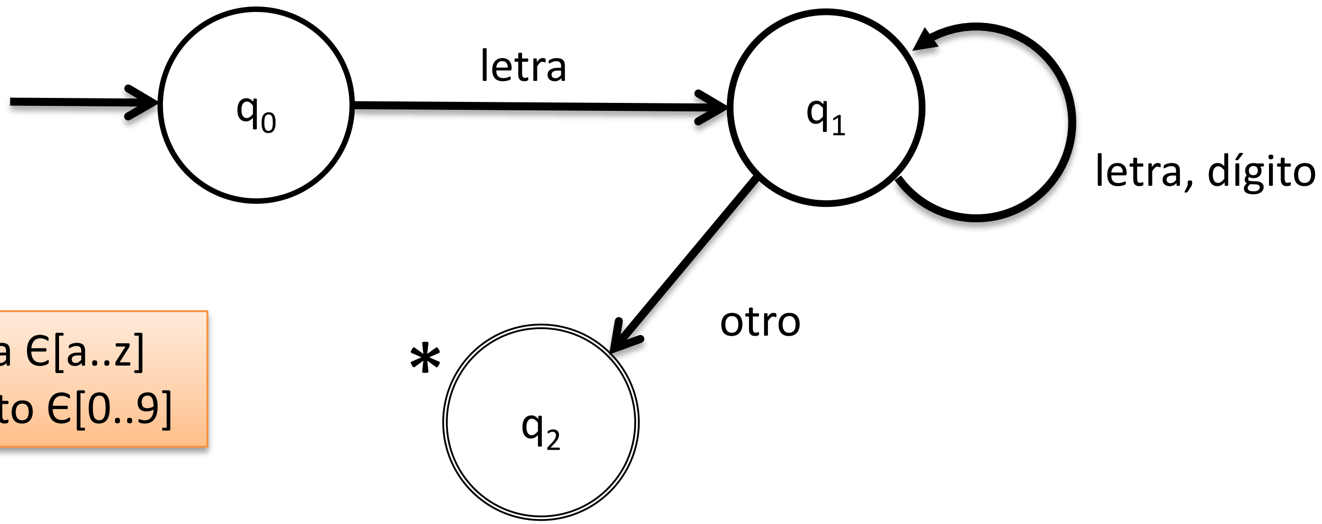
Con objeto de simplificar el trabajo con los AFD, dentro de la teoría de compiladores se utilizan las siguientes extensión de la notación

- Las transiciones ausentes se asumen como errores
- Se omiten los estados de absorción de errores
- Se usa la etiqueta *otro* para representar el resto de alternativas
- Los estados finales paran el proceso y emiten token
- Los estados finales con * recuperan el carácter de tránsito
- Se espera a reconocer el posible token de lexema más largo
  - El estado final ya no es final
  - Al llegar un carácter terminador se pasa a uno final
  - Entonces se emite el token
  - Si es necesario se marca con * el estado final

Especificación de Analizadores Léxicos

Extensión de los Autómata Finito Determinista

- Representación gráfica
  - Gráficamente, un autómata finito determinista es una colección de estados unidos por arcos, donde para cada estado existe un arco etiquetado con cada elemento de T. El estado inicial tiene una flecha entrante y los estados finales tienen un doble borde

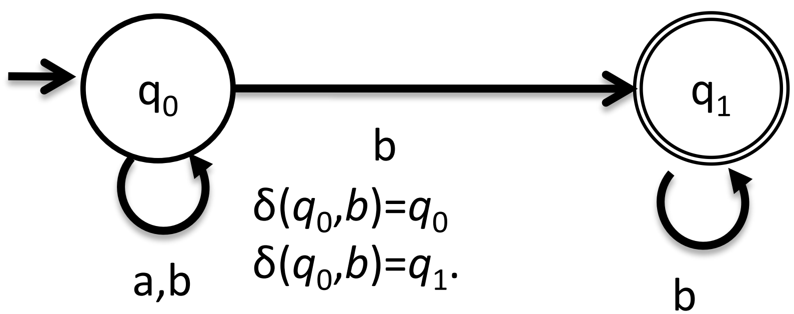


letra ∈[a..z]
dígito ∈[0..9]

Autómata Finito No Determinista (AFND)

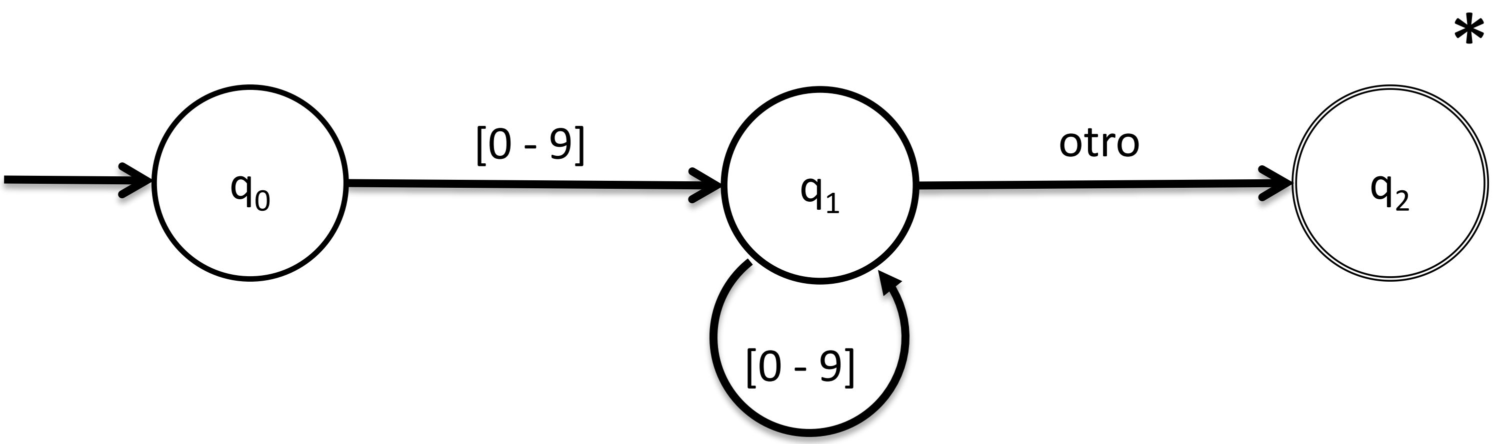
- La construcción sistemática de autómatas genera autómatas finitos no deterministas. Un autómata finito no determinista es un autómata finito donde cada estado
  - Posee al menos un estado $q \in Q$, tal que para un símbolo $a \in \Sigma$ del alfabeto, existe más de una transición $\delta(q,a)$ posible.
  - Puede tener varias transiciones con la misma etiqueta.
    - En un AFND puede darse cualquiera de estos dos casos:
    - Que existan transiciones del tipo $\delta(q,a)=q1$ y $\delta(q,a)=q2$, siendo $q1 \neq q2$;
    - Que existan transiciones del tipo $\delta(q, \varepsilon)$, siendo q un estado no-final, o bien un estado final pero con transiciones hacia otros estados.

AFND que reconoce la expresión regular **$(a|b)^*b^+$**



Diagramas de Transiciones

NúmeroEntero → Digito+



ESTADO	0-9	otro	token	retroceso
q_0	q_1	error	-	-
q_1	q_1	q_2	-	-
q_2	-	-	Num-entero	q_1

Ejercicios

- Defina, grosso modo, el lenguaje necesario para interpretar una sentencia SQL Select mediante Definiciones Regulares.
- Ejemplo:
  - **Select** nombre,apellidos,dni, count(dni) as relacionados **from** personas join estudiantes on personas.dni=estudiantes.dni **where** nacimiento>1980 **group by** apellidos **order by** nacimiento.

```
Select nombre,apellidos,dni,
count(dni) as relacionados
from personas join estudiantes on
personas.dni=estudiantes.dni
where nacimiento>1980
group by apellidos
order by nacimiento.
```

Ejercicios

- Dada la tabla de transiciones para este AF, indicar:
  - A- ¿Es determinista o no determinista? ¿Por qué?
  - B- ¿Cuál es el alfabeto Σ?¿Qué palabras W reconoce?
  - C- Dibuje el grafo de estados

ESTADOS	a	b	ε
0	{0,1}	{0}	Ø
1	Ø	{2}	Ø
2	Ø	{3}	Ø
3	Ø	Ø	Ø

Fuentes

- Para la elaboración de estas transparencias se han utilizado:
  - Transparencias de cursos previos (elaboradas por los profesores Dr. D. Salvador Sánchez, Dr. D. José Luis Cuadrado).
  - Libros de referencia.