

Procesadores de lenguajes

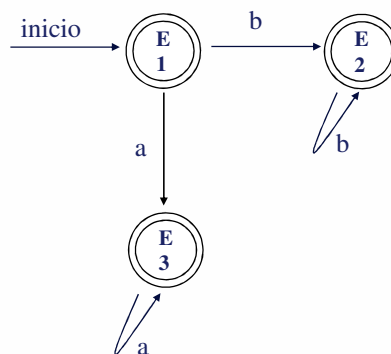
Ejercicios de Análisis Léxico



Salvador Sánchez Alonso, Daniel Rodríguez García
Departamento de Ciencias de la Computación
Universidad de Alcalá

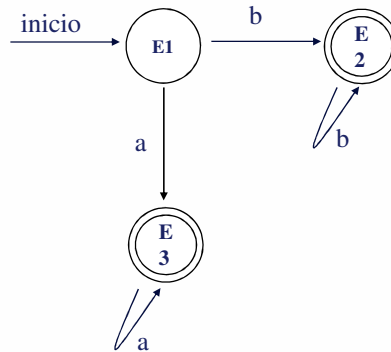


AFD y diagrama para reconocer la expresión: $a^|b^+$*





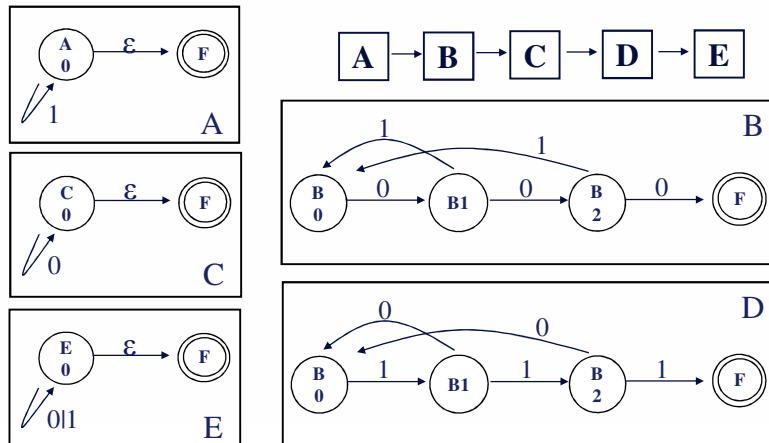
AFD y diagrama para reconocer la expresión: $a^+|b^+$



Procesadores de lenguaje – Ejercicios de Análisis Léxico
Salvador Sanchez, Daniel Rodríguez



“Todas las palabras del alfabeto $\{0, 1\}$ que cumplan que, de contener la subcadena 000, deben tener a continuación la subcadena 111 (no tiene por qué ir a continuación)”

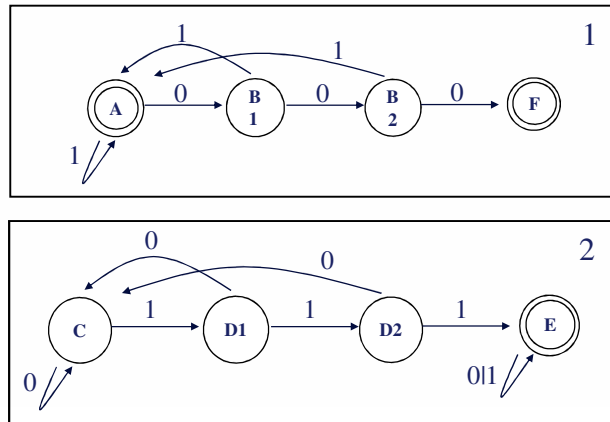


Procesadores de lenguaje – Ejercicios de Análisis Léxico
Salvador Sanchez, Daniel Rodríguez





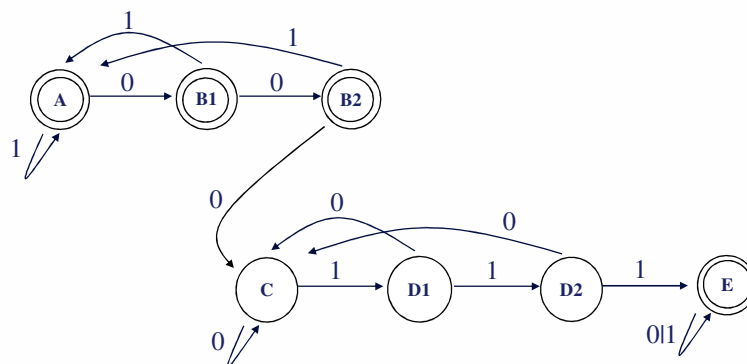
Integrando "A" en "B" (1) y "C" y "E" en "D" (2)...



Procesadores de lenguaje – Ejercicios de Análisis Léxico
Salvador Sanchez, Daniel Rodríguez



Poniéndolo todo junto...

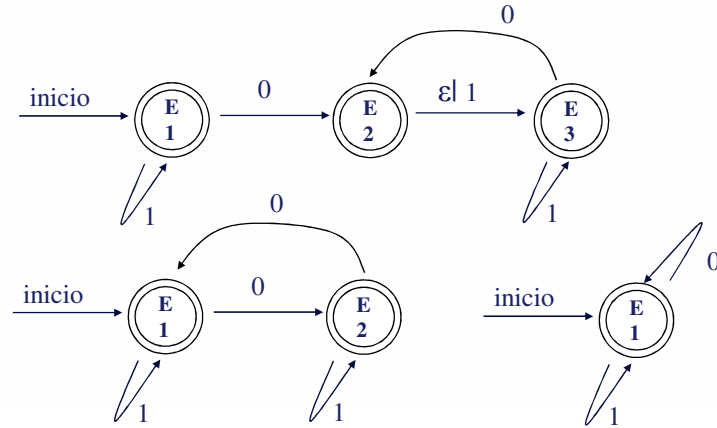


Procesadores de lenguaje – Ejercicios de Análisis Léxico
Salvador Sanchez, Daniel Rodríguez





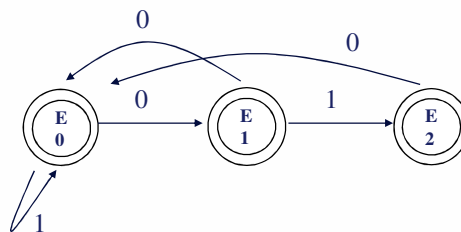
AFD y diagrama para reconocer la expresión: $((\epsilon|0)1^*)^*$



Procesadores de lenguaje – Ejercicios de Análisis Léxico
Salvador Sanchez, Daniel Rodríguez



AFD para reconocer cadenas que no contengan "011"



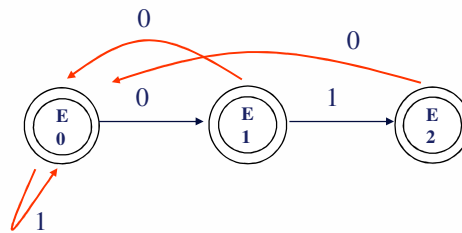
Procesadores de lenguaje – Ejercicios de Análisis Léxico
Salvador Sanchez, Daniel Rodríguez





AFD para reconocer cadenas que no contengan "011"

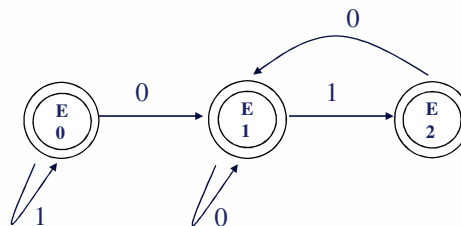
Error: un estado con una transición a múltiples "1" no puede tener como entrada un 0.



Procesadores de lenguaje – Ejercicios de Análisis Léxico
Salvador Sanchez, Daniel Rodríguez



La solución correcta propuesta es:



La expresión regular más simple es: $1^*(0+1)^*$

Procesadores de lenguaje – Ejercicios de Análisis Léxico
Salvador Sanchez, Daniel Rodríguez



→ Ejercicio propuesto

Se desea construir un analizador léxico para una aplicación de cálculo de áreas de figuras geométricas.

Los datos se leen de un archivo de entrada que incluye una lista con el nombre y los puntos de cada figura en el siguiente formato:

```
Triangulo1 (8.02,-4.32), Triangulo2 (8.02,-4.32),
Circulo (3,1.5) ...
```



→ Ejercicio propuesto

Definición de las expresiones regulares:

delimitador → `[\\t\\n]`

letra → `[A-Za-z]`

digito → `[0-9]`

parent_de → `'\''`

parent_iz → `'('`

coma → `','`

coordenada → `digito+ (.digito+)?`

figura → `letra+`

punto → `parent_iz coordenada coma coordenada parent_de`



→ Ejercicio propuesto

Identificar los tokens (códigos y atributos):

Paréntesis derecho: <PARENT_D,->

Paréntesis izquierdo: <PARENT_I,->

Coma: <COMA,->

Coordenada: <COORDENADA,valor>

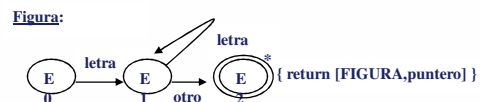
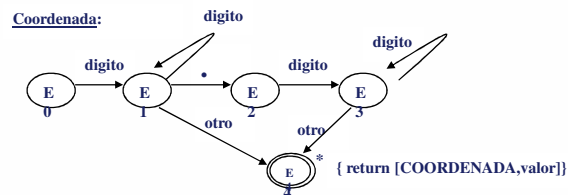
Figura: <FIGURA,direccion_tabla_simbolos>

Punto: <PUNTO,direccion_tabla_simbolos>



→ Ejercicio propuesto

Construir los diagramas de transiciones (AFD) y completar los autómatas con acciones semánticas:



→ Ejercicio propuesto

Se desea construir un analizador léxico para reconocer la declaración de una clase en java (no detallar la composición interna de una sentencia).

Ejemplo:

```
public class nombreClase extends NombreClasePadre{  
    private int x;  
    public void f(){...}  
    ...  
}
```

