



Universidad
de Huelva

Tema 2

Análisis léxico

2.1 Introducción

2.2 Especificación de categorías léxicas

2.3 Autómatas Finitos No Deterministas

2.4 Autómatas Finitos Deterministas

2.5 Implementación de un analizador léxico

2.1 Introducción

2.2 Especificación de categorías léxicas

2.3 Autómatas Finitos No Deterministas

2.4 Autómatas Finitos Deterministas

2.5 Implementación de un analizador léxico

2.1.1 Características del análisis léxico

- Lee caracteres.
- Produce componentes léxicos (tokens).
- Filtra comentarios.
- Filtra separadores múltiples (espacios, tabuladores y saltos de línea).
- Lleva el contador de línea y columna del texto fuente.
- Genera errores en caso de que la entrada no corresponda a ninguna categoría léxica.

2.1.2 Algunas definiciones

- Categoría léxica:
 - Tipo de símbolo elemental del lenguaje fuente, (identificadores, palabras clave, constantes numéricas, operadores, ...).
- Componente léxico (token):
 - Elemento perteneciente a una categoría léxica.
- Atributos de un componente:
 - Información del componente necesaria en etapas posteriores del análisis (valor de la constante, nombre de una variable, ...).
- Lexema:
 - Cadena de caracteres correspondiente al componente léxico.

2.1.3 Categorías léxicas más habituales

- Palabras clave:
 - Palabras con un significado especial en el lenguaje
(if, then, else, for, while, do, switch, case, ...)
 - Suelen ser palabras reservadas.
- Identificadores:
 - Nombres de variables, de constantes, de funciones, de tipos...
(media, valor0, i1, _PI, ...)
- Operadores:
 - Símbolos que identifican operaciones aritméticas y lógicas.
(+ , - , * , / , % , = , ++ , -- , += , -= , *= , /= , == , != , && , || , & , | , ...)

2.1.3 Categorías léxicas más habituales

- Constantes numéricas:
 - Literales que especifican valores numéricos
(325, 3.141592, 0xA3F2, 0.2e+3)
- Constantes de carácter o cadenas:
 - Literales con el valor de un carácter o de una cadena
('z', '\n', "ejemplo de cadena", ...)
- Símbolos especiales:
 - Separadores, delimitadores, terminadores, etc.
({, }, [,], (,), ;, ,)

2.1.3 Categorías léxicas más habituales

- Blancos:
 - Caracteres de separación de componentes léxicos (espacios, tabuladores, saltos de línea)
 - El análisis léxico se limita a suprimirlos.
- Comentarios:
 - Información para el lector del programa.
(/* comentario multilínea */, // comentario de una línea \n, /** comentario para la documentación */)
 - El análisis léxico los elimina.
- Fin de entrada:
 - Componente ficticio que indica el final de lectura.

2.1 Introducción

2.2 Especificación de categorías léxicas

2.3 Autómatas Finitos No Deterministas

2.4 Autómatas Finitos Deterministas

2.5 Implementación de un analizador léxico

2.2.1 Gramáticas regulares

- Tipos de producciones:

$$\langle A \rangle \rightarrow a \langle B \rangle$$

$$\langle A \rangle \rightarrow a$$

$$\langle A \rangle \rightarrow \lambda$$

- Ejemplo: números enteros

$$\langle S \rangle \rightarrow 0 \langle S \rangle$$

$$\langle S \rangle \rightarrow 5 \langle S \rangle$$

$$\langle S \rangle \rightarrow 0$$

$$\langle S \rangle \rightarrow 5$$

$$\langle S \rangle \rightarrow 1 \langle S \rangle$$

$$\langle S \rangle \rightarrow 6 \langle S \rangle$$

$$\langle S \rangle \rightarrow 1$$

$$\langle S \rangle \rightarrow 6$$

$$\langle S \rangle \rightarrow 2 \langle S \rangle$$

$$\langle S \rangle \rightarrow 7 \langle S \rangle$$

$$\langle S \rangle \rightarrow 2$$

$$\langle S \rangle \rightarrow 7$$

$$\langle S \rangle \rightarrow 3 \langle S \rangle$$

$$\langle S \rangle \rightarrow 8 \langle S \rangle$$

$$\langle S \rangle \rightarrow 3$$

$$\langle S \rangle \rightarrow 8$$

$$\langle S \rangle \rightarrow 4 \langle S \rangle$$

$$\langle S \rangle \rightarrow 9 \langle S \rangle$$

$$\langle S \rangle \rightarrow 4$$

$$\langle S \rangle \rightarrow 9$$

2.2.1 Gramáticas regulares

- Notación simplificada

$$\langle S \rangle \rightarrow 0 \langle S \rangle \mid 1 \langle S \rangle \mid 2 \langle S \rangle \mid 3 \langle S \rangle \mid 4 \langle S \rangle \mid 5 \langle S \rangle \mid 6 \langle S \rangle \mid 7 \langle S \rangle \mid 8 \langle S \rangle \mid 9 \langle S \rangle$$
$$\langle S \rangle \rightarrow 0 \mid 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9$$

- Ejemplo: palabra clave “protected”

$$\langle S \rangle \rightarrow p \langle A1 \rangle \quad \langle A1 \rangle \rightarrow r \langle A2 \rangle \quad \langle A2 \rangle \rightarrow o \langle A3 \rangle \quad \langle A3 \rangle \rightarrow t \langle A4 \rangle$$
$$\langle A4 \rangle \rightarrow e \langle A5 \rangle \quad \langle A5 \rangle \rightarrow c \langle A6 \rangle \quad \langle A6 \rangle \rightarrow t \langle A7 \rangle \quad \langle A7 \rangle \rightarrow e \langle A8 \rangle$$
$$\langle A8 \rangle \rightarrow d$$

- Ejemplo: identificador

$$\langle S \rangle \rightarrow a \langle A \rangle \mid b \langle A \rangle \mid c \langle A \rangle \mid d \langle A \rangle \mid \dots \mid A \langle A \rangle \mid B \langle A \rangle \mid C \langle A \rangle \mid D \langle A \rangle \dots$$
$$\langle A \rangle \rightarrow a \langle A \rangle \mid \dots \mid Z \langle A \rangle \mid 0 \langle A \rangle \mid \dots \mid 9 \langle A \rangle \mid a \mid \dots \mid Z \mid 0 \mid \dots \mid 9$$

2.2.2 Expresiones regulares

- Concatenación de lenguajes L y M
 - $LM = \{ xy \mid x \in L \wedge y \in M \}$
- Notación:
 - $L^k = L$ concatenado consigo mismo k-1 veces
 - $L^0 = \{ \lambda \}$
 - $L^1 = L$
- Clausura de L
 - Conjunto de cadenas que pueden obtenerse concatenando un número arbitrario de cadenas de L
 - $L^* = \bigcup_{k=0} L^k$

2.2.2 Expresiones regulares

- Ejemplos:
 - $L = \{ "a", "b", "c" \}$
 - $M = \{ "x", "y", "z" \}$
 - $LM = \{ "ax", "ay", "az", "bx", "by", "bz", "cx", "cy", "cz" \}$
 - $L^3 = \{ "aaa", "aab", "aac", "aba", "abb", "abc", "aca", "acb", "acc", "baa", "bab", "bac", "bba", "bbb", "bbc", "bca", "bcb", "bcc", "caa", "cab", "cac", "cba", "cbb", "cbc", "cca", "ccb", "ccc" \}$
 - $L^* = \{ \lambda, "a", "b", "c", "aa", "ab", "ac", "ba", "bb", "bc", "ca", "cb", "cc", "aaa", \dots \}$

2.2.2 Expresiones regulares

- Expresión regular ϕ : lenguaje vacío $L_\phi = \phi$
- Expresión regular λ : lenguaje de la cadena vacía $L_\lambda = \{ \lambda \}$
- Expresión regular a (con $a \in \Sigma$): lenguaje $L_a = \{ a \}$
- Expresión regular rs : lenguaje $L_r L_s$
- Expresión regular $r|s$: lenguaje $L_r \cup L_s$
- Expresión regular r^* : lenguaje $(L_r)^*$

2.2.2 Expresiones regulares

- Ejemplos:

- $(\text{"a"} \mid \text{"b"})^* = \{ \lambda, \text{"a"}, \text{"b"}, \text{"aa"}, \text{"ab"}, \text{"ba"}, \text{"bb"}, \text{"aaa"}, \text{"aab"}, \text{"aba"}, \text{"abb"}, \dots \}$
- $n^{\circ} \text{ enteros} : (\text{"0"} \mid \text{"1"} \mid \text{"2"} \mid \text{"3"} \mid \text{"4"} \mid \text{"5"} \mid \text{"6"} \mid \text{"7"} \mid \text{"8"} \mid \text{"9"})^*$
- $n^{\circ} \text{ enteros} : ((\text{"1"} \mid \text{"2"} \mid \text{"3"} \mid \text{"4"} \mid \text{"5"} \mid \text{"6"} \mid \text{"7"} \mid \text{"8"} \mid \text{"9"}) (\text{"0"} \mid \text{"1"} \mid \text{"2"} \mid \text{"3"} \mid \text{"4"} \mid \text{"5"} \mid \text{"6"} \mid \text{"7"} \mid \text{"8"} \mid \text{"9"})^* \mid \lambda)$
- $n^{\circ} \text{ reales} : (\text{"0"} \mid \text{"1"} \mid \text{"2"} \mid \text{"3"} \mid \text{"4"} \mid \text{"5"} \mid \text{"6"} \mid \text{"7"} \mid \text{"8"} \mid \text{"9"})^* (\text{"."} (\text{"0"} \mid \text{"1"} \mid \text{"2"} \mid \text{"3"} \mid \text{"4"} \mid \text{"5"} \mid \text{"6"} \mid \text{"7"} \mid \text{"8"} \mid \text{"9"})^* \mid \lambda)$

2.2.2 Expresiones regulares

- Abreviaturas (expresiones regulares extendidas):
 - $["a", "f", "j"] = ("a" | "f" | "j")$ (los corchetes representan opciones)
 - $["0"-"9"] = ["0", "1", "2", "3", "4", "5", "6", "7", "8", "9"]$
(el guión indica intervalo)
 - $["0"-"9", "a"-"z", "A"-"Z", "_"]$
(puede aparecer más de un guión entre corchetes)
 - $r+ = rr^*$ (el signo + indica clausura positiva)
 - $r? = (r | \lambda)$ (el signo ? indica opcionalidad)
 - $\sim["0"-"9"] =$ cualquier carácter excepto un dígito
(el símbolo ~ significa caracteres excluidos)
 - $\sim[] =$ cualquier carácter

2.2.2 Expresiones regulares

- Ejemplos:
 - Identificadores: `["a"-"z","A"-"Z","_"]["a"-"z","A"-"Z","0"-"9","_"]*`
 - Números enteros: `(["1"-"9"] ["0"-"9"]* | "0")`
 - Números reales: `["0"-"9"]+ ('.' ["0"-"9"]+)?`
 - Números hexadecimales: `"0x" ["0"-"9","a"-"f","A"-"F"]+`
 - Comentarios en C: `"/*" (~["*"] | ("*")+ ~["*","/"])* ("*")+ "/"`
 - Palabras clave: `"if" | "then" | "else"`

2.1 Introducción

2.2 Especificación de categorías léxicas

2.3 Autómatas Finitos No Deterministas

2.4 Autómatas Finitos Deterministas

2.5 Implementación de un analizador léxico

2.3.1 Reconocedores de lenguajes regulares

- Un reconocedor de lenguaje es un programa que toma como entrada una cadena y devuelve verdadero o falso en función de si la cadena pertenece o no al lenguaje.
- Los lenguajes regulares (descritos por gramáticas regulares o por expresiones regulares) pueden ser reconocidos por medio de *Autómatas Finitos* (*Finite State Machines*)
- Existen dos tipos de Autómatas Finitos:
 - Autómatas Finitos No Deterministas (AFN)
 - Autómatas Finitos Deterministas (AFD)

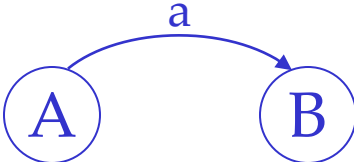
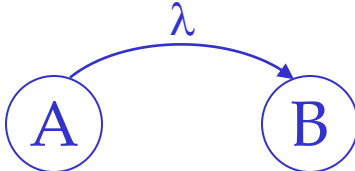
2.3.2 Definición de AFN

- Quintupla: (Q, Σ, E, q_0, F)
 - Q : Conjunto finito de estados
 - Σ : Alfabeto de símbolos
 - E : Conjunto de arcos $(Q \times \Sigma \times Q)$ que representan transiciones entre estados ante un determinado símbolo (incluido λ)
 - q_0 : Estado inicial ($q_0 \in Q$)
 - F : Conjunto de estados finales ($F \subset Q$)

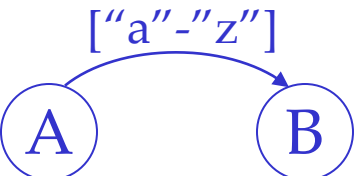
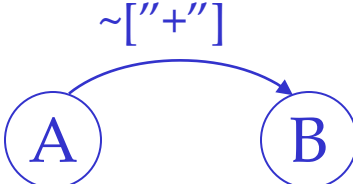
2.3.2 Definición de AFN

- Representación gráfica:

– Estado: 

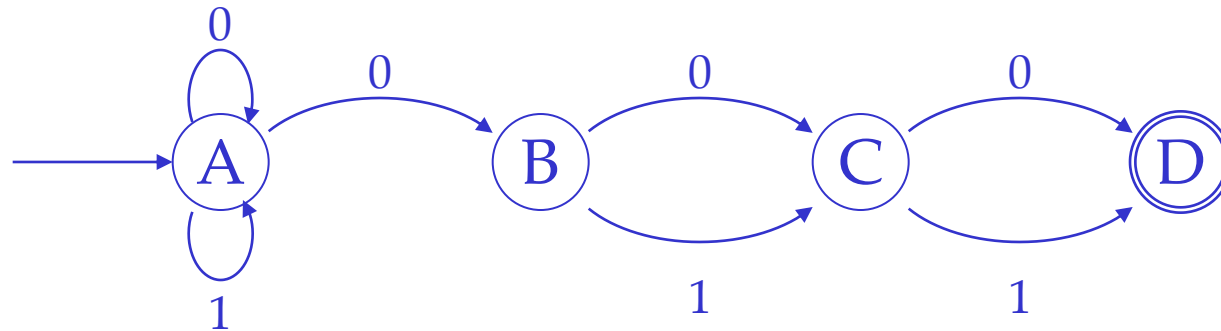
– Arco:  

– Estado final: 

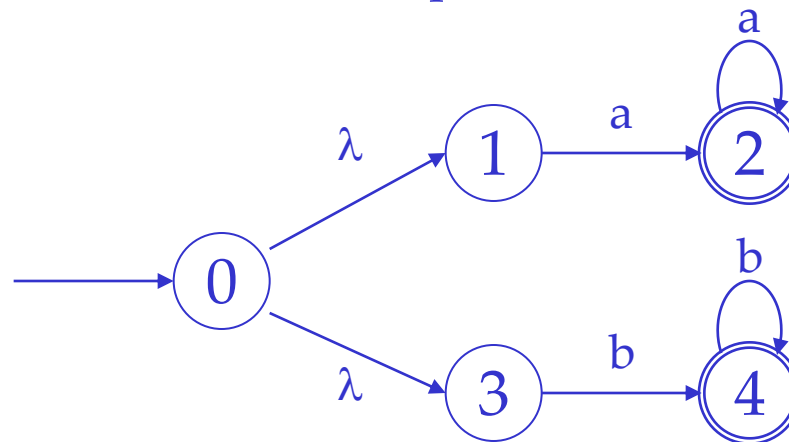
– Arcos agrupados:  

2.3.2 Definición de AFN

- Ejemplo: binarios cuyo tercer último dígito es un 0



- Ejemplo: cadenas formadas por 'a' o cadenas por 'b'



2.3.3 Funcionamiento de un AFN

- Camino: lista de estados en la que a partir de cada estado se puede llegar al siguiente por medio de una transición del autómata.
- Cadena aceptada x : si existe un camino que parta de q_0 , acepte la entrada x y termine en un estado final.
- Lenguaje reconocido por el autómata: conjunto de cadenas aceptadas
- En un AFN, dada una cadena pueden existir varios caminos. Una cadena es aceptada si al menos uno de los caminos conduce a un estado final

2.3.3 Funcionamiento de un AFN

- Dada una cadena x , se puede considerar el estado del AFN (macroestado) como el conjunto de estados a los que se puede llegar partiendo del estado inicial y realizando transiciones siguiendo la cadena x
- Una cadena es reconocida por el AFN si su macroestado contiene algún estado final
- Ejemplo:
 - Binarios cuyo tercer último dígito es un 0
 - Cadena “01001”
 - Macroestado: { A, C, D }

2.3.3 Funcionamiento de un AFN

- Dado un conjunto de estados E , se denomina *clausura- λ (E)* al conjunto de estados formado por todos los estados de E más aquellos estados a los que se pueda acceder desde E mediante transiciones λ .
- El macroestado inicial del AFN es *clausura- λ (q_0)*.
- Partiendo de un macroestado E_i , el resultado de una transición con el símbolo ' a ' es un macroestado $E_{i+1} = \text{clausura-}\lambda(E')$, donde E' está formado por todos aquellos estados que se puedan alcanzar con transiciones con el símbolo ' a ' a partir de alguno de los estados de E_i .

2.3.4 Creación de un AFN a partir de una gramática regular

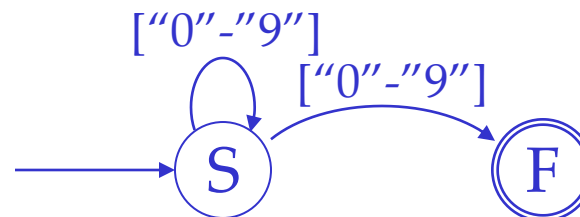
- Cada símbolo no terminal genera un estado
- El símbolo inicial genera el estado inicial
- Se añade un estado final $\langle F \rangle$
- Cada producción de tipo $\langle A \rangle \rightarrow a \langle B \rangle$ genera una transición del estado $\langle A \rangle$ al $\langle B \rangle$ con el símbolo 'a'
- Cada producción del tipo $\langle A \rangle \rightarrow a$ genera una transición del estado $\langle A \rangle$ al estado final $\langle F \rangle$ con el símbolo 'a'

2.3.4 Creación de un AFN a partir de una gramática regular

- Ejemplo: números enteros

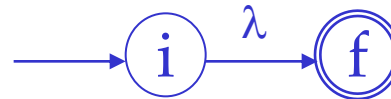
$\langle S \rangle \rightarrow 0 \langle S \rangle$	$\langle S \rangle \rightarrow 5 \langle S \rangle$	$\langle S \rangle \rightarrow 0$	$\langle S \rangle \rightarrow 5$
$\langle S \rangle \rightarrow 1 \langle S \rangle$	$\langle S \rangle \rightarrow 6 \langle S \rangle$	$\langle S \rangle \rightarrow 1$	$\langle S \rangle \rightarrow 6$
$\langle S \rangle \rightarrow 2 \langle S \rangle$	$\langle S \rangle \rightarrow 7 \langle S \rangle$	$\langle S \rangle \rightarrow 2$	$\langle S \rangle \rightarrow 7$
$\langle S \rangle \rightarrow 3 \langle S \rangle$	$\langle S \rangle \rightarrow 8 \langle S \rangle$	$\langle S \rangle \rightarrow 3$	$\langle S \rangle \rightarrow 8$
$\langle S \rangle \rightarrow 4 \langle S \rangle$	$\langle S \rangle \rightarrow 9 \langle S \rangle$	$\langle S \rangle \rightarrow 4$	$\langle S \rangle \rightarrow 9$

- AFN

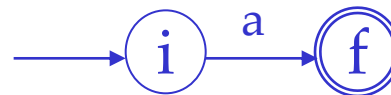


2.3.5 Creación de un AFN a partir de una expresión regular

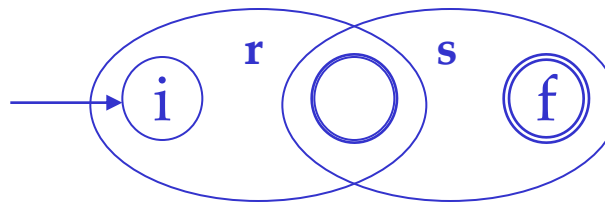
- Expresión regular λ :



- Expresión regular **a**:

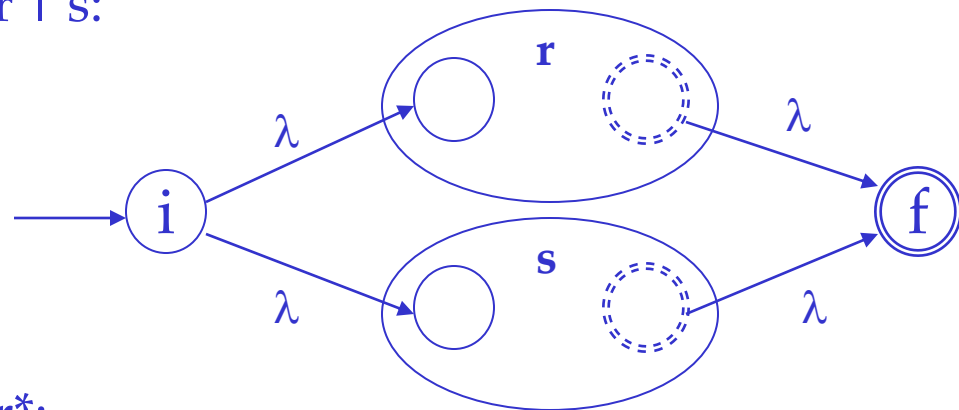


- Expresión regular **rs**:

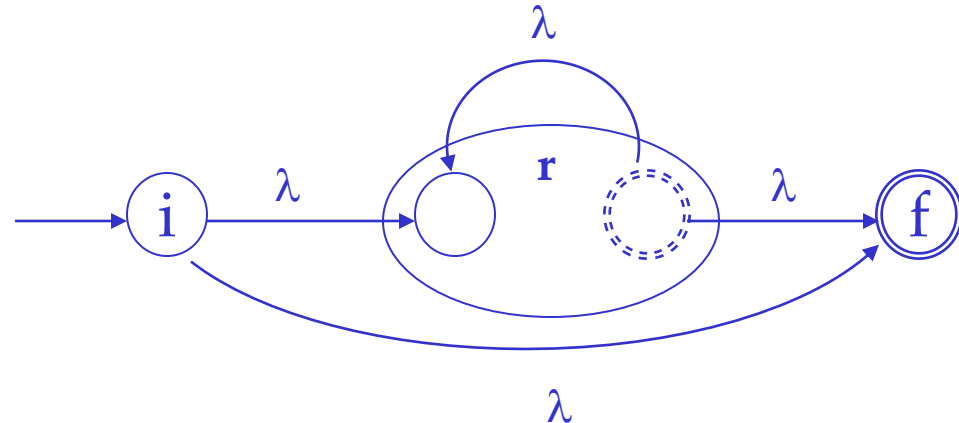


2.3.5 Creación de un AFN a partir de una expresión regular

- Expresión regular $r \mid s$:



- Expresión regular r^* :



2.3.5 Creación de un AFN a partir de una expresión regular

- Propiedades
 - El AFN tiene como máximo el doble de estados que el número de símbolos y operadores de la expresión regular
 - El AFN tiene un estado inicial y un estado final
 - Cada estado tiene tan sólo una transición con un símbolo del alfabeto, o dos transiciones λ .

2.1 Introducción

2.2 Especificación de categorías léxicas

2.3 Autómatas Finitos No Deterministas

2.4 Autómatas Finitos Deterministas

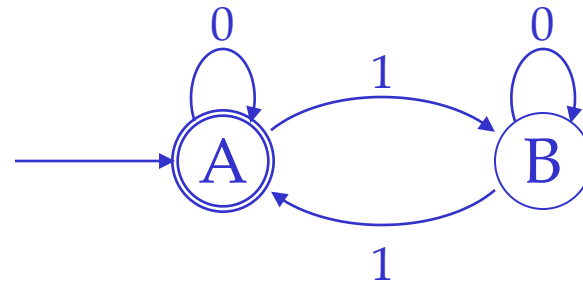
2.5 Implementación de un analizador léxico

2.4.1 Definición de AFD

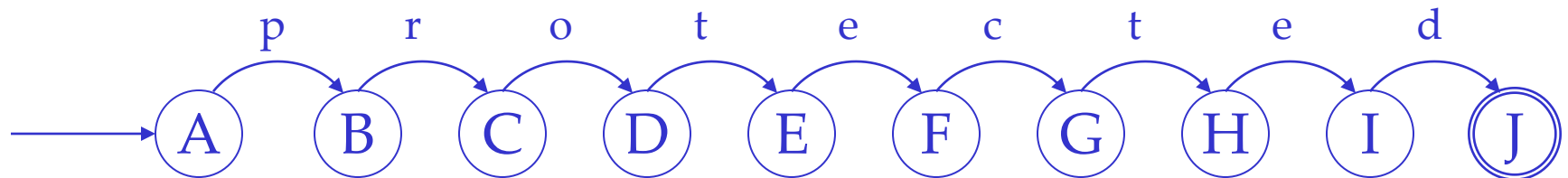
- Quintupla: (Q, Σ, E, q_0, F)
 - Q : Conjunto finito de estados
 - Σ : Alfabeto de símbolos
 - E : Conjunto de arcos $(Q \times \Sigma \times Q)$ que representan transiciones entre estados ante un determinado símbolo
 - q_0 : Estado inicial ($q_0 \in Q$)
 - F : Conjunto de estados finales ($F \subset Q$)
 - No se admiten transiciones λ
 - Condición de determinismo: no existen transiciones que partan del mismo estado con el mismo símbolo

2.4.1 Definición de AFD

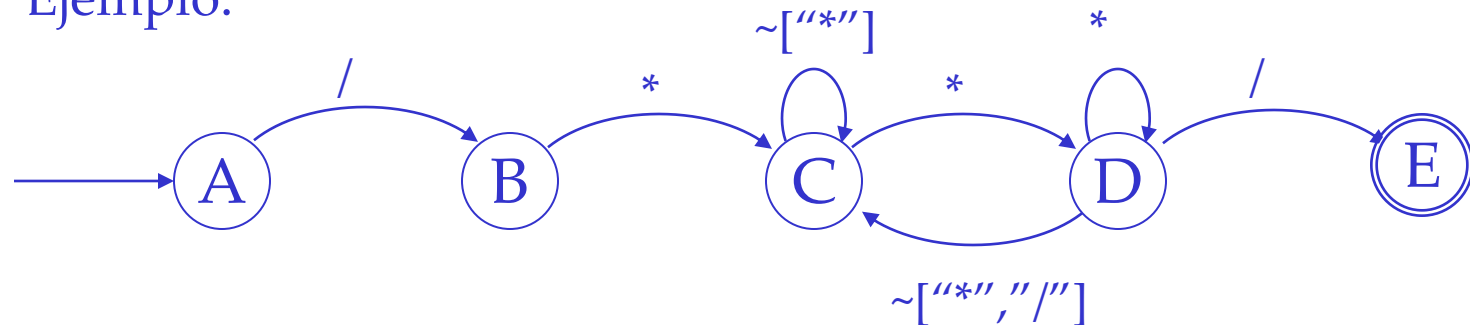
- Ejemplo:



- Ejemplo:



- Ejemplo:



2.4.2 Funcionamiento de un AFD

- Camino: lista de estados en la que a partir de cada estado se puede llegar al siguiente por medio de una transición del autómata.
- Cadena aceptada x : si existe un camino que parta de q_0 , acepte la entrada x y termine en un estado final.
- Lenguaje reconocido por el autómata: conjunto de cadenas aceptadas
- En un AFD sólo existe un camino posible, es decir, dada una cadena, sólo es posible llegar a un estado a partir del estado inicial.

2.4.3 Creación de un AFD a partir de un AFN

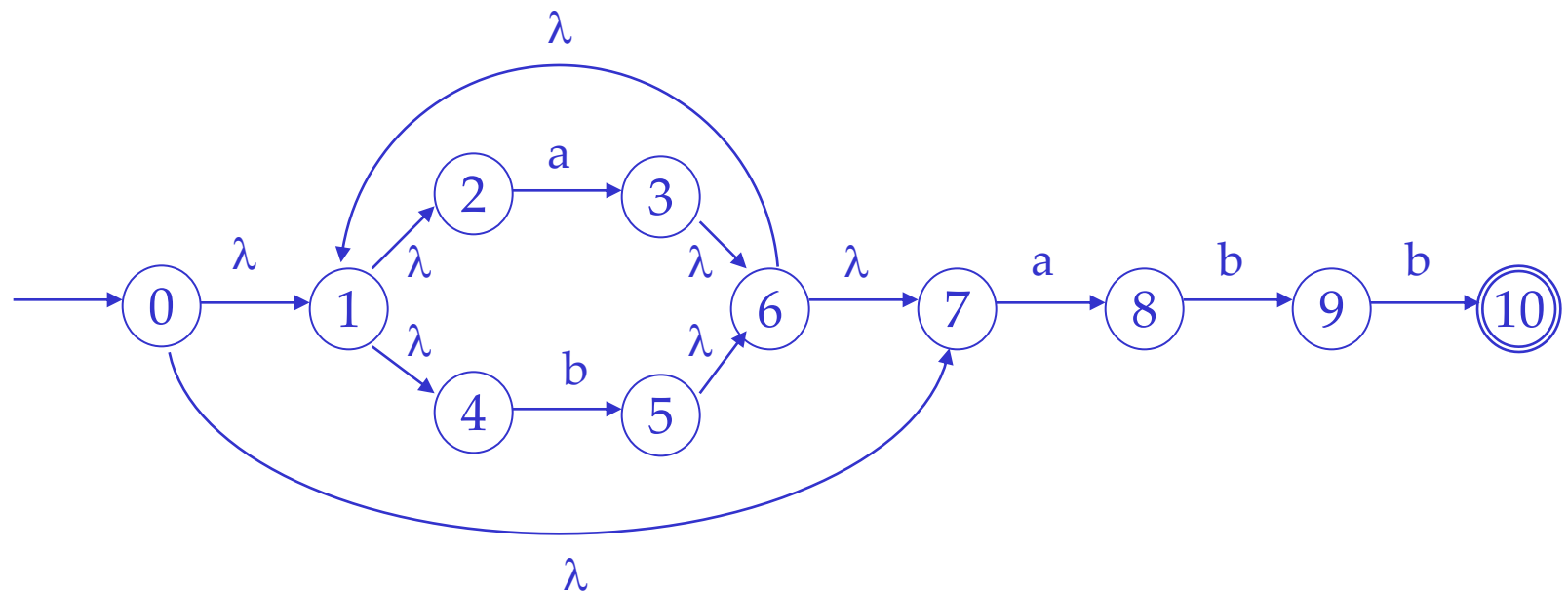
- Los AFD tienen la misma capacidad expresiva que los AFN, es decir, dado un AFN existe un AFD capaz de reconocer el mismo lenguaje
- Cada macroestado del AFN corresponde a un estado del AFD
- Potencialmente, para un AFN de N estados existen 2^N macroestados posibles, aunque la inmensa mayoría son estados inalcanzables

2.4.3 Creación de un AFD a partir de un AFN

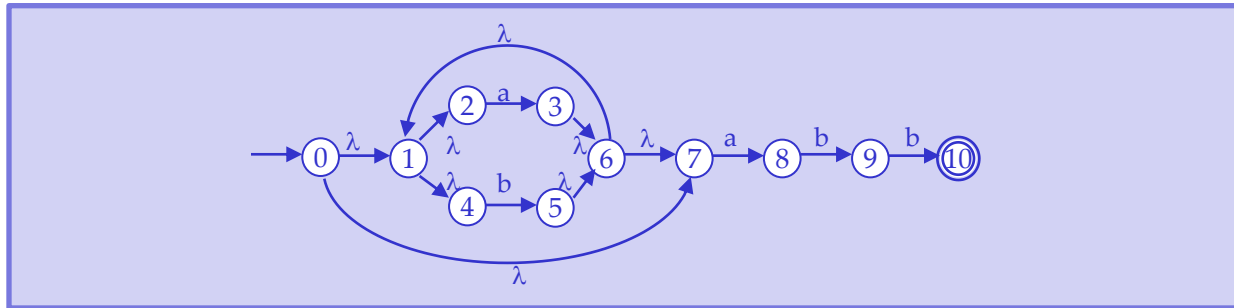
- Algoritmo para construir un AFD a partir de un AFN:
 - Generar el estado inicial del AFD como el macroestado inicial del AFN e incluirlo en una lista de estados por analizar
 - Analizar el primer estado de la lista por analizar:
 - Extraer el primer estado de la lista e introducirlo en la lista de estados analizados
 - Estudiar las transiciones del estado para cada símbolo del alfabeto
 - Si el macroestado correspondiente a una transición no ha aparecido con anterioridad, crear un nuevo estado del AFD correspondiente a dicho macroestado e incluirlo en la lista de estados por analizar
 - Repetir el paso anterior hasta que no queden estados por analizar
 - Los estados finales del AFD son aquellos que correspondan a macroestados que contengan algún estado final del AFN

2.4.3 Creación de un AFD a partir de un AFN

- Ejemplo: $(a \mid b)^* abb$



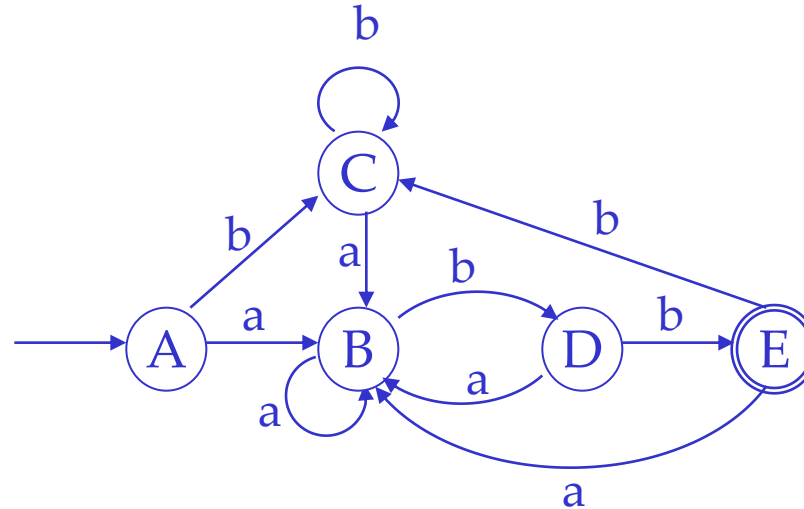
2.4.3 Creación de un AFD a partir de un AFN



- Estado inicial: $A = \{ 0, 1, 2, 4, 7 \}$
- Transición de A con 'a': $B = \{ 1, 2, 3, 4, 6, 7, 8 \}$
- Transición de A con 'b': $C = \{ 1, 2, 4, 5, 6, 7 \}$
- Transición de B con 'a': B
- Transición de B con 'b': $D = \{ 1, 2, 4, 5, 6, 7, 9 \}$
- Transición de C con 'a': B
- Transición de C con 'b': C
- Transición de D con 'a': B
- Transición de D con 'b': $E = \{ 1, 2, 4, 5, 6, 7, 10 \}$ (final)
- Transición de E con 'a': B
- Transición de E con 'b': C

2.4.3 Creación de un AFD a partir de un AFN

- Resultado



2.4.4 Creación de un AFD a partir de una expresión regular

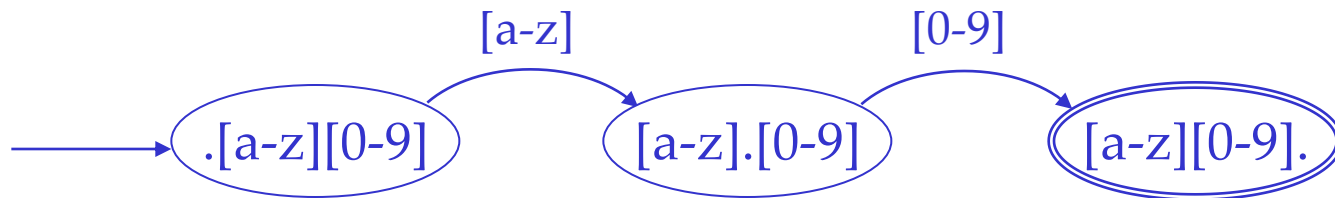
- 1ª opción:
 - Generar un AFN a partir de la expresión regular
 - Generar el AFD a partir del AFN
- 2ª opción:
 - Generar el AFD directamente a partir de la expresión regular

2.4.4 Creación de un AFD a partir de una expresión regular

- Características del algoritmo:
 - Se introduce un punto en la expresión regular para indicar la parte reconocida en cada momento
 - Un estado del autómata está asociado a un conjunto de expresiones regulares con puntos
 - El estado inicial se obtiene colocando el punto al comienzo de la expresión regular
 - Las transiciones de cada estado corresponden al consumo de algún símbolo, y dan lugar al desplazamiento del punto en la expresión regular

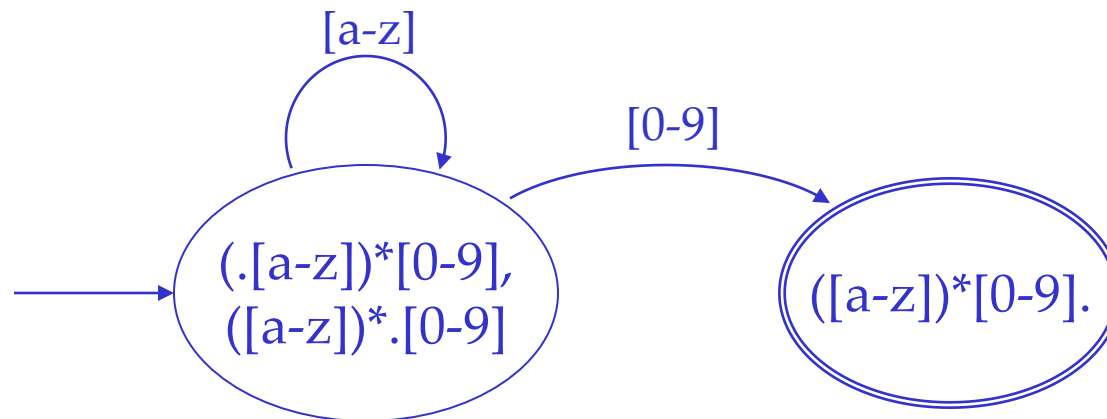
2.4.4 Creación de un AFD a partir de una expresión regular

- Ejemplo: $[a-z][0-9]$
 - Estados: $\{.[a-z][0-9]\}$, $\{[a-z].[0-9]\}$, $\{[a-z][0-9].\}$
 - Autómata:



2.4.4 Creación de un AFD a partir de una expresión regular

- Ejemplo: $([a-z])^*[0-9]$
 - Estados: $\{ ([a-z])^*[0-9], ([a-z])^*[0-9] \cdot \}$
 - Autómata:

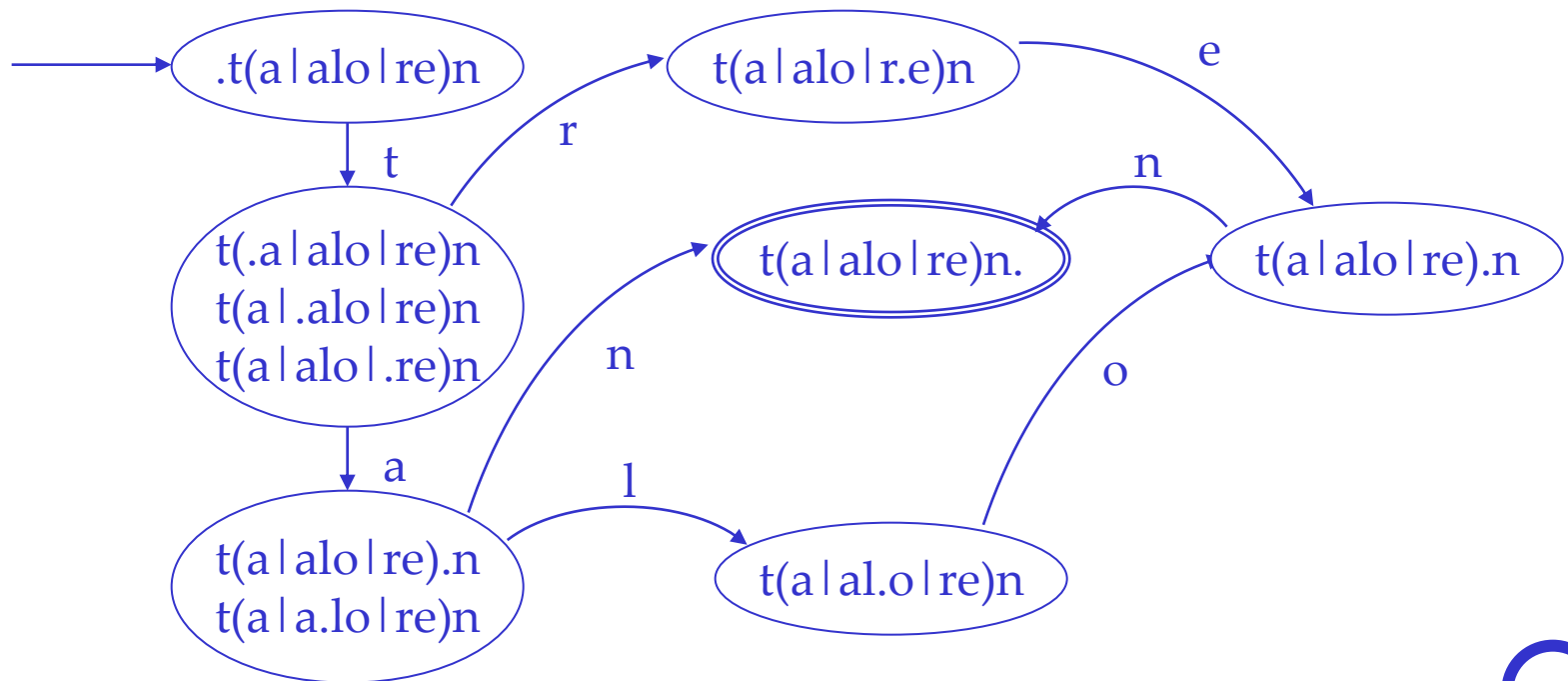


2.4.4 Creación de un AFD a partir de una expresión regular

- Ejemplo: $t(a|alo|re)n$
 - Estados: $\{ .t(a|alo|re)n \},$
 $\{ t(.a|alo|re)n, t(a|.alo|re)n, t(a|alo|.re)n \},$
 $\{ t(a|alo|re).n, t(a|a.lo|re)n \},$
 $\{ t(a|alo|r.e)n \},$
 $\{ t(a|alo|re)n. \},$
 $\{ t(a|al.o|re)n \},$
 $\{ t(a|alo|re).n \}$

2.4.4 Creación de un AFD a partir de una expresión regular

- Ejemplo: $t(a|alo|re)n$
 - Autómata:



2.4.4 Creación de un AFD a partir de una expresión regular

- Formalización:

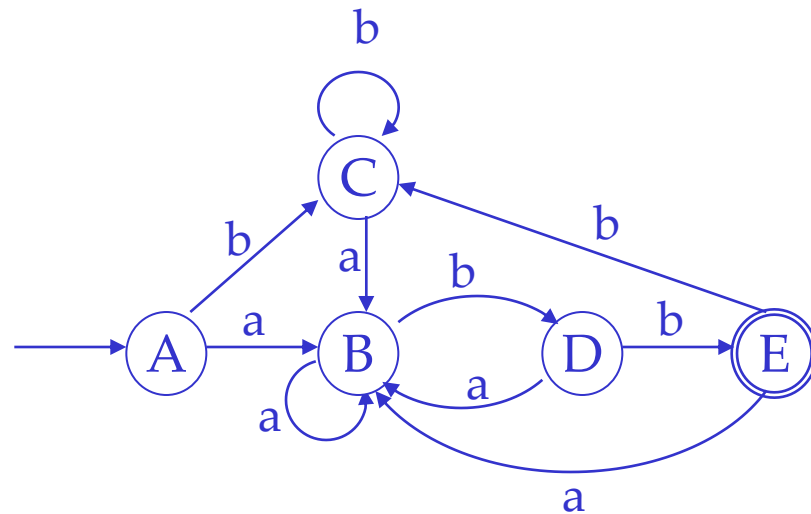
Ítem no básico	Ítems básicos
$\alpha.\lambda\gamma$	$\alpha\lambda.\gamma$
$\alpha.(\beta)\gamma$	$\alpha(. \beta)\gamma$
$\alpha.(\beta)^*\gamma$	$\alpha(. \beta)^*\gamma$ $\alpha(\beta)^*.\gamma$
$\alpha(\beta.)^*\gamma$	$\alpha(\beta)^*.\gamma$ $\alpha(\beta)^*.\gamma$
$\alpha.(\beta_1 \beta_2 \dots)\gamma$	$\alpha(. \beta_1 \beta_2 \dots)\gamma$ $\alpha(\beta_1 . \beta_2 \dots)\gamma$...
$\alpha(\dots \beta. \dots)\gamma$	$\alpha(\dots \beta \dots).\gamma$

2.4.5 Minimización de estados de un AFD

- Algoritmo:
 - (Paso1) Crear una partición con dos grupos: los estados finales y los no finales
 - (Paso 2) Para cada grupo con varios estados, dividir el grupo en subgrupos tales que dos estados, s y t , estén en el mismo subgrupo si y solo si para cada símbolo a , si existe la transición desde s con el símbolo a hacia un estado de un cierto grupo, entonces debe existir la transición desde t con el símbolo a hacia un estado del mismo grupo.
 - (Paso 3) Repetir el paso 2 hasta que no se dividan más grupos
 - (Paso 4) Cada grupo representa un estado en el AFD minimizado
 - (Paso 5) Eliminar los estados no alcanzables desde el estado inicial y los que no tengan transiciones que puedan conducir a un estado final

2.4.5 Minimización de estados de un AFD

- Ejemplo



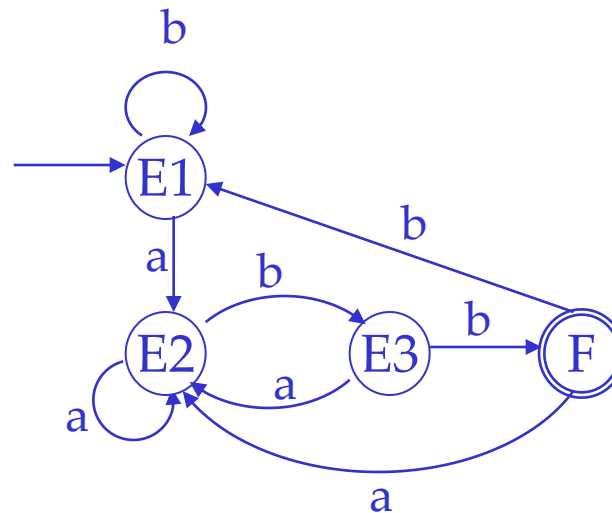
	a	b
A	B	C
B	B	D
C	B	C
D	B	E
E	B	C

		a	b
E1	A	E1	E1
	B	E1	E1
	C	E1	E1
	D	E1	F
F	E	E1	E1

		a	b
E1	A	E1	E1
	B	E1	E2
	C	E1	E1
E2	D	E1	F
F	E	E1	E1

2.4.5 Minimización de estados de un AFD

• Ejemplo



	a	b
A	B	C
B	B	D
C	B	C
D	B	E
E	B	C

		a	b
E1	A	E2	E1
	C	E2	E1
E2	B	E2	E3
E3	D	E2	F
F	E	E2	E1

- No existen estados no alcanzables
- No existen estados que no conduzcan a un estado final

2.4.6 Comparación AFD vs AFN

- Tienen la misma capacidad descriptiva
- Número de estados (memoria ocupada):
 - en un AFN crece linealmente con el tamaño de la expresión. Una expresión de tamaño r puede ser reconocida por un AFN de $2 \cdot r$ estados
 - en un AFD crece exponencialmente con el tamaño de la expresión. En el peor caso, una expresión de tamaño r puede necesitar un AFD con 2^r estados
- El tiempo de análisis:
 - en un AFD es de orden $O(n)$
 - en un AFN es de orden $O(n \cdot r)$
 - n (tamaño de la cadena), r (tamaño de la expresión)

2.1 Introducción

2.2 Especificación de categorías léxicas

2.3 Autómatas Finitos No Deterministas

2.4 Autómatas Finitos Deterministas

2.5 Implementación de un analizador léxico

2.5.1 Características

- Objetivo:
 - Dividir el flujo de entrada en componentes léxicos
- Características:
 - Cada categoría léxica tiene asociada su expresión regular y un conjunto de acciones (emitir u omitir, calcular un valor,...)
 - Estrategia avariciosa: intentar reconocer la cadena más larga posible antes de cambiar de categoría léxica.
 - Utilizaremos máquinas discriminadoras deterministas (MDD) para implementarlos

2.5.2 Máquina discriminadora determinista

- Funcionamiento:
 - Muy similar al autómata finito determinista
 - Tienen asociadas acciones a los estados finales
- Creación de una MDD
 - Paso 1: Añadir un símbolo especial a cada expresión regular
 - Paso 2: Unir todas las expresiones en una
 - Paso 3: Construir el AFD de la expresión
 - Paso 4: Eliminar los estados finales y los arcos de símbolos especiales
 - Paso 5: Convertir en estados finales los estados origen de los arcos de símbolos especiales
 - Paso 6: Asociar a estos estados las acciones correspondientes

2.5.2 Máquina discriminadora determinista

- Ejemplo (Paso 1):

categoría	Expresión regular
entero	$[0-9][0-9]^{\#}_{\text{entero}}$
real	$[0-9][0-9]^{\#} \backslash . [0-9][0-9]^{\#}_{\text{real}}$
identificador	$[a-zA-Z][a-zA-Z0-9]^{\#}_{\text{identificador}}$
asignación	$:=^{\#}_{\text{asignación}}$
rango	$\backslash . \backslash .^{\#}_{\text{rango}}$
blanco	$[\backslash t \backslash n][\backslash t \backslash n]^{\#}_{\text{blanco}}$
eof	$\text{eof}^{\#}_{\text{eof}}$

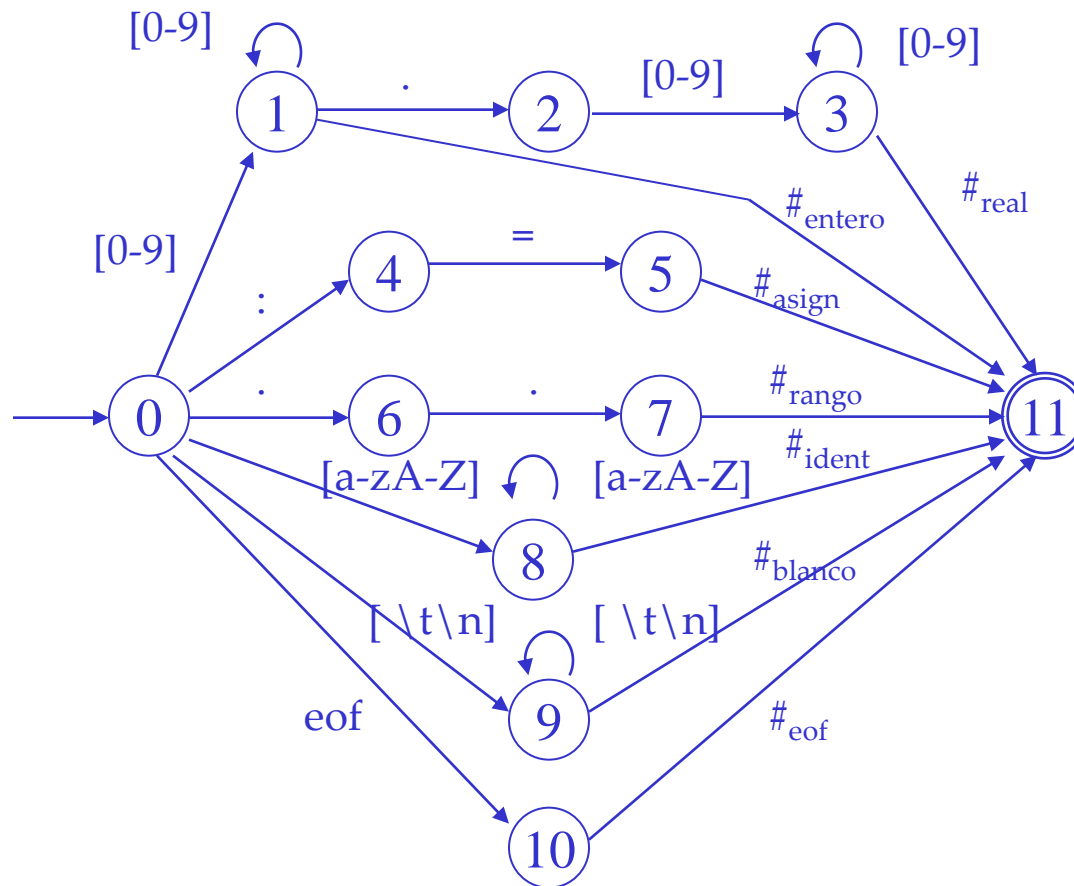
2.5.2 Máquina discriminadora determinista

- Ejemplo (Paso 2):

Expresión regular
$\begin{aligned} & [0-9][0-9]^* \#_{\text{entero}} \mid [0-9][0-9]^* \backslash . [0-9][0-9]^* \#_{\text{real}} \\ & \mid [a-zA-Z][a-zA-Z0-9]^* \#_{\text{identificador}} \mid := \#_{\text{asignación}} \\ & \mid \backslash . \backslash . \#_{\text{rango}} \mid [\backslash t \backslash n] [\backslash t \backslash n]^* \#_{\text{blanco}} \mid \text{eof} \#_{\text{eof}} \end{aligned}$

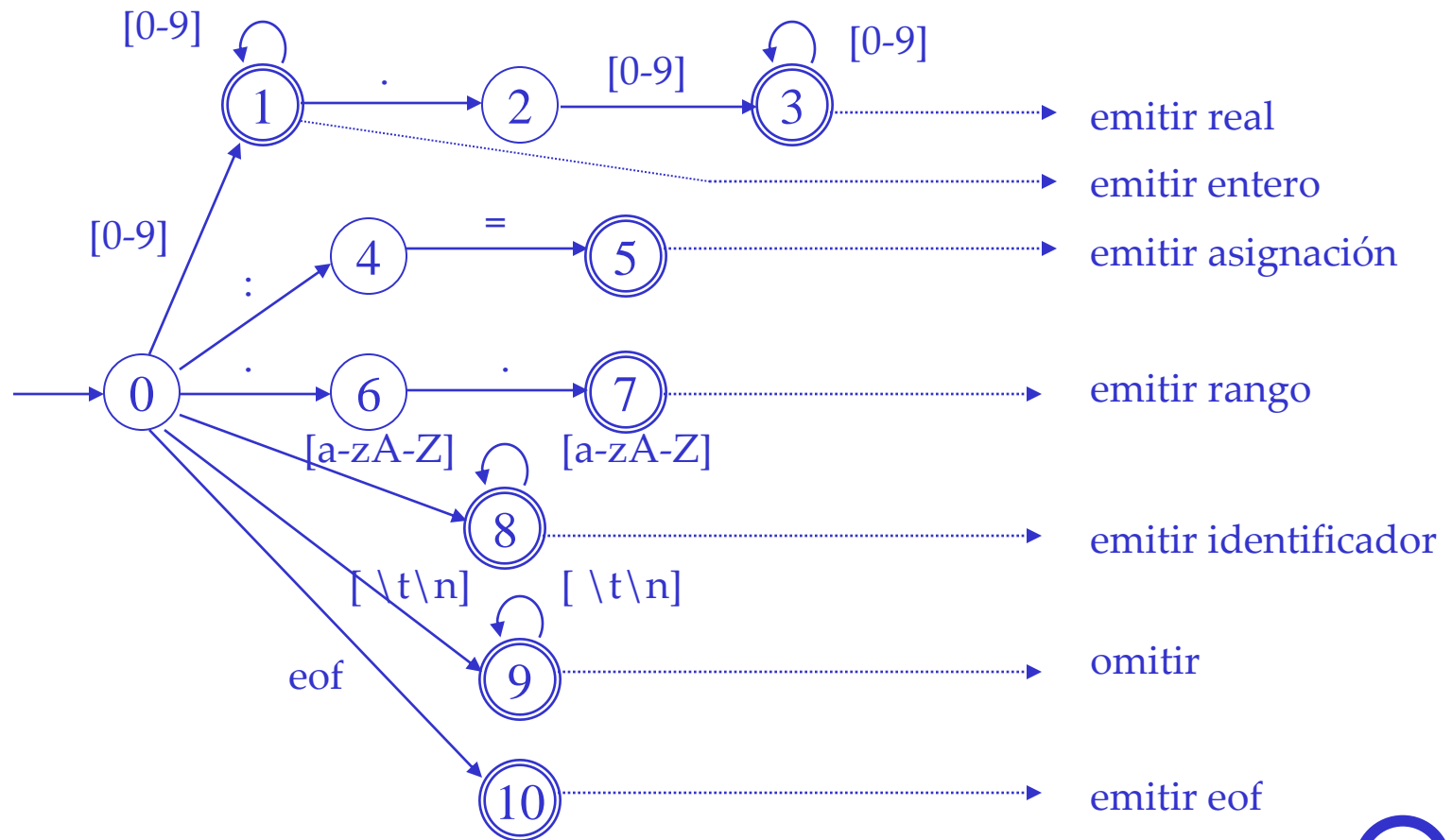
2.5.2 Máquina discriminadora determinista

- Ejemplo (Paso 3):



2.5.2 Máquina discriminadora determinista

- Ejemplo (Paso 4-6):



2.5.2 Máquina discriminadora determinista

- Funcionamiento:
 - Inicialmente la máquina se encuentra en el estado 0.
 - Evoluciona como un AFD hasta encontrar un carácter no reconocido (estrategia avariciosa).
 - Retrocede hasta el último estado final encontrado.
 - Ejecuta las acciones asociadas al estado final.
 - Vuelve al estado inicial.

2.5.3 Tratamiento de errores

- Detección:
 - Cuando no se ha alcanzado ningún estado final siguiendo el comportamiento anterior.
- Tratamiento:
 - Generar un error léxico.
 - Devolver los caracteres leídos.
 - Eliminar el primer carácter.
 - Continuar el análisis.

2.5.3 Tratamiento de errores

- Tratamiento alternativo:
 - Crear expresiones regulares que describan errores léxicos
 - Ejemplo:
 - ErrorNúmeroReal: `\.[0-9]+`
 - ErrorRango: `\.`
 - Permite dar mucha más información con el error

2.5.3 Tratamiento de errores

- Herramientas automáticas:
 - Suelen abortar al detectar errores léxicos
 - Solución:
 - Añadir categorías léxicas que detecten los errores
 - Emitir tokens de error léxico

2.5.3 Tratamiento de errores

- Errores en las acciones asociadas :
 - Suelen corresponder a lexemas con valores no permitidos (números fuera de rango, por ejemplo)
 - Solución no recomendada:

- Utilizar expresiones regulares que lo eviten

- Ejemplo: literales tipo byte (0-255) → No se debe hacer.

- $[0-9](b|B) \mid [0-9][0-9](b|B) \mid [01][0-9][0-9](b|B) \mid$
 $2[0-4][0-9](b|B) \mid 25[0-5](b|B)$

Es mejor
analizar el literal
y, a nivel semántico,
comprobar que esté
en el rango.

2.5.3 Tratamiento de errores

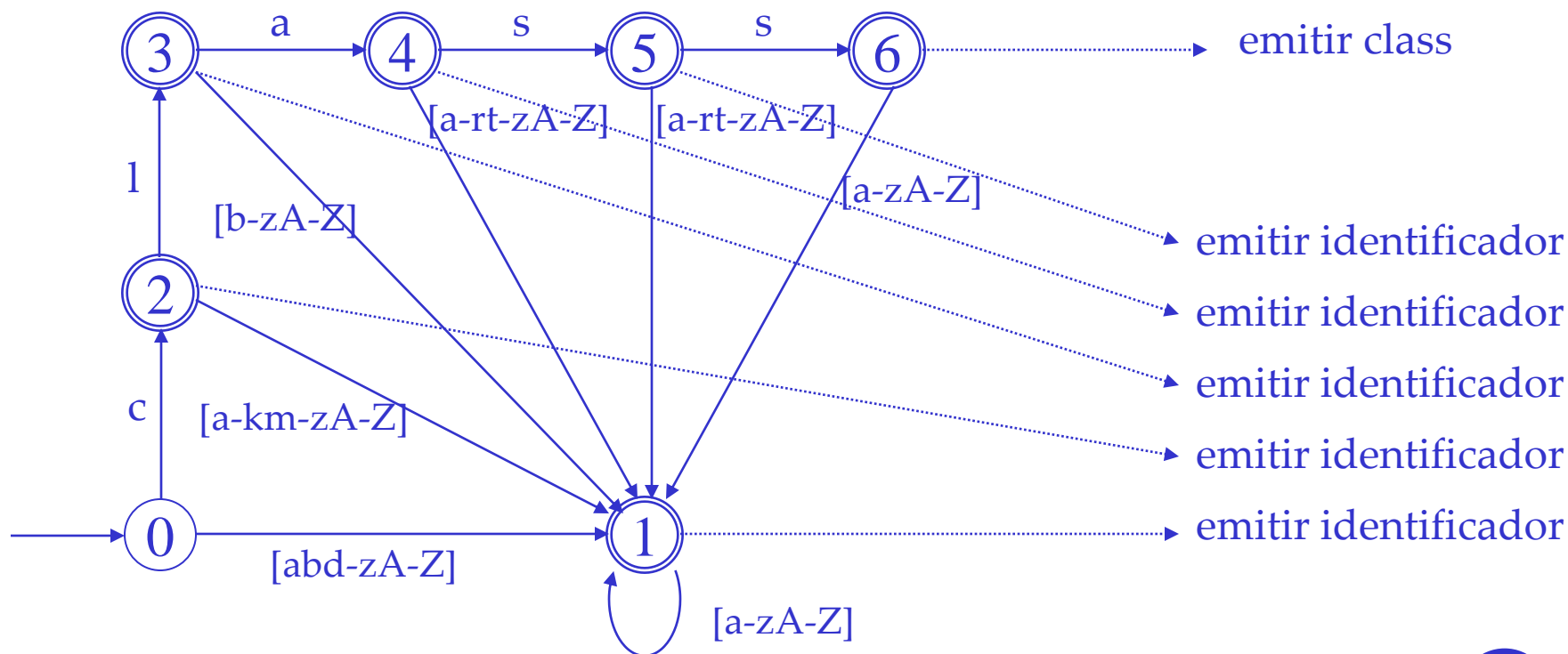
- Errores en las acciones asociadas :
 - Solución recomendada:
 - Detectarlos antes de emitir el token
 - Emitir tokens de error léxico
 - Tratamiento en las herramientas automáticas:
 - El análisis léxico sólo devuelve lexemas.
 - Los lexemas se evalúan a nivel sintáctico o semántico

2.5.4 Tratamiento de palabras reservadas

- Problema:
 - Se confunden con identificadores
- Solución (muy poco recomendable):
 - Modificar la expresión regular del identificador para evitarlo
 - Generar el AFD de la forma habitual

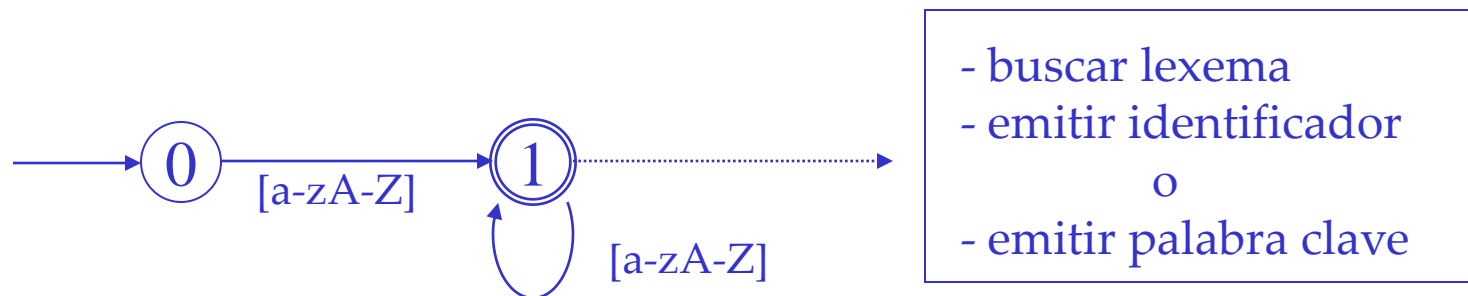
2.5.4 Tratamiento de palabras reservadas

- Ejemplo: (identificador y class)

NO RECOMENDABLE

2.5.4 Tratamiento de palabras reservadas

- Solución:
 - Tratar las palabras reservadas como identificadores
 - Almacenar todas las palabras reservadas en una tabla
 - Antes de emitir el identificador, buscar en la tabla
 - Si aparece en la tabla, emitir el token de la palabra reservada
 - Si no aparece en la tabla, emitir el token de identificador



2.5.4 Tratamiento de palabras reservadas

- Solución alternativa:
 - Utilizar Máquinas Discriminadoras No Deterministas

2.5.5 Máquina discriminadora no determinista

- Funcionamiento:
 - Similar a las MDD
 - Basadas en Autómatas Finitos No Deterministas
- Creación:
 - Paso 1: Ordenar las diferentes expresiones regulares
 - Paso 2: Unir todas las expresiones regulares
 - Paso 3: Generar el Autómata Finito No Determinista
 - Paso 4: Añadir las acciones a los estados finales

2.5.5 Máquina discriminadora no determinista

- Ejemplo (Paso 1):

categoría	Expresión regular
class	class
int	int
long	long
double	double
identificador	[a-zA-Z][a-zA-Z0-9]*
blanco	[\t\n][\t\n]*
eof	eof

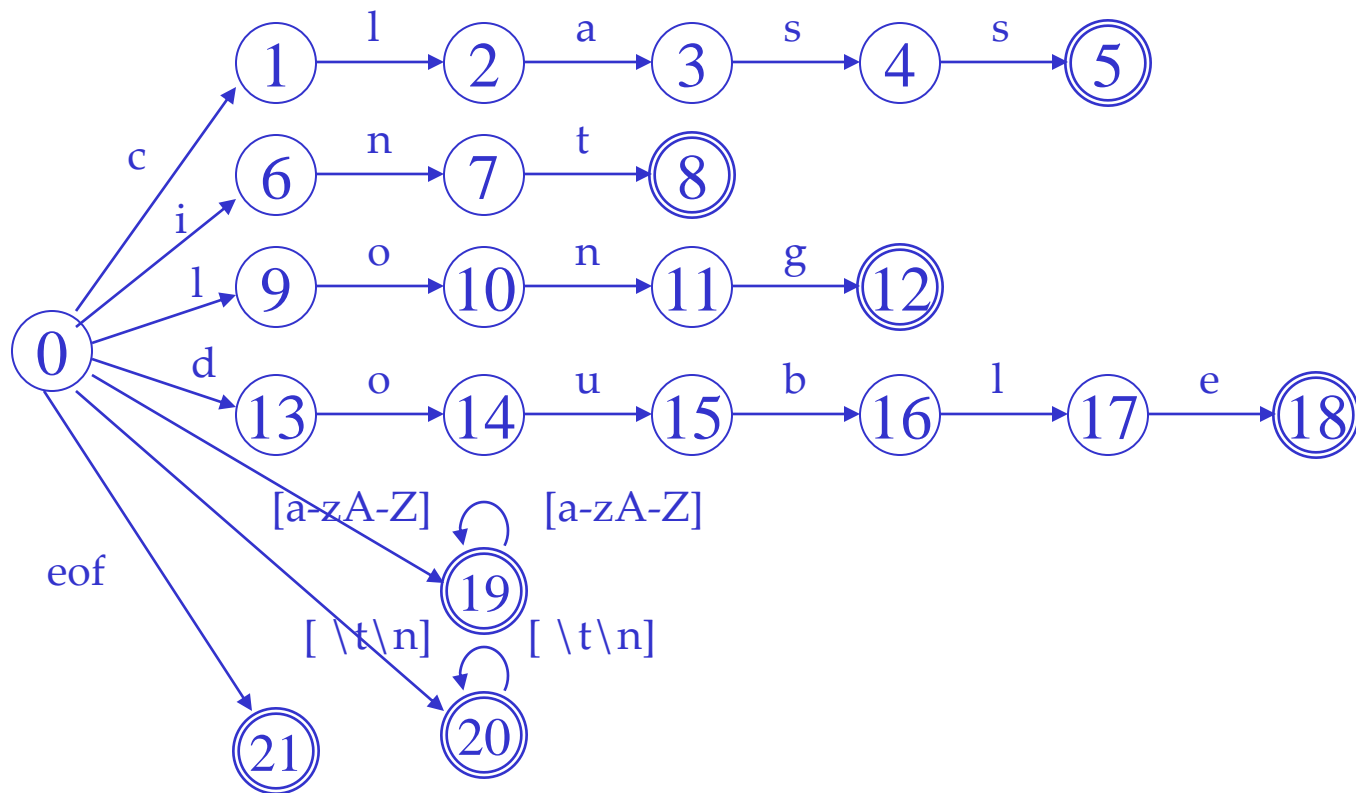
2.5.5 Máquina discriminadora no determinista

- Ejemplo (Paso 2):

Expresión regular
<code>class int long double [a-zA-Z][a-zA-Z0-9]* [\t\n][\t\n]* eof</code>

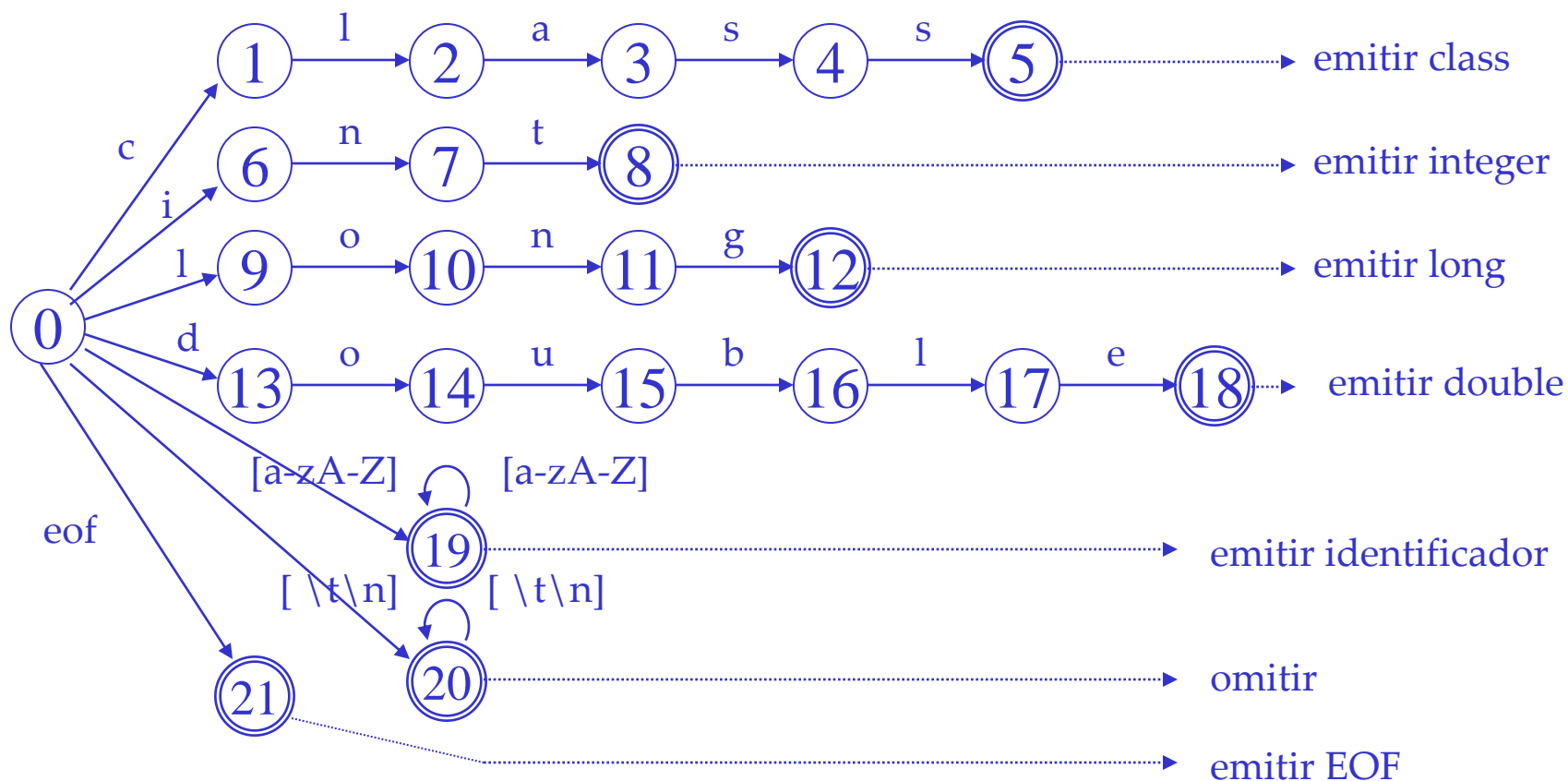
2.5.5 Máquina discriminadora no determinista

- Ejemplo (Paso 3):



2.5.5 Máquina discriminadora no determinista

- Ejemplo (Paso 4):



2.5.5 Máquina discriminadora no determinista

- Funcionamiento:
 - Inicialmente la máquina se encuentra en el estado 0.
 - Evoluciona como un AFN hasta encontrar un carácter no reconocido (estrategia avariciosa).
 - Retrocede hasta la última transición que contuviera al menos un estado final.
 - Si encuentra varios estados finales, selecciona el primero en el orden establecido en el paso 1.
 - Ejecuta las acciones asociadas al estado final seleccionado.
 - Vuelve al estado inicial.

2.5.5 Máquina discriminadora no determinista

- La mayoría de herramientas automáticas se basan en máquinas discriminadoras no deterministas
- Utilizan el orden de declaración de las expresiones regulares para resolver los conflictos entre estados finales
- Las palabras reservadas se declaran antes del token identificador (lo que resuelve el conflicto entre palabras reservadas e identificadores)
- Se suele introducir una última expresión regular que acepta cualquier carácter y que identifica al token de error léxico (de esta manera se evita que se aborte al encontrar un error léxico)
- Sólo utiliza las acciones emitir token y omitir (skip).

2.5.5 Máquina discriminadora no determinista

- Ejemplo: JavaCC

```
SKIP :  
{  
    " "  
|   "\r"  
|   "\n"  
|   "\t"  
|   <SINGLE_LINE_COMMENT: "//" (~["\n", "\r"])* (" \n" | " \r" | " \r\n")>  
|   <MULTI_LINE_COMMENT: "/*" (~["*"])* "*" (  
        "*" | (~["*", "/"] (~["*"])* "*" ) ) * "/">  
}
```

2.5.5 Máquina discriminadora no determinista

- Ejemplo: JavaCC

```
TOKEN : /* palabras clave */
{
  <CLASS: "class">
| <INTEGER: "int">
| <LONG: "long">
| <DOUBLE: "double">
}

TOKEN: /* identificador */
{
  <ID : ["a"-"z","A"-"Z","_"] (["a"-"z","A"-"Z","0"-"9","_"])* >
}
```