AUTÓMATAS

Regex convertido en robot

24 de agosto de 2022

Víctor Orozco

Universidad Rafael Landivar

ANÁLISIS LÉXICO

1. Especificación léxica
2. Autómata finito
3. Expresiones hacia autómatas
4. NFA hacia DFA
5. Implementación de un autómata finito

ESPECIFICACIÓN LÉXICA

1. Al menos uno: A+
2. Unión: A | B
3. Opción: A?
4. Rango: ‘a’+’b’+…+’z’
5. Rango (exclusión): Complemento de $[a-z]$

1. AA*
2. A+B
3. A+ε
4. $[a-z]$
5. $\hat{}[a-z]$

ESPECIFICACIÓN

¿Como saber si?

$$s \in L(R)$$

ESPECIFICACIÓN LÉXICA

1. Escribir una regex para los lexemas de cada clase (categoría) de token
   - ☐ Número = digit+
   - ☐ Palabra clave = ‘if’ + ‘else’ + …
   - ☐ Identificador= letter (letter + digit)*
   - ☐ OpenPar = ‘(‘

ESPECIFICACIÓN LÉXICA

2. Construir una *R* que coincida con todos los lexemas de todos los tokens

R = Palabra clave + Identificador + Número + ...
= R1+R2+...

ESPECIFICACIÓN LÉXICA

3. Considerando la entrada $x_1 x_n$
Para $1 \leqslant i \leqslant n$ verificar
$x_1 x_n \in L(R)$
4. Si éxito, entonces sabemos que:
$x_1 x_n \in L(R_j)$ para algun j
5. Eliminar $x_1 ... X_i$ de la entrada e ir a (3)

ESPECIFICACIÓN LÉXICA

¿Es utilizada toda la entrada?

ESPECIFICACIÓN LÉXICA

¿Cual token se debe utilizar?

ESPECIFICACIÓN LÉXICA

¿Y si la cadena no coincide?

RESUMEN

- Las expresiones regulares son una notación concisa para patrones de cadenas
- El uso en análisis léxico requiere pequeñas extensiones
  - Para resolver ambigüedades
  - Para manejar errores
- Buenos algoritmos conocidos
  - Requerir solo una pasada sobre la entrada
  - Pocas operaciones por carácter (búsqueda de tabla)

ANÁLISIS LÉXICO

1. Especificación léxica
2. Autómata finito
3. Expresiones hacia autómatas
4. NFA hacia DFA
5. Implementación de un autómata finito

AUTÓMATA FINITO

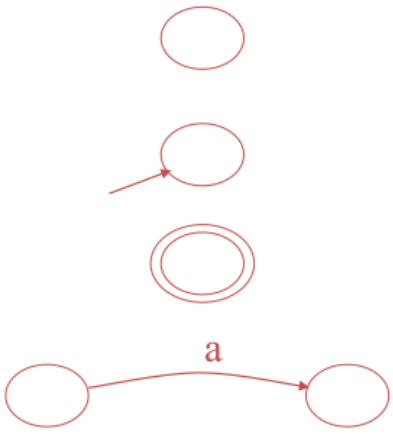
- Expresiones regulares = especificación
- Autómatas finitos = implementación
- Un autómata finito consta de
  - Un alfabeto de entrada Σ
  - Un conjunto de estados S
  - Un estado de inicio n
  - Un conjunto de estados de aceptación (metas) $F \subseteq S$
  - Un conjunto de transiciones estado $estado \rightarrow^{entrada} estado$

AUTÓMATA FINITO

- Transición
  $s_1 \rightarrow^a s_2$
- Se interpreta como:
  "En el estado s_1 con la entrada a dirigirse hacia s_2
- Si finaliza la entrada y está en estado de aceptación => aceptar
- De lo contrario => rechazar

AUTÓMATA FINITO

- Un estado
- Estado inicial
- Estado de aceptación (meta)
- Una transición



AUTÓMATA FINITO

Un autómata finito que solo acepta “1“

AUTÓMATA FINITO

Un autómata finito que acepta cualquier cantidad de ”1”seguido de un 0 simple.
Alfabeto: 0,1

AUTÓMATA FINITO

Seleccione el lenguaje regular que denota el mismo lenguaje que este autómata finito:

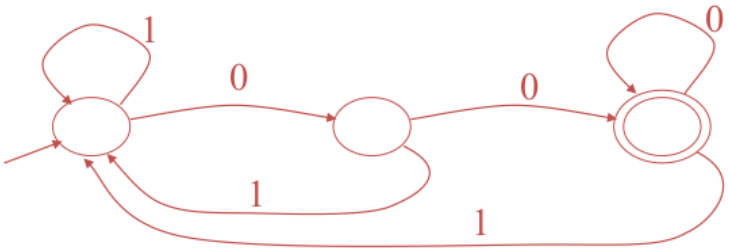


Figura: Automata Finito

AUTÓMATA FINITO

- $(0 + 1)*$
- $(1 * +0)(1 + 0)$
- $1 * +(01) * +(001) * +(000 * 1)*$
- $(0 + 1) * 00$

AUTÓMATA FINITO - EPSILON

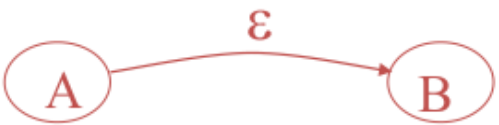


Figura: Autómata epsilon

AUTÓMATAS FINITOS

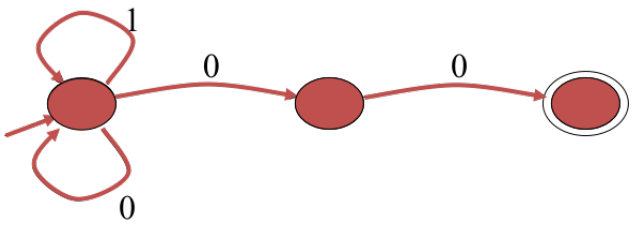
- Autómatas finitos deterministas (DFA)
  - Una transición por entrada por estado
  - Sin movimientos ε
- Autómatas finitos no deterministas (NFA)
  - Puede tener múltiples transiciones para una entrada en un estado dado
  - Puede tener movimientos ε

AUTÓMATAS FINITOS

- Un DFA toma solo un camino a través del gráfico de estado
- Un NFA puede elegir

AUTÓMATAS FINITOS

Un NFA puede desplazarse hacia varios estados



- Input: 1 0 0

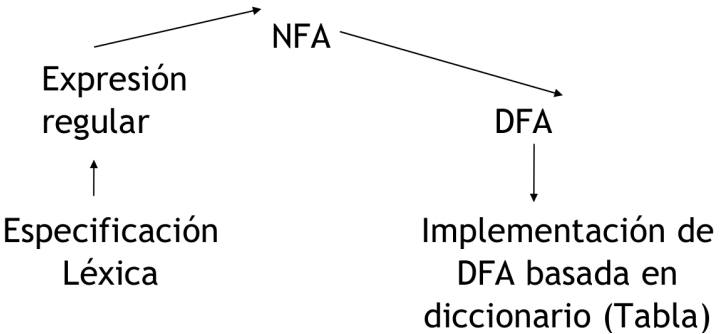
RESUMEN

- NFA y DFA reconocen el mismo conjunto de idiomas
- idiomas regulares
- Los DFA son más rápidos de ejecutar
- En los DFA para cualquier estado no hay opciones a considerar, es decir, es determinista
- Las NFA son, en general, más pequeñas

ANÁLISIS LÉXICO

1. Especificación léxica
2. Autómata finito
3. Expresiones hacia autómatas
4. NFA hacia DFA
5. Implementación de un autómata finito

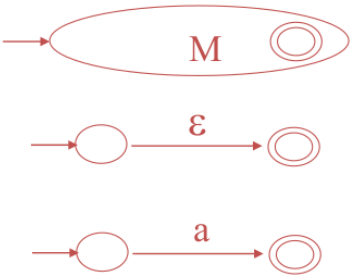
EXPRESIONES HACIA AUTÓMATAS



EXPRESIONES HACIA AUTÓMATAS

Para cada regex, definir un NFA

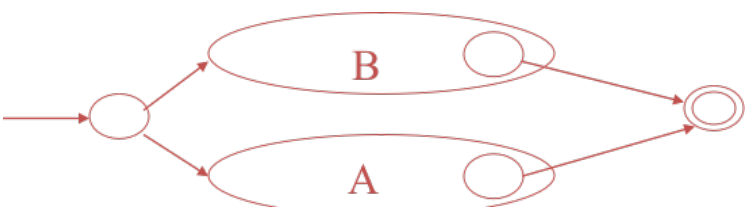
- Notación: NFA para regex M
- Para ε
- Para la entrada a



EXPRESIONES HACIA AUTÓMATAS

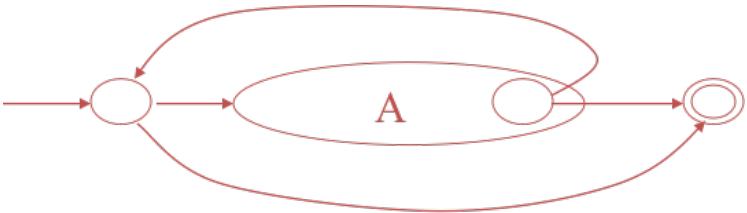
Para AB

Para A+B



EXPRESIONES HACIA AUTÓMATAS

Para A*

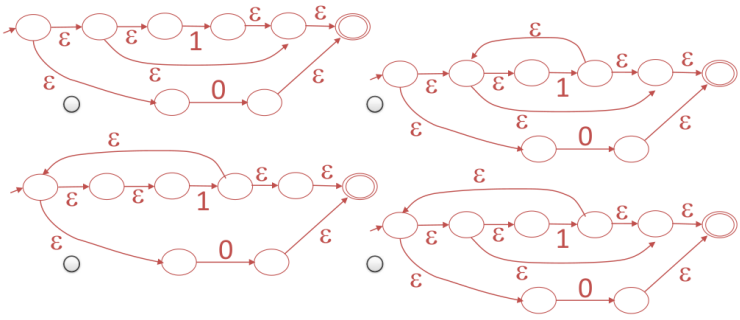


EXPRESIONES HACIA AUTÓMATAS

Considere la expresión regular (1+0)*1

EXPRESIONES HACIA AUTÓMATAS

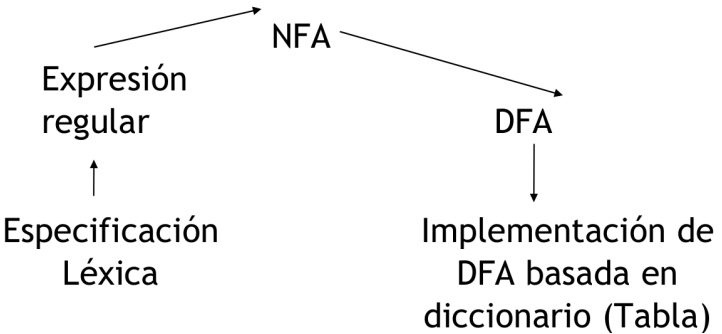
NFA de ¿1*+0?



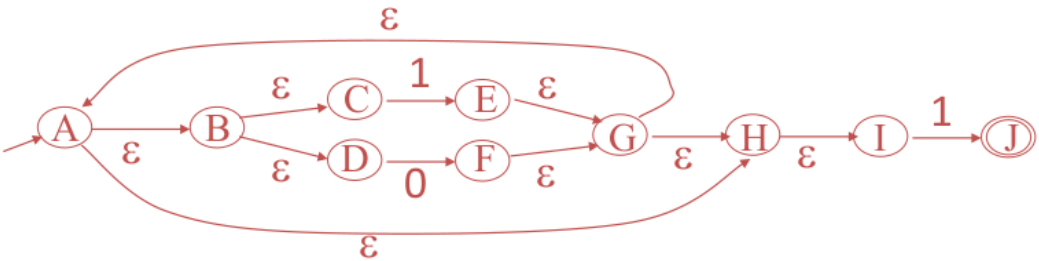
ANÁLISIS LÉXICO

1. Especificación léxica
2. Autómata finito
3. Expresiones hacia autómatas
4. NFA hacia DFA
5. Implementación de un autómata finito

NFA HACIA DFA



NFA HACIA DFA

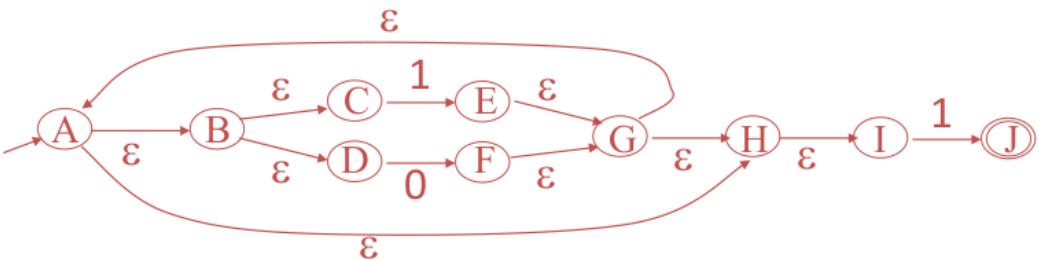


NFA HACIA DFA

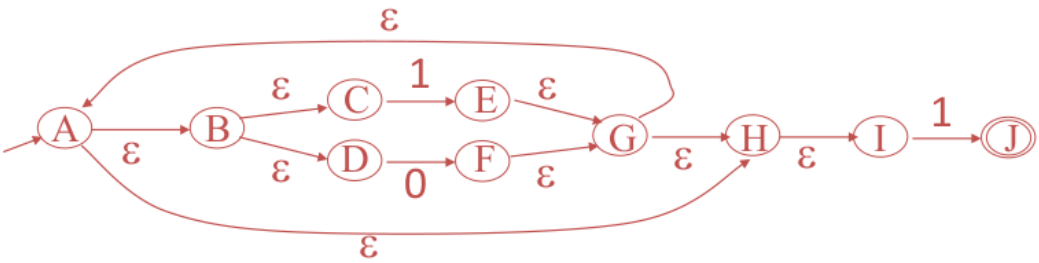
- Un NFA puede estar en varios estados en cualquier momento
- ¿Cuantos?

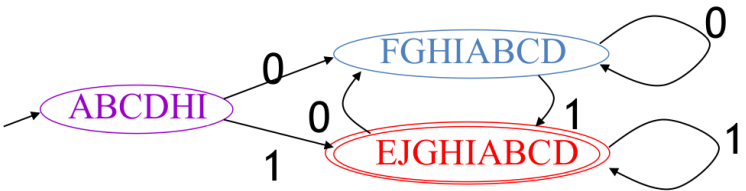
NFA HACIA DFA

NFA HACIA DFA

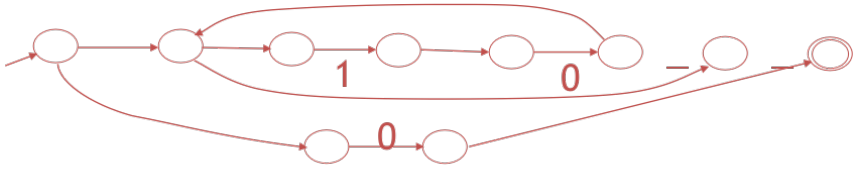


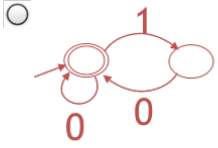
NFA HACIA DFA

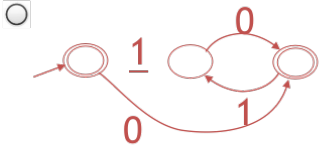


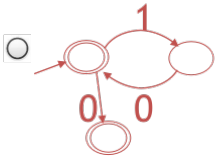


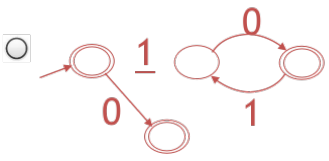
NFA HACIA DFA







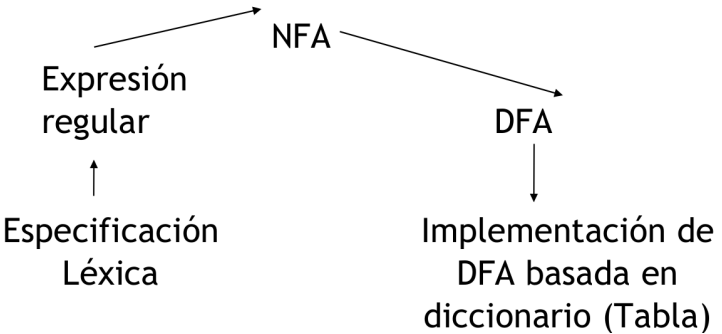




ANÁLISIS LÉXICO

1. Especificación léxica
2. Autómata finito
3. Expresiones hacia autómatas
4. NFA hacia DFA
5. Implementación de un autómata finito

IMPLEMENTACIÓN DE UN AUTÓMATA FINITO

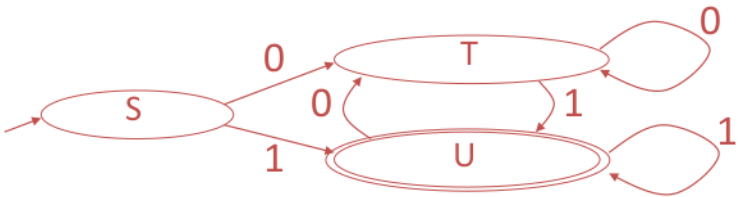


IMPLEMENTACIÓN DE UN AUTÓMATA FINITO

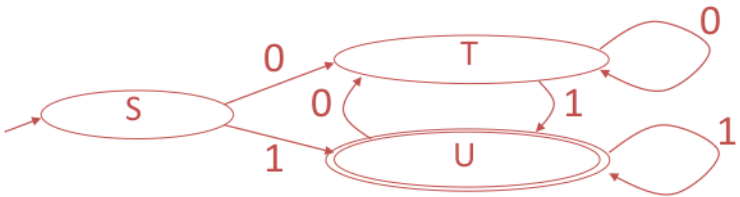
Un DFA puede ser implementado por una tabla T de 2D

- Una dimensión son los estados
- Otra dimensión es el símbolo de entrada
- Para cada transición $S_i \rightarrow^a S_k$ define $T[i, a] = k$

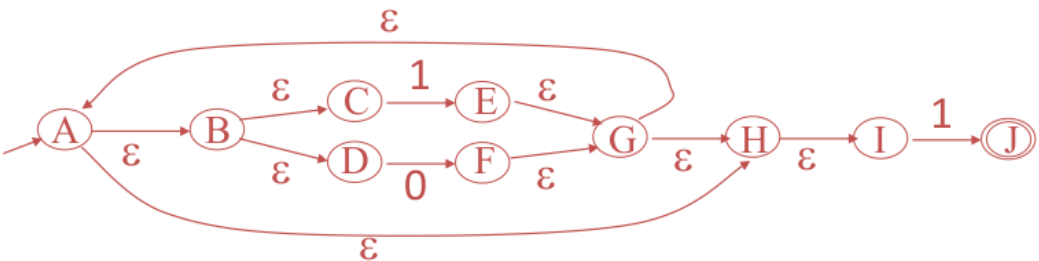
IMPLEMENTACIÓN DE UN AUTÓMATA FINITO



IMPLEMENTACIÓN DE UN AUTÓMATA FINITO



IMPLEMENTACIÓN DE UN AUTÓMATA FINITO



IMPLEMENTACIÓN DE UN AUTÓMATA FINITO

- NFA -> DFA: Conversión clave
- Velocidad vs. espacio