IC-5701
Compiladores e Intérpretes

Profesor:
Ing. Allan Rodríguez Dávila, MGP

Agenda

TEC

- Gramáticas Libres de Contexto
- Autómatas Finitos
- Autómatas Finitos No Deterministas

Gramáticas Libres de Contexto

Gramáticas

```
programa -> bloque

bloque -> { decls instr }

….

instr -> if (expr) instr else instr
```

Gramáticas libres de contexto

- Pueden ser reconocidos por autómatas de pila
- Generan lenguajes libres de contexto
- Permiten bloques anidados en lenguajes de programación

Backus-Naur Form

- Notación popular para escribir gramáticas libres de contexto
  - Común para definir sintaxis de lenguajes de programación
- Las reglas tienen la siguiente forma:
  - `<simbolo> ::= <expresión con símbolos>`

Gramáticas libres de contexto

- Consisten en:
  - Terminales (Σ)
    - Símbolos básicos
  - No Terminales (N)
    - Variables sintácticas
  - Símbolo Inicial (S)
    - Sus producciones "inician" la gramática
  - Producciones ($P \subseteq N \times (N \cup \Sigma)^*$)
    - Especifican cómo pueden relacionarse terminales y no terminales

Estructura Producciones

- Un no terminal como encabezado.
- Símbolo → o ::=
- Cero o más terminales y no terminales como cuerpo

```
ImportDeclaration ::= import [static] Identifier { . Identifier } [.*] ;
```

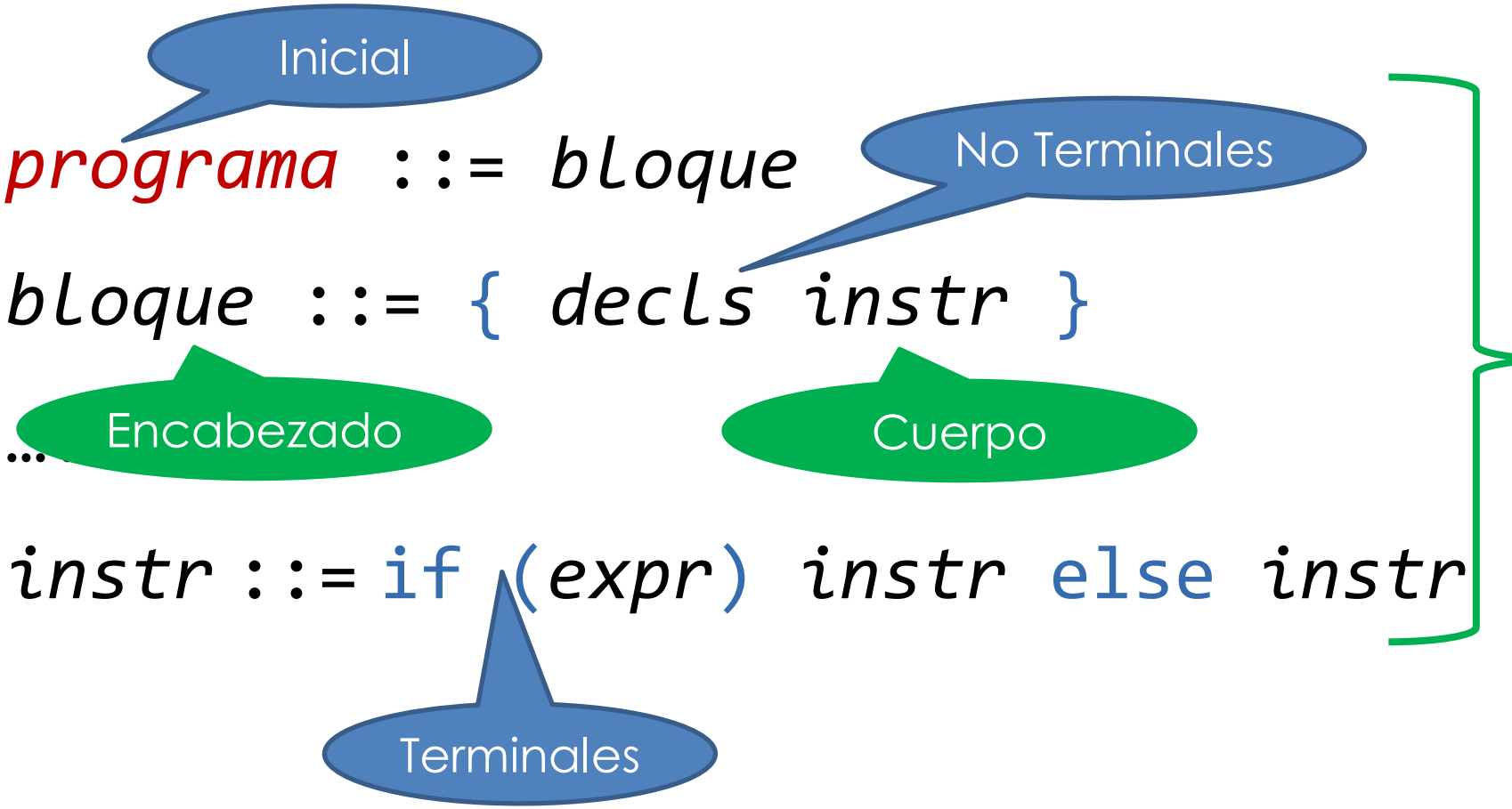
Notación Terminales

- Los símbolos de operadores: +, -, *, etc
- Los símbolos de puntuación: ,, ;, (,), etc
- Los dígitos 0, 1, …, 9
- Las cadenas no cursivas (en digital con negrita): **do**, **while**, etc

Notación No Terminales

- Nombres en cursiva y minúscula: *expr*, *instr*, etc
- En casos de estudio se puede utilizar letras mayúscula y la letra S como símbolo inicial.

Gramáticas libres de contexto



Derivaciones

- Se construye un árbol de derivación, o de análisis sintáctico, derivado de la reescritura de las producciones.
- En la reescritura se sustituye a un no terminal por el cuerpo de una de sus producciones.

Derivaciones

expresion ::= *expresion* + *expresion*

expresion ::= *expresion* - *expresion*

expresion ::= *expresion* * *expresion*

expresion ::= – *expresion*

expresion ::= (*expresion*)

expresion ::= **id**

- Derivación de –(**id**):

expresion ⇒ – *expresion* ⇒ – (*expresion*) ⇒ –(**id**)

Derivaciones

expresion ::= *expresion* + *expresion*

expresion ::= *expresion* - *expresion*

expresion ::= *expresion* * *expresion*

expresion ::= – *expresion*

expresion ::= (*expresion*)

expresion ::= **id**

- Derivación de **id**+(**id***id**):

expresion ⇒ *expresion* + *expresion*

⇒ **id** + *expresion*

⇒ **id** + (*expresion*)

⇒ **id** + (*expresion* * *expresion*)

⇒ **id** + (**id** * **id**)

Ejemplos de Gramáticas

- Palíndromo binario

palindromo → ε

palindromo → 0

palindromo → 1

palindromo → 0*palindromo*0

palindromo → 1*palindromo*1

Ejemplos de Gramáticas

- Secuencia de paréntesis balanceados

cadenaPar → *cadenaPar parentestis*
cadenaPar → *parentestis*
parentestis → (*cadenaPar*)
parentestis →)*cadenaPar*(
parentestis → ()
parentestis →)(

Ejemplos de Gramáticas

- Asignación aritmética tipo Pascal

sentenciaAsignacion → *var* := *expresion*

expresion → *expresion* + *termino*

expresion → *expresion* – *termino*

expresion → *termino*

termino → *termino* * *factor*

termino → *termino* / *factor*

termino → *factor*

factor → (*expresion*)

factor → *var*

factor → *num*

var → [a-zA-Z][a-zA-Z0-9_]*

num → 0|[1-9][0-9]*

Ejemplos de Gramáticas

- Expresiones aritméticas en postfijo

```
digito ::= [0-9]
numero ::= {digito}+
expresion ::= expresion expresion +
expresion ::= expresion expresion -
expresion ::= expresion expresion *
expresion ::= expresion -
expresion ::= (expresion)
expresion ::= numero
```

Portafolio #2: Gramáticas

Gramática que permita un único main y cero o muchas funciones (main y funciones con una única expresión de asignación o creación)

Investigación

- Gramáticas Libres de contexto:
  - Definición
  - Notación
  - Derivaciones
  - Gramática Pascal
  - Gramática C
  - Gramática Java
    - https://docs.oracle.com/javase/specs/jls/se7/html/index.html

Autómatas Finitos

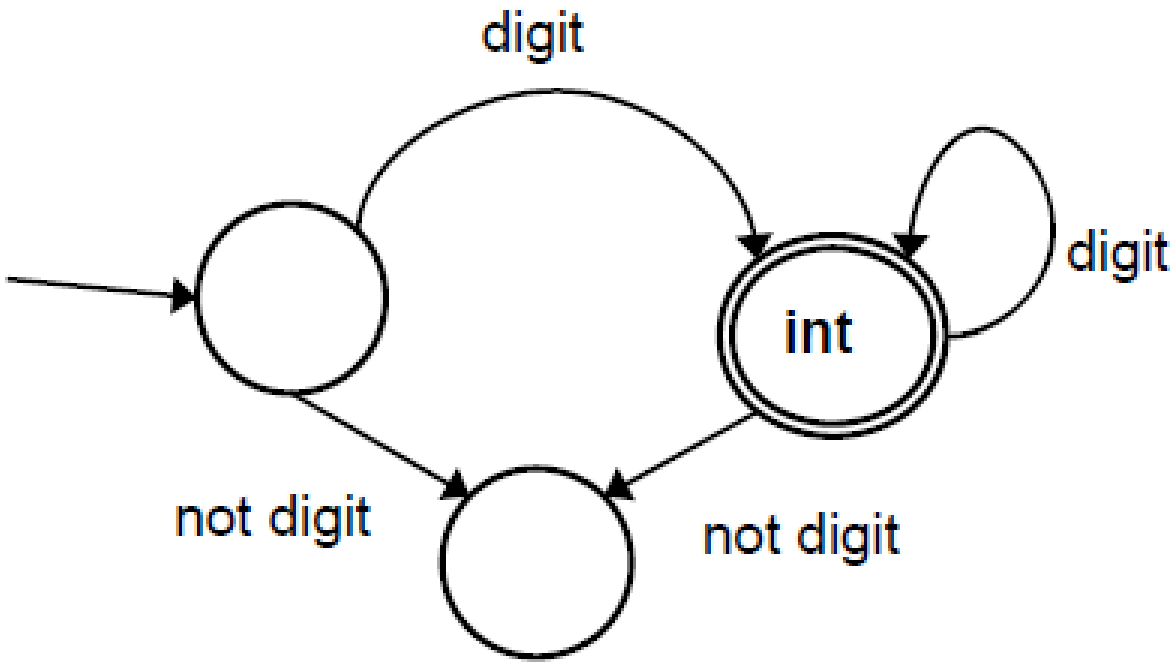
Autómatas Finitos

- Grafos de transición de estados.
- Son reconocedores.
- Son capaces de reconocer lenguajes regulares.
  - Reconocen tokens

Autómatas Finitos

- Todo lenguaje definido por un autómata finito es también definido por una expresión regular
- Todo lenguaje definido por una expresión regular es también definido por un autómata finito
- Son utilizados para construir analizadores léxicos

Autómatas Finitos



Clasificación

- Autómatas Finitos No Deterministas
  - No tienen restricciones en etiquetas
- Autómatas Finitos Deterministas
  - Para cada estado tienen una única arista con el símbolo que sale del estado.

Autómatas Finitos No Deterministas

Autómatas Finitos no Determinísticos

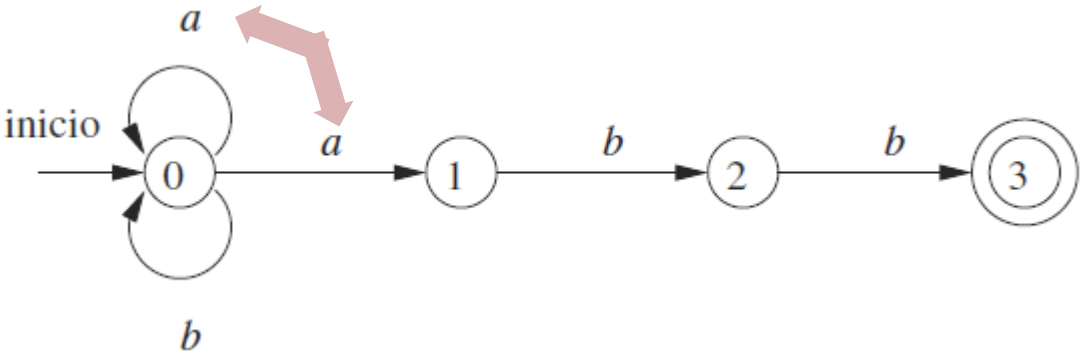
- Posee un conjunto finito de estados ***S***.
- Un conjunto de símbolos de entrada **Σ**
  - Alfabeto de entrada
  - **ε** no pertenece a **Σ**
- La función de transición proporciona un conjunto de estados siguientes para **Σ U {ε}**

Autómatas Finitos no Determinísticos

- S_0 se conoce como estado inicial.
- Posee un conjunto de estados ***F***, subconjunto de ***S***, aceptantes (estados finales).
- Se representan cómo grafos de transición de estados.

Autómatas Finitos no Determinísticos

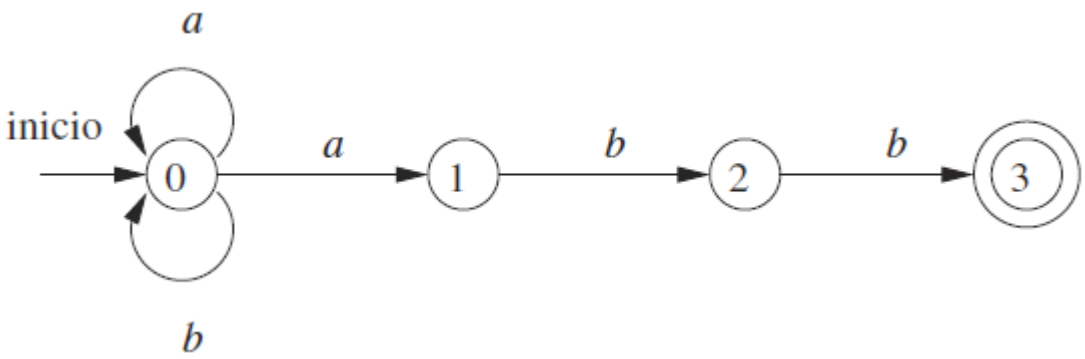
- Grafo de transición para un AFN que reconoce el lenguaje de la expresión regular `(a|b)*abb`.



- El mismo símbolo puede etiquetar aristas de un estado a diferentes estados.

Autómatas Finitos no Determinísticos

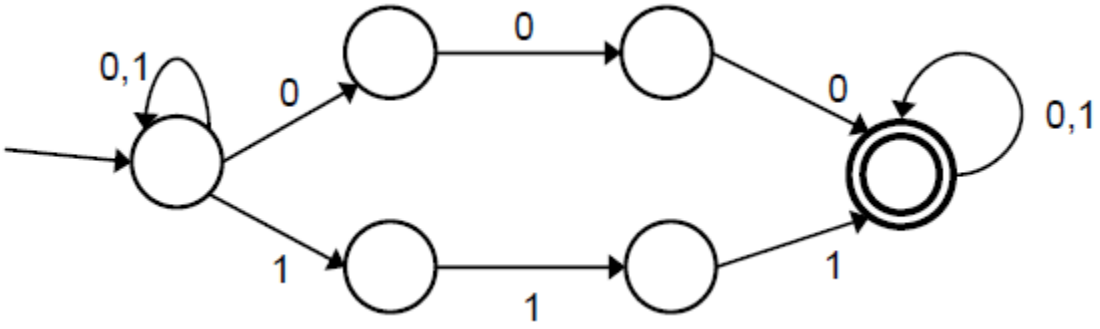
- Grafo de transición para un AFN que reconoce el lenguaje de la expresión regular `(a|b)*abb`.



- Una arista puede etiquetarse por **ε** y por los símbolos del alfabeto de entrada.

Autómatas Finitos no Determinísticos

- Grafo de transición para un AFN que reconoce el lenguaje de la expresión regular `(0|1)*(000|111)(0|1)*`.



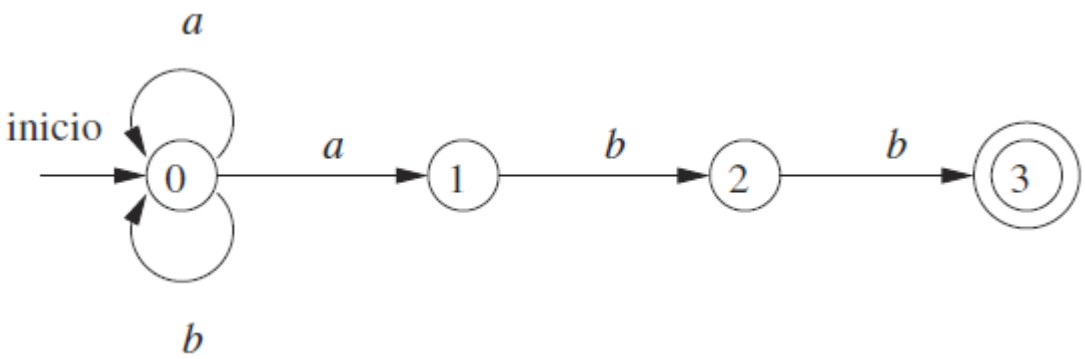
Tablas de Transición

- Sirven como variante para representar AFN.
- Filas son los estados.
- Columnas son los símbolos de entrada y **ε.**

Tablas de Transición

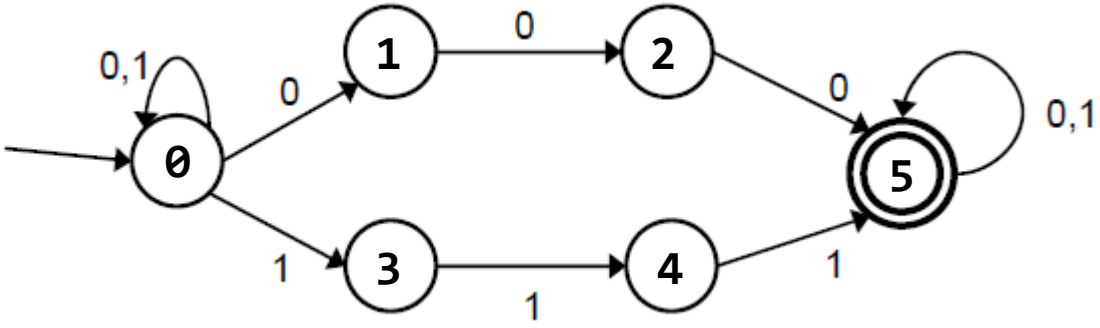
- Los valores de las celdas corresponden al valor de la función de transición.
- Si la función de transición no tiene información o no aplica para los valores de estado-entrada se coloca ∅.

Tablas de Transición



Estado	a	b	ε
0	{0,1}	{0}	∅
1	∅	{2}	∅
2	∅	{3}	∅
3	∅	∅	∅

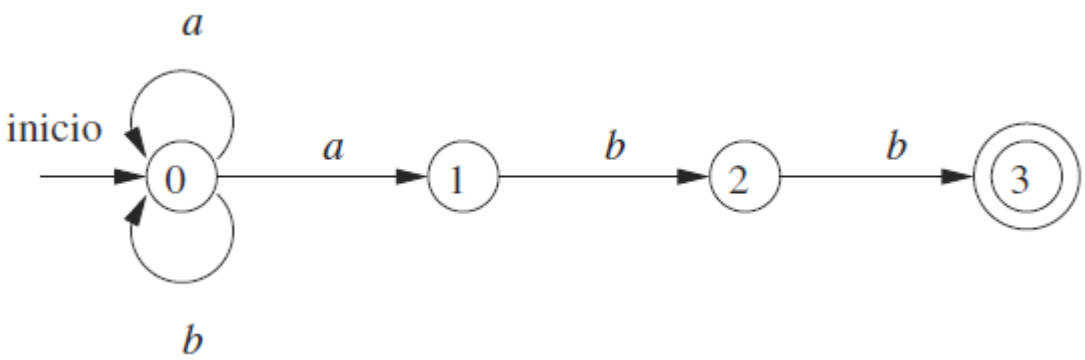
Tablas de Transición



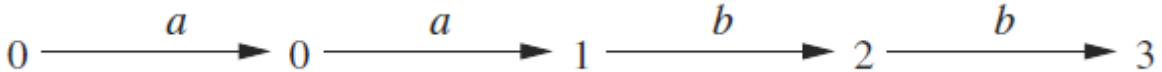
Estado	0	1	ε
0	{0,1}	{0,3}	∅
1	{2}	∅	∅
2	{5}	∅	∅
3	∅	{4}	∅
4	∅	{5}	∅
5	{5}	{5}	∅

Caminos

- Dado:



- Exprese el camino para la entrada:
  - aabb

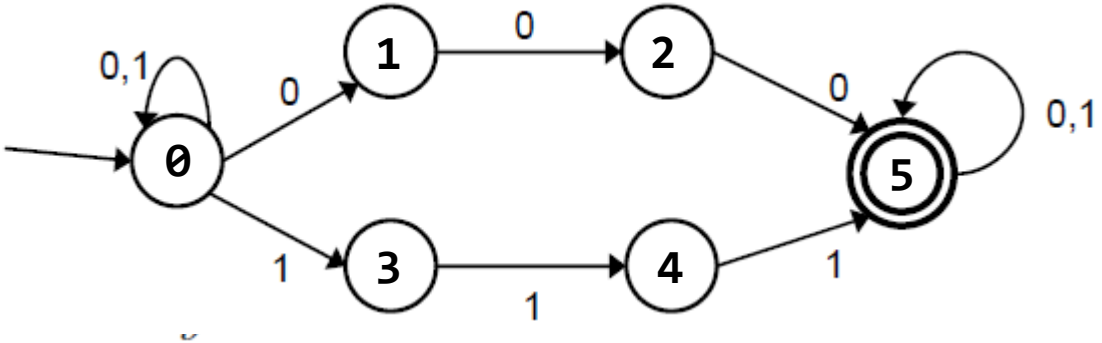


  - abbbaaabb

TEC

Caminos

- Dado:



- Exprese el camino para la entrada:
  - 111
  - 111000

Ejemplo AFN

- AFN del siguiente lenguaje denotado por la siguiente expresión regular:

  a(a | b)*(a | b)(a | b)+

TEC

Portafolio #2

- Desarrolle los AFN de los lenguajes denotados por las siguientes expresiones regulares y su tabla de transición:

  a(a|b)a(a|b)

  abc|a(b|d)d

  $11^{+}0$

  (011+)+

Bibliografía

- Programming Language Processors in Java: compilers and interpreters. Watt, David, Brown, Deryck. Pearson Education. 2000

- Compilers: principles, techniques and tools (2da. ed.). Aho, Alfred. Pearson Education. 2007


TEC
Tecnológico
de Costa Rica