



# SINTAXIS Y SEMÁNTICA DEL LENGUAJE.

**Tema: Gramáticas Formales**

# LENGUAJES DE PROGRAMACIÓN

Un lenguaje de programación (LP) es una notación formal para describir algoritmos ejecutables por una computadora. Posee dos componentes:

## ❖ *Sintaxis*

Conjunto de reglas formales que especifican la forma correcta de escribir sus sentencias. Se expresan a través de reglas gramaticales y/o diagramas sintácticos.

## ❖ *Semántica*

Especifica el significado de las sentencias de un programa sintácticamente válido escrito en dicho lenguaje.



# SINTAXIS $\leftrightarrow$ SEMÁNTICA

- Program ejemplo;

var

x, i: integer;

begin

readln(i);

if (i > 2) then

    x:= 4\*i;

.....

.....

end.

Cada instrucción tiene un significado particular para el S.O.

## Ejemplo:

La declaración de una variable produce la reserva de espacio en memoria para almacenar el dato correspondiente al tipo de la variable declarada.



# ¿CÓMO DESCRIBIR LA SINTAXIS DE UN LP?

- Todo lenguaje  $L$  (natural o artificial) está formado por un conjunto de sentencias.
- Las sentencias están construidas con caracteres que pertenecen a un alfabeto  $\Sigma$  dado.
- Por lo tanto, la sintaxis de un lenguaje se define por dos conjuntos de reglas:
  - Reglas léxicas
  - Reglas sintácticas



# REGLAS LÉXICAS

- Especifican el conjunto de caracteres que conforman el alfabeto sobre el cual se define el lenguaje L.
- Los elementos léxicos básicos se denominan lexemas, son predefinidos y no necesitan ser explicados. Son siempre válidos.
- Abarcan a los literales numéricos, alfabéticos, operadores, símbolos especiales y palabras claves o reservadas: begin, end, var, +, and,....



# REGLAS LÉXICAS

- Cada grupo de lexemas se agrupa en categorías y se lo identifica con un símbolo.

**Ejemplo:** los números son lexemas,  
<dig> es la categoría

<dig> = 0 | 1 | 2 | ..... | 9 → esto significa que un dígito  
puede ser un 0 o un 1 o  
un 2 o.....un 9



# REGLAS SINTÁCTICAS

- Determinan el modo en que se combinan los caracteres del alfabeto para formar una sentencia válida.
- Cada lenguaje tiene sus propias reglas.

## Ejemplo:

En Pascal no hay distinción entre mayúsculas y minúsculas, pero en C sí.

En Pascal el símbolo de la desigualdad es  $\neq$ , pero en C y en Python el símbolo es  $\neq$



# ¿CÓMO DESCRIBIR LA SINTAXIS DE UN LP?

- La sintaxis se expresa por medio de una notación formal universal: BNF (Backus - Naur Form).
- BNF es un metalenguaje y utiliza *gramáticas* para establecer las reglas de escritura de un lenguaje dado.





# GRAMÁTICA

- Es la descripción formalizada de las frases de un lenguaje y está basada en reglas gramaticales. (Chomsky)
- Una gramática provee un conjunto de reglas o patrones que permiten reconocer y/o generar las sentencias válidas de un lenguaje. Dichas reglas se denominan *reglas gramaticales o reglas de producción*. (RP)



# REGLA GRAMATICAL O DE PRODUCCIÓN

- Una *regla gramatical* es una expresión de la forma:

$A \rightarrow B$ , y se lee A se reemplaza por B o bien,  
A deriva en B

Tanto A como B son cadenas de símbolos en las cuales pueden aparecer:

- Elementos del alfabeto, llamados *terminales (minúscula)*
- Elementos nuevos variables, llamados *no terminales (mayúscula)*



# REGLA GRAMATICAL O DE PRODUCCIÓN

- La aplicación de una regla  $A \rightarrow B$   
sobre una palabra  $xAy$   
produce una palabra derivada  $xBy$ .
- Esto se denomina *paso de derivación*



# DEFINICIÓN FORMAL DE GRAMÁTICA

Formalmente, una gramática  $G$  es una cuádrupla:  
 $G = \{RP, T, NT, S\}$ , donde:

$RP$ - Un conjunto de reglas gramaticales o *reglas de producción o reglas de derivación*.

$T$ - Un conjunto  $T$  de *símbolos terminales*, que son los caracteres del alfabeto.

$NT$ - Un conjunto  $NT$  de *símbolos no terminales*, que deben ser definidos a través de una regla de producción.

$S$ - *símbolo inicial* a partir del cual se derivan todas las palabras válidas del lenguaje.



# CLASIFICACIÓN DE LAS GRAMÁTICAS

Del mismo modo que hemos visto la clasificación de los lenguajes según Chomsky, se aplica el mismo criterio para las gramáticas dado que cada tipo de lenguaje es generado y verificado por un tipo determinado de gramática:

Gramáticas tipo 3 o regulares

Gramáticas tipo 2 o libres de contexto

Gramáticas tipo 1 o sensibles al contexto

Gramáticas tipo 0 (sin restricciones para los lenguajes recursivamente enumerables)



# CLASIFICACIÓN DE LAS GRAMÁTICAS

Nosotros veremos solo las gramáticas regulares (asociadas a los lenguajes regulares) y las gramáticas libres de contexto (asociados a los lenguajes libres de contexto), ya que los lenguajes de programación se ubican dentro de estas dos clasificaciones (la mayoría de los lenguajes de programación son lenguajes libres de contexto).



# GRAMÁTICAS REGULARES

- Son las más restringidas. Generan los lenguajes regulares. Reciben este nombre porque presentan ‘regularidades’ o repeticiones en la conformación de sus palabras.
- $L1 = \{ab, abab, ababab, \dots\}$  contiene repeticiones de ab
- $L2 = \{\varepsilon, abc, cc, aabb, abccc, \dots\}$   $a^n b^n c^m$   $n \geq 0$   $m \geq 0$
- Si L es un lenguaje finito también se lo considera regular.

**Ejemplo:**

$L = \{anita, lava, la, tina\}$



# GRAMÁTICAS REGULARES

## Características de una gramática regular:

- El lado derecho de la RP debe contener un símbolo terminal, y a lo sumo, un símbolo no terminal.

Los símbolos **terminales** se escriben en minúscula;  
los **no terminales** en mayúscula

$A \rightarrow a$	$\langle A \rangle ::= a$	en BNF
$A \rightarrow aA$	$\langle A \rangle ::= a \langle A \rangle$	en BNF

- Se puede incluir la producción  $S \rightarrow \varepsilon$  si el lenguaje debe incluir la cadena vacía.
- Lineal a derecha:  $A \rightarrow aB$  o  $A \rightarrow a$  'B aparece a derecha de a'
- Lineal a izquierda:  $A \rightarrow Ba$  o  $A \rightarrow a$





# GRAMÁTICAS REGULARES

**Ejemplo 1:** Sea  $L = \{w / w = ab^n, n \geq 1\}$   $\Sigma = \{a, b\}$

Cuando  $n=1$   $w_1=ab$

Para  $n=2$   $w_2=abb$

Para  $n=3$   $w_3=abbb$

Es decir que las palabras empiezan con una sola 'a' seguida de una o más letras 'b'.

- $S \rightarrow aB$  la regla S debe decir que la palabra empieza con 'a', seguidas de letras 'b'
- $B \rightarrow b$  ahora la regla para B debe permitirme tener 1 sola 'b'
- $B \rightarrow bB$  o varias 'b', entonces uso reglas recursivas

# GRAMÁTICAS REGULARES

Para hacer la definición formal de la gramática, tenemos en cuenta la cuádrupla  $GR = \{ S, NT, T, RP \}$ , entonces:

$$RP = \{$$
$$S \rightarrow aB$$
$$B \rightarrow b$$
$$B \rightarrow bB \}$$
$$NT = \{B\}$$
$$T = \{a, b\}$$


# GRAMÁTICAS REGULARES

**Ejemplo 2:** Sea  $L = \{ w \mid w \text{ termina en } 1 \}$  sobre  $\Sigma = \{0,1\}$   
La palabra más corta es  $w_1=1$ ;  $w_2=01$ ;  $w_3=11$ ;  
 $w_4=001$ ;  $w_5=101$ ; .....

- $S \rightarrow 1$  la primer regla representa la palabra más corta
- $S \rightarrow A1$  luego la regla inicial debe decir que las palabras terminan en '1'
- Ahora hay que pensar que puede venir antes del '1' final: un solo '0', varios '0', un solo '1' o varios '1', o varios '0' y '1' mezclados.
- $A \rightarrow 0$  permite que venga un solo '0' antes del '1' final.
- $A \rightarrow A0$  permite que vengan varios '0' seguidos antes del '1' final.
- $A \rightarrow 1$  permite que venga un solo 1 antes del '1' final.
- $A \rightarrow A1$  permite que vengan varios 1 seguidos antes del '1' final.



# GRAMÁTICAS REGULARES

**Ejemplo 3:** Sea  $L = \{ w / w = 0^{2m}, m \geq 1 \}$  sobre  $\Sigma = \{0\}$   
Para  $m=1$   $w_1=00$ ;  $m=2$   $w_2=0000$ ;  $m=3$   $w_3=000000$ , .....


a)  $S \rightarrow 0A$  la regla inicial  $S$  debe permitirme escribir palabras con varios 0 seguidos

b)  $A \rightarrow 0$  esta regla me asegura obtener  $w_1$ , la palabra más corta  
Ahora debo pensar cómo escribir la regla siguiente para asegurarme que las palabras tengan un número par de letras '0'.

La regla recursiva  $A \rightarrow 0A$  no me sirve porque no controla que sea cantidad par.

c)  $A \rightarrow 0S$  hago una regla con recursión indirecta.

Partiendo de la regla a), desde  $S$  ya tengo un 0 seguido de  $A$ ; si aplico la regla c), y reemplazo la  $A$ , tengo 00 seguido de  $S$ , y usando la regla a) de nuevo para reemplazar  $S$  tengo 000 $A$ . Si finalmente para reemplazar la  $A$  aplico la regla b), obtengo 0000 que es la palabra  $w_2$ .



# GRAMÁTICAS REGULARES

**Ejemplo 4:** Sea  $L = \{ w / w = bc^n, n \geq 0 \}$  sobre  $\Sigma = \{b, c\}$   
Para  $n=0$   $w_1=b$ ;  $n=1$   $w_2=bc$ ;  $n=2$   $w_3=bcc$ ; .....  
Las palabras comienzan con una sola 'b', seguidas de  
cero o más letras 'c'.

La regla inicial S debe reflejar la palabra más corta  
 $w_1=b$

$S \rightarrow b$

$S \rightarrow bC$       la regla inicial debe permitirme generar el  
resto de las palabras que tienen una o  
varias letras 'c', después de la primer 'b'

$C \rightarrow c$       una sola 'c'

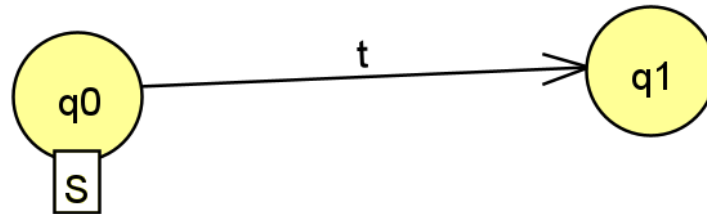
$C \rightarrow cC$       esta regla debe permitirme escribir varias  
'c' seguidas



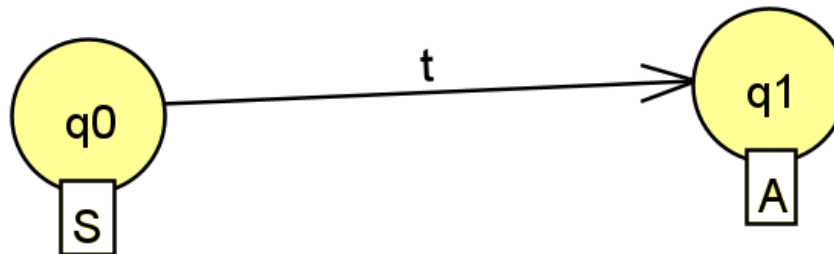
# CONVERSIÓN AF A GRAMÁTICA REGULAR

- Pasos:

- 1- Asociar al estado inicial  $q_0$  el símbolo  $S$

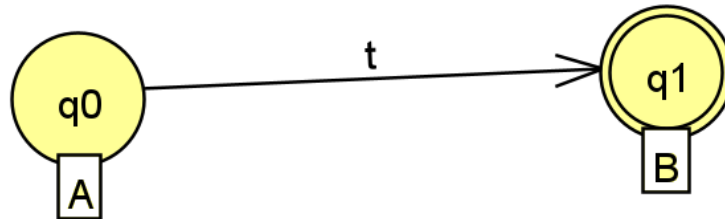


- 2- Asociar a cada estado  $q_i$  restante del autómata un símbolo no terminal



# CONVERSIÓN AF A GRAMÁTICA REGULAR

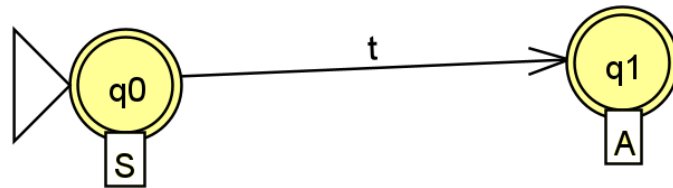
- 3- Para cada transición 't' que lleva del estado A al estado B, agregar a las reglas de producción, la regla  $A \rightarrow tB$  siendo A y B los símbolos no terminales.



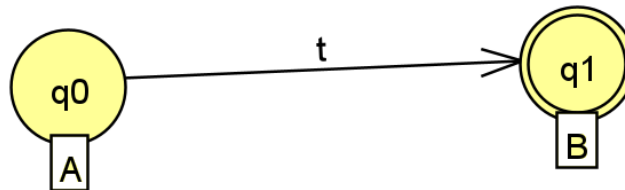
- 4- Si B es un estado final, agregar la regla de producción:  $B \rightarrow \varepsilon$

# CONVERSIÓN AF A GRAMÁTICA REGULAR

- 5- Si el estado inicial también fuese final, agregar la regla :  $S \rightarrow \varepsilon$



- 6- Si el estado B es final y hay una transición 't' que lleva del estado A al B, se agrega la regla  $A \rightarrow t$





# CONVERSIÓN AF A GR

## Ejemplo 1:

RP={

$S \rightarrow aA$

$S \rightarrow bS$

$A \rightarrow aB$

$A \rightarrow bS$

$B \rightarrow aC$

$B \rightarrow bS$

$C \rightarrow aC$

$C \rightarrow bC$

$S \rightarrow \varepsilon$  /\* por ser S, A y B finales \*/

$A \rightarrow \varepsilon$

$B \rightarrow \varepsilon$

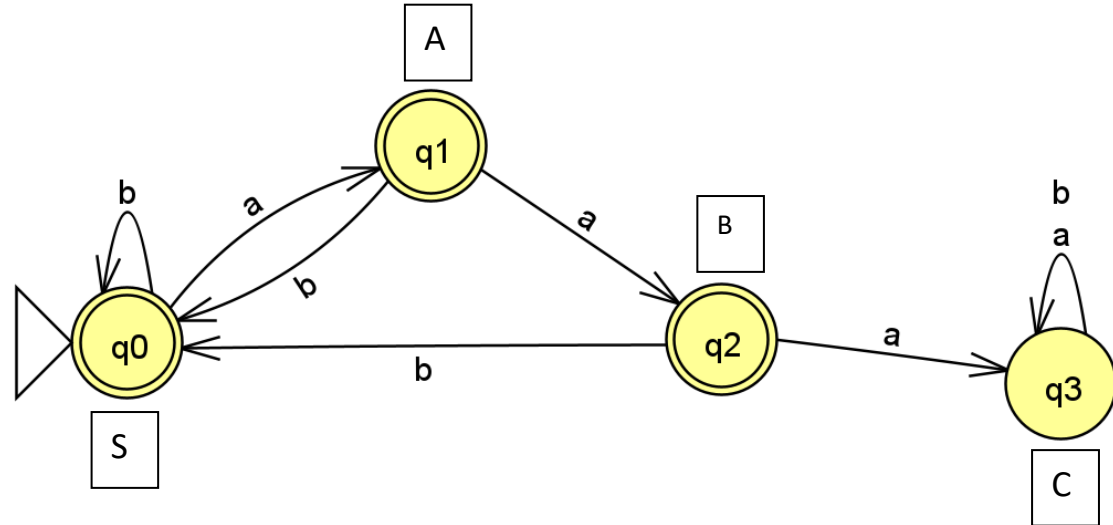
$A \rightarrow a$  /\*por llegar con 'a' al estado final B \*/

$A \rightarrow b$  /\* por llegar con 'b' al estado final S\*/

$S \rightarrow b$  /\* por llegar con 'b' al estado final S (rulo)\*/

$S \rightarrow a$  /\* por llegar con 'a' al estado final A \*/

$B \rightarrow b$  } /\* por llegar con 'b' al estado final S\*/



$\Sigma=\{a,b\}$   $G=\{ NT=\{A,B,C\}, T=\{a,b\}, RP, S\}$  No van los casos bases si en el AF no esta la flecha correspondiente y el nodo no es final.

# CONVERSIÓN AF A GR

## ○ Ejemplo 2:

RP={

$S \rightarrow 0A$

$S \rightarrow 1B$

$A \rightarrow 0C$

$A \rightarrow 1B$

$B \rightarrow 1C$

$B \rightarrow 0A$

$C \rightarrow 0C$

$C \rightarrow 1C$

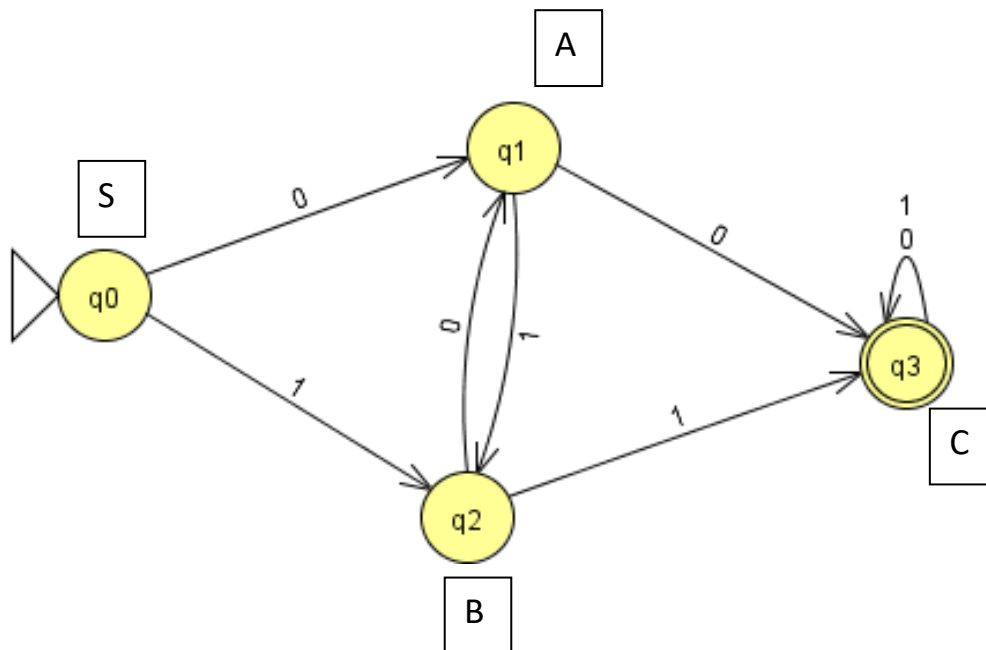
$C \rightarrow 0$

$C \rightarrow 1$

$A \rightarrow 0$

$B \rightarrow 1$

$C \rightarrow \varepsilon$  }



$L = \{w / w \text{ contiene la subcadena } 00 \text{ o la subcadena } 11\}$

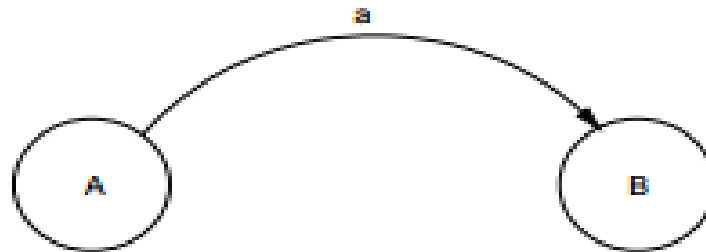
$\Sigma = \{0, 1\}$      $G = \{ \text{ NT} = \{A, B, C\}, \text{ T} = \{0, 1\}, \text{ RP}, S \}$

# CONVERSIÓN DE GR A AF

- 1- A cada símbolo **no terminal** se le asocia 1 estado (nodo) del autómata.

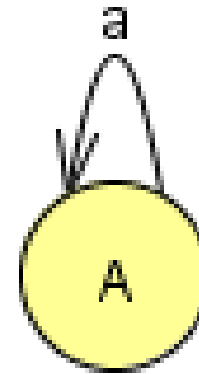
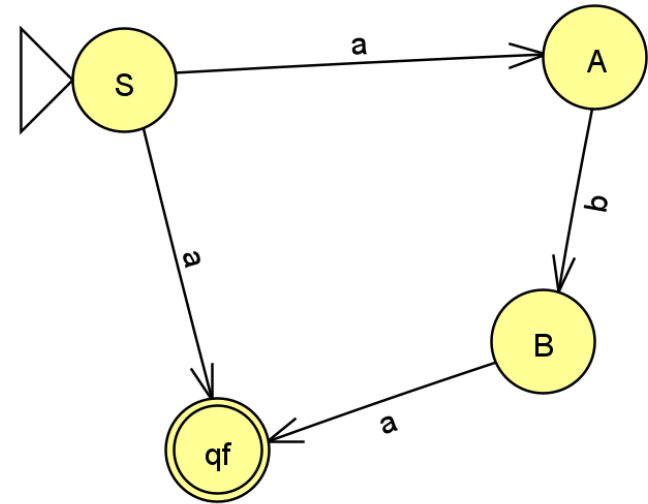
S es el símbolo inicial que será el nodo inicial del AF

- 2- Si hay una regla de la forma:  $A \rightarrow aB$  se la grafica:



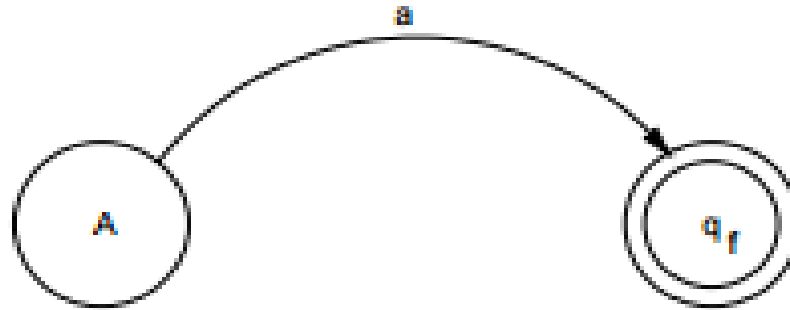
# CONVERSIÓN DE GR A AF

- 3- Se introduce el estado qf como único estado final
- 4- Si hay una regla de la forma:  
 $A \rightarrow aA$  se la grafica:



# CONVERSIÓN GR A AF

- Si hay una regla de la forma:  $A \rightarrow a$  se la grafica



Donde  $q_f$  es estado final y **no corresponde a ningún no terminal de la gramática.**

**El autómata que se obtiene puede ser un  
AFD o un AFN  
(en general son AFN)**



# CONVERSIÓN GR A AF

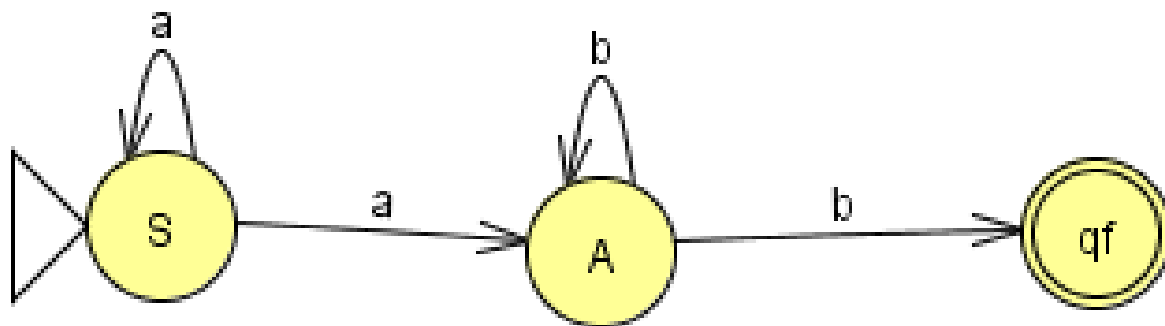
## ○ Ejemplo 1:

$S \rightarrow aS$

$S \rightarrow aA$

$A \rightarrow bA$

$A \rightarrow b$



$\Sigma = \{a, b\}$

Observar que el AF obtenido es no determinístico.



# CONVERSIÓN GR A AF

## ○ Ejemplo 2:

$A \rightarrow aA$

$A \rightarrow bB$

$B \rightarrow aA$

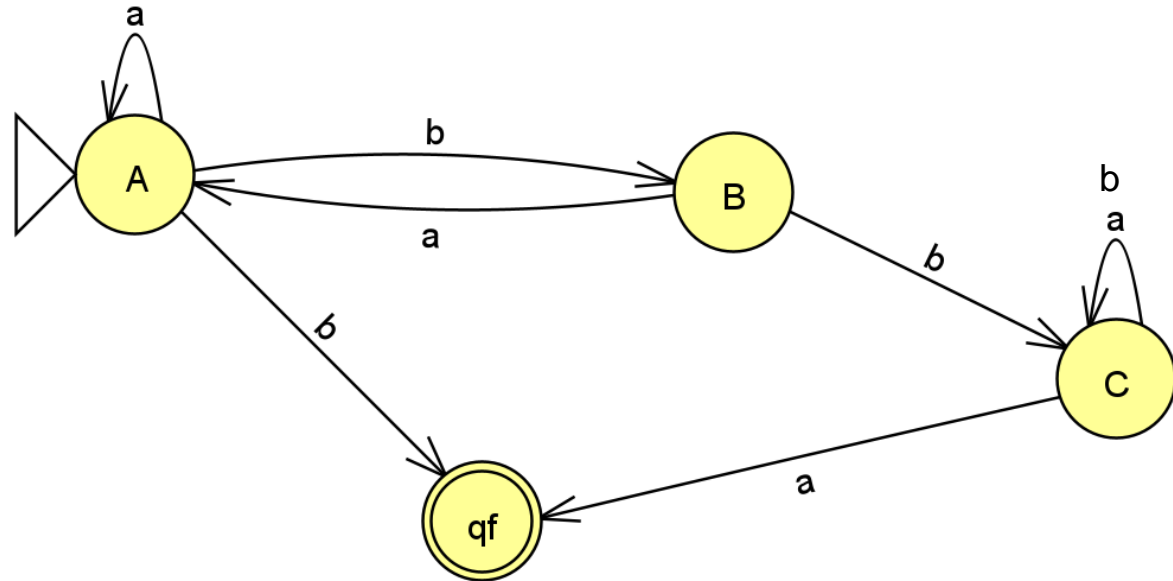
$B \rightarrow bC$

$C \rightarrow aC$

$C \rightarrow bC$

$C \rightarrow a$

$A \rightarrow b$



$\Sigma = \{a, b\}$

Observar que el AF obtenido es no determinístico.



