Universidad de La Laguna

Grado en Ingeniería Informática

Computabilidad y Algoritmia

# Tema 3: Lenguajes y gramáticas independientes del contexto

**Francisco de Sande González - Gara Miranda Valladares**

Curso 2019-2020

# Indice

# Indice

# Introducción

## Las expresiones regulares y los autómatas finitos

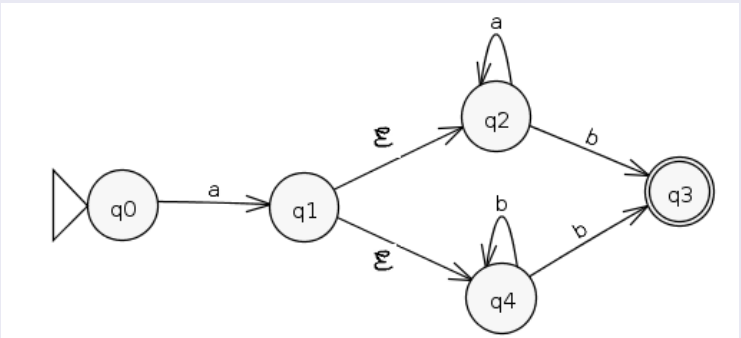
- Nos proporcionan dos medios para especificar o definir lenguajes.
- Las expresiones regulares nos proporcionan una plantilla o patrón para las cadenas del lenguaje.
- Un autómata finito especifica un lenguaje como el conjunto de todas las cadenas que lo hacen pasar del estado inicial a uno de sus estados de aceptación:
  - Interpretar un autómata como un *generador de cadenas del lenguaje*.

# Introducción

## Las expresiones regulares y los autómatas finitos

- Nos proporcionan dos medios para especificar o definir lenguajes.
- Las expresiones regulares nos proporcionan una plantilla o patrón para las cadenas del lenguaje.
- Un autómata finito especifica un lenguaje como el conjunto de todas las cadenas que lo hacen pasar del estado inicial a uno de sus estados de aceptación:
  - Interpretar un autómata como un *generador de cadenas del lenguaje*.

## Ejemplo: $a(a^*|b^*)b$



# Introducción

## Ejemplo: $a(a^*|b^*)b$

- Cadenas formadas por una "a" seguidas de una parte final: $S \rightarrow aE$
- Para la parte final tenemos dos opciones:
  - Cadenas de "aes" seguidas de una "b": $E \rightarrow A, A \rightarrow aA, A \rightarrow b$
  - Cadenas de "bes": $E \rightarrow B, B \rightarrow bB, B \rightarrow b$

# Introducción

## Ejemplo: $a(a^*|b^*)b$

- Cadenas formadas por una "a" seguidas de una parte final: $S \to aE$
- Para la parte final tenemos dos opciones:
  - Cadenas de "aes" seguidas de una "b": $E \to A, A \to aA, A \to b$
  - Cadenas de "bes": $E \to B, B \to bB, B \to b$

## Reglas de sustitución

- $S \to aE$
- $E \to A$
- $E \to B$
- $A \to b$
- $A \to aA$
- $B \to b$
- $B \to bB$

# Introducción

## Reglas de sustitución para la generación de cadenas

- $S \rightarrow aE$
- $E \rightarrow A$
- $E \rightarrow B$
- $A \rightarrow b$
- $A \rightarrow aA$
- $B \rightarrow b$
- $B \rightarrow bB$

# Introducción

## Reglas de sustitución para la generación de cadenas

- $S \rightarrow aE$
- $E \rightarrow A$
- $E \rightarrow B$
- $A \rightarrow b$
- $A \rightarrow aA$
- $B \rightarrow b$
- $B \rightarrow bB$

## Abreviando...

- $S \rightarrow aE$
- $E \rightarrow A|B$
- $A \rightarrow aA|b$
- $B \rightarrow bB|b$

# Introducción

## Reglas de sustitución para la generación de cadenas

- $S \rightarrow aE$
- $E \rightarrow A|B$
- $A \rightarrow aA|b$
- $B \rightarrow bB|b$

# Introducción

## Reglas de sustitución para la generación de cadenas

- $S \rightarrow aE$
- $E \rightarrow A|B$
- $A \rightarrow aA|b$
- $B \rightarrow bB|b$

## Elementos

- **Símbolos terminales**: símbolos del alfabeto $= \{a, b\}$
- **Símbolos no terminales**: colección de nuevos símbolos para representar a las porciones de cadenas que no han sido generadas $= \{S, E, A, B\}$
- **Reglas de producción**:
  - $<$ No terminal $> \rightarrow <$ Terminales y no terminales $>$
  - Lado izquierdo y lado derecho de las reglas de producción.
  - El símbolo que se encuentra a la izquierda de la flecha se puede sustituir por la cadena de la derecha.

# Introducción

## Reglas de sustitución para la generación de cadenas

- $S \rightarrow aE$
- $E \rightarrow A|B$
- $A \rightarrow aA|b$
- $B \rightarrow bB|b$

# Introducción

## Reglas de sustitución para la generación de cadenas

- $S \rightarrow aE$
- $E \rightarrow A|B$
- $A \rightarrow aA|b$
- $B \rightarrow bB|b$

## Ejemplo para generar la cadena $aaab$:

- Comenzar con el símbolo de arranque $S$.
- Coger el símbolo no terminal de la cadena y reemplazarlo por el lado derecho de alguna regla que tenga ese símbolo en la parte izquierda.
- Repetir el paso anterior hasta que todos los símbolos de la cadena sean símbolos terminales.

# Introducción

## Reglas de sustitución para la generación de cadenas

- $S \to aE$
- $E \to A|B$
- $A \to aA|b$
- $B \to bB|b$

## Ejemplo para generar la cadena $aaab$:

- Comenzar con el símbolo de arranque $S$.
- Coger el símbolo no terminal de la cadena y reemplazarlo por el lado derecho de alguna regla que tenga ese símbolo en la parte izquierda.
- Repetir el paso anterior hasta que todos los símbolos de la cadena sean símbolos terminales.

$$S \Rightarrow aE \Rightarrow aA \Rightarrow aaA \Rightarrow aaaA \Rightarrow aaab$$

# Indice

# Gramática regular

## Definición

Una *gramática regular* $G$ es una 4-tupla $G = (\Sigma, N, S, P)$, donde:

# Gramática regular

## Definición

Una *gramática regular* $G$ es una 4-tupla $G = (\Sigma, N, S, P)$, donde:

- $\Sigma$ es un alfabeto.

# Gramática regular

## Definición

Una *gramática regular* $G$ es una 4-tupla $G = (\Sigma, N, S, P)$, donde:

- $\Sigma$ es un alfabeto.
- $N$ es una colección de símbolos no terminales.

# Gramática regular

## Definición

Una *gramática regular* $G$ es una 4-tupla $G = (\Sigma, N, S, P)$, donde:

- $\Sigma$ es un alfabeto.
- $N$ es una colección de símbolos no terminales.
- $S$ es un no terminal llamado *símbolo inicial o de arranque*.

# Gramática regular

## Definición

Una *gramática regular* $G$ es una 4-tupla $G = (\Sigma, N, S, P)$, donde:

- $\Sigma$ es un alfabeto.
- $N$ es una colección de símbolos no terminales.
- $S$ es un no terminal llamado *símbolo inicial o de arranque*.
- $P$ es una colección de reglas de sustitución denominadas *producciones*, tal que:

# Gramática regular

## Definición

Una *gramática regular* $G$ es una 4-tupla $G = (\Sigma, N, S, P)$, donde:

- $\Sigma$ es un alfabeto.
- $N$ es una colección de símbolos no terminales.
- $S$ es un no terminal llamado *símbolo inicial o de arranque*.
- $P$ es una colección de reglas de sustitución denominadas *producciones*, tal que:
  - Todas las producciones son regulares por la derecha: $A \to uB|v$ $(A, B \in N$ y $u, v \in \Sigma^*)$

# Gramática regular

## Definición

Una *gramática regular* $G$ es una 4-tupla $G = (\Sigma, N, S, P)$, donde:

- $\Sigma$ es un alfabeto.
- $N$ es una colección de símbolos no terminales.
- $S$ es un no terminal llamado *símbolo inicial o de arranque*.
- $P$ es una colección de reglas de sustitución denominadas *producciones*, tal que:
  - Todas las producciones son regulares por la derecha: $A \to uB|v$ $(A, B \in N$ y $u, v \in \Sigma^*)$
  - O, todas las producciones son regulares por la izquierda: $A \to Bu|v$ $(A, B \in N$ y $u, v \in \Sigma^*)$

# Gramática regular

## Definición

Una *gramática es lineal por la derecha* si todas sus producciones son regulares por la derecha.

# Gramática regular

## Definición

Una *gramática es lineal por la derecha* si todas sus producciones son regulares por la derecha.

## Definición

Una *gramática es lineal por la izquierda* si todas sus producciones son regulares por la izquierda.

# Gramática regular

**Definición**

Una *gramática es lineal por la derecha* si todas sus producciones son regulares por la derecha.

**Definición**

Una *gramática es lineal por la izquierda* si todas sus producciones son regulares por la izquierda.

**Definición**

Una *gramática es regular* si es lineal por la derecha o lineal por la izquierda.

# Gramática regular

## Ejemplo: $0(10)^*$

- Gramática lineal por la derecha:
  $S \to 0A$
  $A \to 10A|\varepsilon$
- Gramática lineal por la izquierda:
  $S \to S10|0$

# Gramática regular

## Definición

El *lenguaje generado* por la gramática regular $G$ se denota por $L(G)$.

# Gramática regular

## Definición

El *lenguaje generado* por la gramática regular $G$ se denota por $L(G)$.

## Ejemplo

Sea la gramática $G = (\Sigma, N, S, P)$, donde:

- $\Sigma = \{a, b\}$
- $N = \{S, A\}$
- $P$:
  - $S \to bA$
  - $A \to aaA|b|\varepsilon$

# Gramática regular

## Definición

El *lenguaje generado* por la gramática regular $G$ se denota por $L(G)$.

## Ejemplo

Sea la gramática $G = (\Sigma, N, S, P)$, donde:

- $\Sigma = \{a, b\}$
- $N = \{S, A\}$
- $P$:
  - $S \to bA$
  - $A \to aaA|b|\varepsilon$

Entonces $L(G) = b(aa)^*(b|\varepsilon)$

# Gramática regular a partir de un DFA

Sea $L$ un lenguaje regular.
Ya hemos visto que si $L$ es un lenguaje regular, entonces existe un DFA $M(\Sigma, Q, s, F, \delta)$, tal que $L = L(M)$.

# Gramática regular a partir de un DFA

Sea $L$ un lenguaje regular.
Ya hemos visto que si $L$ es un lenguaje regular, entonces existe un DFA $M(\Sigma, Q, s, F, \delta)$, tal que $L = L(M)$.

**Teorema**

Existe una gramática regular $G \equiv (\Sigma', N, S, P)$ tal que $L(G) = L(M)$.

# Gramática regular a partir de un DFA

Sea $L$ un lenguaje regular.
Ya hemos visto que si $L$ es un lenguaje regular, entonces existe un DFA $M(\Sigma, Q, s, F, \delta)$, tal que $L = L(M)$.

## Teorema

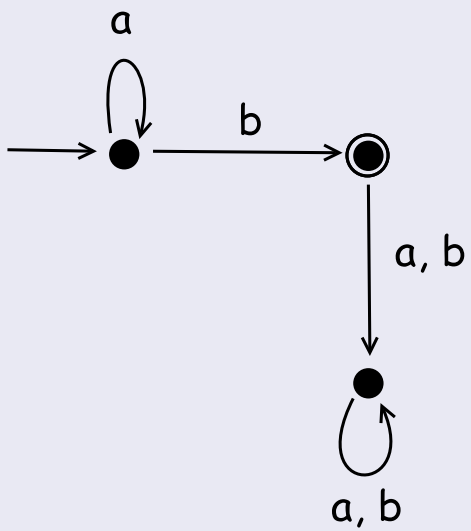
Existe una gramática regular $G \equiv (\Sigma', N, S, P)$ tal que $L(G) = L(M)$.

## Definimos $G$ como:

- $\Sigma' = \Sigma$
- $N = Q$
- $S = s$
- $P = \{q \to ap \mid \delta(q, a) = p\} \cup \{q \to \varepsilon \mid q \in F\}$

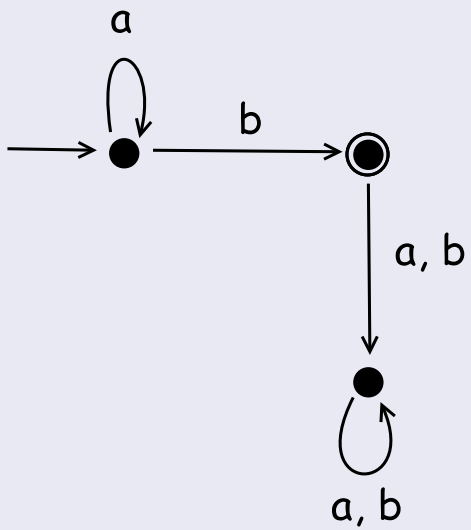
# Gramática regular a partir de un DFA

## Ejemplo: $a^*b$



# Gramática regular a partir de un DFA

## Ejemplo: $a^*b$




- $S \rightarrow aS|bA$
- $A \rightarrow aB|bB|\varepsilon$
- $B \rightarrow aB|bB$

# NFA a partir de gramática regular

Sea $G \equiv (V, \Sigma, S, P)$ una gramática regular.

**Teorema**

Existe un NFA $M \equiv (Q, \Sigma', \delta, q_0, F)$ tal que $L(M) = L(G)$.

# NFA a partir de gramática regular

Sea $G \equiv (V, \Sigma, S, P)$ una gramática regular.

## Teorema

Existe un NFA $M \equiv (Q, \Sigma', \delta, q_0, F)$ tal que $L(M) = L(G)$.

Inicialmente el NFA se define como:
$\Sigma' = \Sigma$, $Q = V \cup \{f\}$, $F = \{f\}$, $q_0 = S$

# NFA a partir de gramática regular

Sea $G \equiv (V, \Sigma, S, P)$ una gramática regular.

**Teorema**

Existe un NFA $M \equiv (Q, \Sigma', \delta, q_0, F)$ tal que $L(M) = L(G)$.

Inicialmente el NFA se define como:
$\Sigma' = \Sigma$, $Q = V \cup \{f\}$, $F = \{f\}$, $q_0 = S$

Luego, empezamos a añadir estados a $Q$:

# NFA a partir de gramática regular

Sea $G \equiv (V, \Sigma, S, P)$ una gramática regular.

## Teorema

Existe un NFA $M \equiv (Q, \Sigma', \delta, q_0, F)$ tal que $L(M) = L(G)$.

Inicialmente el NFA se define como:
$\Sigma' = \Sigma$, $Q = V \cup \{f\}$, $F = \{f\}$, $q_0 = S$

Luego, empezamos a añadir estados a $Q$:

- Si $(A \to a_1 a_2 ... a_n B) \in P$:
  - $Q = Q \cup \{q_1, q_2, ..., q_{n-1}\}$
  - $\delta(A, a_1 a_2 ... a_n) = \delta(q_1, a_2 ... a_n) = ... = \delta(q_{n-1}, a_n) = B$

# NFA a partir de gramática regular

Sea $G \equiv (V, \Sigma, S, P)$ una gramática regular.

## Teorema

Existe un NFA $M \equiv (Q, \Sigma', \delta, q_0, F)$ tal que $L(M) = L(G)$.

Inicialmente el NFA se define como:
$\Sigma' = \Sigma$, $Q = V \cup \{f\}$, $F = \{f\}$, $q_0 = S$

Luego, empezamos a añadir estados a $Q$:

- Si $(A \to a_1 a_2 ... a_n B) \in P$:
  - $Q = Q \cup \{q_1, q_2, ..., q_{n-1}\}$
  - $\delta(A, a_1 a_2 ... a_n) = \delta(q_1, a_2 ... a_n) = ... = \delta(q_{n-1}, a_n) = B$
- Si $(A \to c_1 c_2 ... c_m) \in P$:
  - $Q = Q \cup \{p_1, p_2, ..., p_{m-1}\}$
  - $\delta(A, c_1 c_2 ... c_m) = \delta(p_1, c_2 ... c_m) = ... = \delta(p_{m-1}, c_m) = f$

# NFA a partir de gramática regular

## Ejemplo

- $S \to aB|bA|\varepsilon$
- $A \to abaS$
- $B \to babS$

# NFA a partir de gramática regular

## Ejemplo

- $S \to aB|bA|\varepsilon$
- $A \to abaS$
- $B \to babS$

## Inicialmente:

$Q = \{S, A, B, f\} \quad F = \{f\} \quad q_0 = S$

# NFA a partir de gramática regular

## Ejemplo

- $S \to aB|bA|\varepsilon$
- $A \to abaS$
- $B \to babS$

## Inicialmente:

$Q = \{S, A, B, f\} \quad F = \{f\} \quad q_0 = S$

## Analizamos regla a regla:

# NFA a partir de gramática regular

## Ejemplo

- $S \to aB|bA|\varepsilon$
- $A \to abaS$
- $B \to babS$

## Inicialmente:

$Q = \{S, A, B, f\} \quad F = \{f\} \quad q_0 = S$

## Analizamos regla a regla:

- $S \to aB$
  - $Q = \{S, A, B, f\}$
  - $\delta(S, a) = B$

# NFA a partir de gramática regular

## Ejemplo

- $S \to aB|bA|\varepsilon$
- $A \to abaS$
- $B \to babS$

## Inicialmente:

$Q = \{S, A, B, f\} \quad F = \{f\} \quad q_0 = S$

## Analizamos regla a regla:

# NFA a partir de gramática regular

## Ejemplo

- $S \to aB|bA|\varepsilon$
- $A \to abaS$
- $B \to babS$

## Inicialmente:

$Q = \{S, A, B, f\} \quad F = \{f\} \quad q_0 = S$

## Analizamos regla a regla:

- $S \to bA$
  - $Q = \{S, A, B, f\}$
  - $\delta(S, b) = A$

# NFA a partir de gramática regular

## Ejemplo

- $S \to aB|bA|\varepsilon$
- $A \to abaS$
- $B \to babS$

## Inicialmente:

$Q = \{S, A, B, f\} \quad F = \{f\} \quad q_0 = S$

## Analizamos regla a regla:

# NFA a partir de gramática regular

## Ejemplo

- $S \to aB|bA|\varepsilon$
- $A \to abaS$
- $B \to babS$

## Inicialmente:

$Q = \{S, A, B, f\} \quad F = \{f\} \quad q_0 = S$

## Analizamos regla a regla:

- $S \to \varepsilon$
  - $Q = \{S, A, B, f\}$
  - $\delta(S, \varepsilon) = f$

# NFA a partir de gramática regular

## Ejemplo

- $S \to aB|bA|\varepsilon$
- $A \to abaS$
- $B \to babS$

## Inicialmente:

$Q = \{S, A, B, f\} \quad F = \{f\} \quad q_0 = S$

## Analizamos regla a regla:

# NFA a partir de gramática regular

## Ejemplo

- $S \to aB|bA|\varepsilon$
- $A \to abaS$
- $B \to babS$

## Inicialmente:

$Q = \{S, A, B, f\} \quad F = \{f\} \quad q_0 = S$

## Analizamos regla a regla:

- $A \to abaS$
  - $Q = Q \cup \{1, 2\} = \{S, A, B, f, 1, 2\}$
  - $\delta(A, aba) = \delta(1, ba) = \delta(2, a) = S$

# NFA a partir de gramática regular

## Ejemplo

- $S \to aB|bA|\varepsilon$
- $A \to abaS$
- $B \to babS$

## Actualmente:

$Q = \{S, A, B, f, 1, 2\} \quad F = \{f\} \quad q_0 = S$

## Analizamos regla a regla:

# NFA a partir de gramática regular

## Ejemplo

- $S \to aB|bA|\varepsilon$
- $A \to abaS$
- $B \to babS$

## Actualmente:

$Q = \{S, A, B, f, 1, 2\} \quad F = \{f\} \quad q_0 = S$

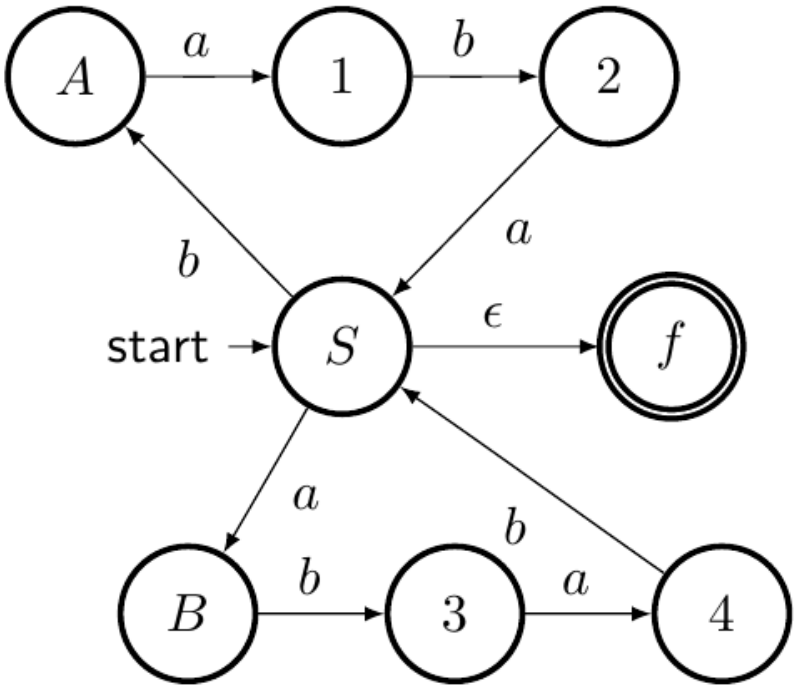
## Analizamos regla a regla:

- $B \to babS$
  - $Q = Q \cup \{3, 4\} = \{S, A, B, f, 1, 2, 3, 4\}$
  - $\delta(B, bab) = \delta(3, ab) = \delta(4, b) = S$

# NFA a partir de gramática regular

## Ejemplo

- $S \rightarrow aB|bA|\varepsilon$
- $A \rightarrow abaS$
- $B \rightarrow babS$



# Indice

# Gramática independiente del contexto

## Definición

Una *gramática independiente del contexto* (context-free grammar, $CFG$) es una 4-tupla $G = (V, \Sigma, S, P)$ donde:

# Gramática independiente del contexto

## Definición

Una *gramática independiente del contexto* (context-free grammar, $CFG$) es una 4-tupla $G = (V, \Sigma, S, P)$ donde:

- $V$ es el conjunto de símbolos no terminales ($V \neq \emptyset$).

# Gramática independiente del contexto

## Definición

Una *gramática independiente del contexto* (context-free grammar, $CFG$) es una 4-tupla $G = (V, \Sigma, S, P)$ donde:

- $V$ es el conjunto de símbolos no terminales ($V \neq \emptyset$).
- $\Sigma$ es el conjunto de símbolos terminales o alfabeto de la gramática ($V \cap \Sigma = \emptyset$).

# Gramática independiente del contexto

## Definición

Una *gramática independiente del contexto* (context-free grammar, $CFG$) es una 4-tupla $G = (V, \Sigma, S, P)$ donde:

- $V$ es el conjunto de símbolos no terminales ($V \neq \emptyset$).
- $\Sigma$ es el conjunto de símbolos terminales o alfabeto de la gramática ($V \cap \Sigma = \emptyset$).
- $S$ es un no terminal que se llama símbolo de arranque o axioma de la gramática ($S \in V$).

# Gramática independiente del contexto

## Definición

Una *gramática independiente del contexto* (context-free grammar, $CFG$) es una 4-tupla $G = (V, \Sigma, S, P)$ donde:

- $V$ es el conjunto de símbolos no terminales ($V \neq \emptyset$).
- $\Sigma$ es el conjunto de símbolos terminales o alfabeto de la gramática ($V \cap \Sigma = \emptyset$).
- $S$ es un no terminal que se llama símbolo de arranque o axioma de la gramática ($S \in V$).
- $P$ es un conjunto de reglas de producción:

$$P = \{A \to \alpha \mid A \in V,\ \alpha \in (V \cup \Sigma)^*\}$$
$$P \subseteq V \times (V \cup \Sigma)^*$$
$$P \neq \emptyset$$

# Convenio de notación

# Convenio de notación

- Habitualmente, para especificar una gramática, se especifican todas sus reglas de producción $P$, y se sigue un convenio de notación que permite determinar todos los elementos.

# Convenio de notación

- Habitualmente, para especificar una gramática, se especifican todas sus reglas de producción $P$, y se sigue un convenio de notación que permite determinar todos los elementos.
- El símbolo de arranque es aquel cuyas producciones aparecen en primer lugar en la lista.

# Convenio de notación

- Habitualmente, para especificar una gramática, se especifican todas sus reglas de producción $P$, y se sigue un convenio de notación que permite determinar todos los elementos.
- El símbolo de arranque es aquel cuyas producciones aparecen en primer lugar en la lista.
- Los símbolos terminales se representan mediante letras minúsculas del principio del alfabeto latino: $a, b, c, d, ... \in \Sigma$

# Convenio de notación

- Habitualmente, para especificar una gramática, se especifican todas sus reglas de producción $P$, y se sigue un convenio de notación que permite determinar todos los elementos.
- El símbolo de arranque es aquel cuyas producciones aparecen en primer lugar en la lista.
- Los símbolos terminales se representan mediante letras minúsculas del principio del alfabeto latino: $a, b, c, d, ... \in \Sigma$
- Los símbolos no terminales se representan mediante letras mayúsculas del principio del alfabeto latino: $A, B, C, D, ... \in V$

# Convenio de notación

- Habitualmente, para especificar una gramática, se especifican todas sus reglas de producción $P$, y se sigue un convenio de notación que permite determinar todos los elementos.
- El símbolo de arranque es aquel cuyas producciones aparecen en primer lugar en la lista.
- Los símbolos terminales se representan mediante letras minúsculas del principio del alfabeto latino: $a, b, c, d, ... \in \Sigma$
- Los símbolos no terminales se representan mediante letras mayúsculas del principio del alfabeto latino: $A, B, C, D, ... \in V$
- Los símbolos gramaticales (terminales y no terminales) se representan mediante letras mayúsculas del final del alfabeto latino: $X, Y, W, Z, ... \in (\Sigma \cup V)$

# Convenio de notación

- Las cadenas (secuencias) de símbolos terminales se representan mediante letras minúsculas del final del alfabeto latino: $w, x, y, u, ... \in \Sigma^*$

# Convenio de notación

- Las cadenas (secuencias) de símbolos terminales se representan mediante letras minúsculas del final del alfabeto latino: $w, x, y, u, ... \in \Sigma^*$
- Las cadenas de símbolos gramaticales se representan mediante letras minúsculas del principio del alfabeto griego: $\alpha, \beta, \gamma, ... \in (\Sigma \cup V)^*$

# Convenio de notación

- Las cadenas (secuencias) de símbolos terminales se representan mediante letras minúsculas del final del alfabeto latino: $w, x, y, u, ... \in \Sigma^*$
- Las cadenas de símbolos gramaticales se representan mediante letras minúsculas del principio del alfabeto griego: $\alpha, \beta, \gamma, ... \in (\Sigma \cup V)^*$
- Si hay varias producciones para un mismo símbolo no terminal, las diferentes alternativas se listan separadas por barras verticales: $A \to \alpha|\beta|...|\gamma$

# Derivaciones

## Sea la siguiente gramática $G = (V, \Sigma, S, P)$ para expresiones aritméticas:

- $V = \{E\}$
- $\Sigma = \{+, *, (, ), -, a\}$
- $S = E$
- $P$: $E \rightarrow E + E | E * E | (E) | - E | a$

# Derivaciones

## Sea la siguiente gramática $G = (V, \Sigma, S, P)$ para expresiones aritméticas:

- $V = \{E\}$
- $\Sigma = \{+, *, (, ), -, a\}$
- $S = E$
- $P$: $E \to E + E | E * E | (E) | - E | a$

## Podemos partir del símbolo de arranque $E$ y realizar reescrituras para obtener una secuencia de sustituciones:

- $E \Rightarrow E * E \Rightarrow (E) * E \Rightarrow (-E) * E \Rightarrow (-E) * a \Rightarrow (-a) * a$
- $E \Rightarrow E + E \Rightarrow E + E * E \Rightarrow E + a * E \Rightarrow E + a * a \Rightarrow a + a * a$
- A la secuencia de sustituciones se le llama una *derivación*: decimos que la cadena ha sido derivada a partir del símbolo de arranque $E$.
- La existencia de estas derivaciones muestra que casos particulares de expresiones son $(-a) * a$ o bien $a + a * a$.

# Derivaciones

## Definición

Dada una CFG, $G \equiv (V, \Sigma, S, P)$, se dice que $\alpha A \beta \Rightarrow \alpha \gamma \beta$ ($\alpha A \beta$ deriva en $\alpha \gamma \beta$) si $(A \rightarrow \gamma) \in P$.
Si $\alpha_1 \Rightarrow \alpha_2 \Rightarrow ... \Rightarrow \alpha_n$ se dice que $\alpha_1$ deriva en $\alpha_n$

# Derivaciones

## Definición

Dada una CFG, $G \equiv (V, \Sigma, S, P)$, se dice que $\alpha A \beta \Rightarrow \alpha \gamma \beta$ ($\alpha A \beta$ deriva en $\alpha \gamma \beta$) si $(A \rightarrow \gamma) \in P$.
Si $\alpha_1 \Rightarrow \alpha_2 \Rightarrow ... \Rightarrow \alpha_n$ se dice que $\alpha_1$ deriva en $\alpha_n$

## Notación para las derivaciones

- El símbolo $\Rightarrow$ se lee "deriva en un paso"
- Para expresar que "deriva en cero o más pasos" utilizaremos $\Rightarrow^*$
- Para expresar que "deriva en uno o más pasos" utilizaremos $\Rightarrow^+$

# Derivaciones

## Definición

Dada una CFG, $G \equiv (V, \Sigma, S, P)$, se dice que $\alpha A \beta \Rightarrow \alpha \gamma \beta$ ($\alpha A \beta$ deriva en $\alpha \gamma \beta$) si $(A \to \gamma) \in P$.
Si $\alpha_1 \Rightarrow \alpha_2 \Rightarrow ... \Rightarrow \alpha_n$ se dice que $\alpha_1$ deriva en $\alpha_n$

## Notación para las derivaciones

- El símbolo $\Rightarrow$ se lee "deriva en un paso"
- Para expresar que "deriva en cero o más pasos" utilizaremos $\Rightarrow^*$
- Para expresar que "deriva en uno o más pasos" utilizaremos $\Rightarrow^+$

## Definiciones

- Se dice que $\alpha \in (V \cup \Sigma)^*$ es una *forma sentencial* si existe una derivación tal que $S \Rightarrow^* \alpha$
- Una *frase o sentencia* es una forma sentencial que consta sólo de símbolos terminales.

# Lenguaje independiente del contexto

## Definición

El lenguaje generado por una CFG $G \equiv (V, \Sigma, S, P)$, se denota por $L(G)$ y se llama *lenguaje independiente del contexto*:

$$L(G) = \{w \in \Sigma^* \mid S \Rightarrow^* w\}$$

$L(G)$ es el conjunto de todas las cadenas que pueden ser derivadas partiendo del símbolo de arranque de $G$.

# Lenguaje independiente del contexto

## Definición

El lenguaje generado por una CFG $G \equiv (V, \Sigma, S, P)$, se denota por $L(G)$ y se llama *lenguaje independiente del contexto*:

$$L(G) = \{w \in \Sigma^* \mid S \Rightarrow^* w\}$$

$L(G)$ es el conjunto de todas las cadenas que pueden ser derivadas partiendo del símbolo de arranque de $G$.

## Equivalencia de gramáticas

Se dice que dos gramáticas son equivalentes, si y sólo si generan el mismo lenguaje:

$$G_1 \equiv G_2 \Leftrightarrow L(G_1) = L(G_2)$$

# Lenguaje independiente del contexto

## Ejemplo

Sea la gramática independiente del contexto $G$ dada por:

$$S \to aSb|\varepsilon$$

# Lenguaje independiente del contexto

## Ejemplo

Sea la gramática independiente del contexto $G$ dada por:

$$S \to aSb|\varepsilon$$

¿Cuál es el lenguaje generado por la gramática $G$?:

# Lenguaje independiente del contexto

## Ejemplo

Sea la gramática independiente del contexto $G$ dada por:

$$S \to aSb|\varepsilon$$

¿Cuál es el lenguaje generado por la gramática $G$?:

$$L(G) = \{a^n b^n \mid n \geq 0\}$$

# Lenguaje independiente del contexto

## Ejemplo

Sea la gramática independiente del contexto $G$ dada por:

$$S \to aSb|\varepsilon$$

¿Cuál es el lenguaje generado por la gramática $G$?:

$$L(G) = \{a^n b^n \mid n \geq 0\}$$

## ¿Es un lenguaje regular?

- Puesto que toda gramática regular es una gramática independiente del contexto, se tiene que todo lenguaje regular es un lenguaje independiente del contexto.
- El conjunto de los lenguajes independientes del contexto contiene a los lenguajes regulares, pero hay lenguajes independientes del contexto que no son lenguajes regulares.

# Lenguaje independiente del contexto

## Ejemplo

Una CFG para $L_1$: cadenas sobre $\{a, b\}$ que contienen la subcadena $aba$:
$S \to AabaA$
$A \to aA|bA|\varepsilon$

# Lenguaje independiente del contexto

## Ejemplo

Una CFG para $L_1$: cadenas sobre $\{a, b\}$ que contienen la subcadena $aba$:
$S \to AabaA$
$A \to aA|bA|\varepsilon$

## Ejemplo

Una CFG para $L_2$: cadenas sobre $\{a, b\}$ con el mismo número de "aes" que de "bes":
$S \to aSb|bSa|\varepsilon$

# Lenguaje independiente del contexto

## Ejemplo

Una CFG para $L_1$: cadenas sobre $\{a, b\}$ que contienen la subcadena $aba$:
$S \to AabaA$
$A \to aA|bA|\varepsilon$

## Ejemplo

Una CFG para $L_2$: cadenas sobre $\{a, b\}$ con el mismo número de "aes" que de "bes":
$S \to aSb|bSa|\varepsilon$ ¿Y la cadena $abba$?

# Lenguaje independiente del contexto

## Ejemplo

Una CFG para $L_1$: cadenas sobre $\{a, b\}$ que contienen la subcadena $aba$:
$S \to AabaA$
$A \to aA|bA|\varepsilon$

## Ejemplo

Una CFG para $L_2$: cadenas sobre $\{a, b\}$ con el mismo número de "aes" que de "bes":
$S \to aSb|bSa|\varepsilon$ ¿Y la cadena $abba$?

## Ejemplo

Una CFG para $L_3$: $\{ww^i \mid w \in \{a, b\}^*\}$:
$S \to aSa|bSb|\varepsilon$

# La clasificación de Chomsky

Las gramáticas pueden ser de cuatro tipos, según la complejidad de las reglas (producciones) que las definen:

# La clasificación de Chomsky

Las gramáticas pueden ser de cuatro tipos, según la complejidad de las reglas (producciones) que las definen:

- **Gramáticas de tipo 0** (sin restricciones o estructuradas por frases): incluye a todas las gramáticas formales y genera los lenguajes recursivamente enumerables.

# La clasificación de Chomsky

Las gramáticas pueden ser de cuatro tipos, según la complejidad de las reglas (producciones) que las definen:

- **Gramáticas de tipo 0** (sin restricciones o estructuradas por frases): incluye a todas las gramáticas formales y genera los lenguajes recursivamente enumerables.
- **Gramáticas de tipo 1** (gramáticas sensibles al contexto): generan los lenguajes sensibles o dependientes del contexto.

# La clasificación de Chomsky

Las gramáticas pueden ser de cuatro tipos, según la complejidad de las reglas (producciones) que las definen:

- **Gramáticas de tipo 0** (sin restricciones o estructuradas por frases): incluye a todas las gramáticas formales y genera los lenguajes recursivamente enumerables.
- **Gramáticas de tipo 1** (gramáticas sensibles al contexto): generan los lenguajes sensibles o dependientes del contexto.
- **Gramáticas de tipo 2** (gramáticas independientes del contexto): generan los lenguajes independientes del contexto.

# La clasificación de Chomsky

Las gramáticas pueden ser de cuatro tipos, según la complejidad de las reglas (producciones) que las definen:

- **Gramáticas de tipo 0** (sin restricciones o estructuradas por frases): incluye a todas las gramáticas formales y genera los lenguajes recursivamente enumerables.
- **Gramáticas de tipo 1** (gramáticas sensibles al contexto): generan los lenguajes sensibles o dependientes del contexto.
- **Gramáticas de tipo 2** (gramáticas independientes del contexto): generan los lenguajes independientes del contexto.
- **Gramáticas de tipo 3** (gramáticas regulares): generan los lenguajes regulares.

# La clasificación de Chomsky

Las gramáticas pueden ser de cuatro tipos, según la complejidad de las reglas (producciones) que las definen:

- **Gramáticas de tipo 0** (sin restricciones o estructuradas por frases): incluye a todas las gramáticas formales y genera los lenguajes recursivamente enumerables.
- **Gramáticas de tipo 1** (gramáticas sensibles al contexto): generan los lenguajes sensibles o dependientes del contexto.
- **Gramáticas de tipo 2** (gramáticas independientes del contexto): generan los lenguajes independientes del contexto.
- **Gramáticas de tipo 3** (gramáticas regulares): generan los lenguajes regulares.

Cada tipo de lenguaje está contenido en el anterior:

$L(G_3) \subset L(G_2) \subset L(G_1) \subset L(G_0)$

# La clasificación de Chomsky

## Gramáticas de tipo 0

- Son las más generales.
- Sus reglas no tienen ninguna restricción y son de la forma $\alpha \to \beta$, donde $\alpha$ y $\beta$ son cadenas arbitrarias de símbolos gramaticales con $\alpha \neq \varepsilon$
- Se definen como $G \equiv (V, \Sigma, P, S)$.
- Estas gramáticas generan los lenguajes recursivamente enumerables o los lenguajes sin restricciones, aunque a pesar de este nombre no son capaces de representar los lenguajes naturales.

# La clasificación de Chomsky

## Gramáticas de tipo 0

Por ejemplo, la gramática siguiente:

$S \to ACaB$
$Ca \to aaC$
$CB \to DB$
$CB \to E$
$aD \to Da$
$AD \to AC$
$aE \to Ea$
$AE \to \varepsilon$

Genera el lenguaje $L = \{a^i \mid i \text{ es una potencia positiva de } 2\}$

# La clasificación de Chomsky

## Gramáticas de tipo 1

- Sus reglas son de la forma:

$$\alpha \to \beta \text{ donde } |\beta| \geq |\alpha|$$

- Es decir, las derivaciones sucesivas no pueden disminuir la longitud de la cadena.
- Se puede demostrar que este tipo de reglas son equivalentes a las de la forma:

$$\alpha A \beta \to \alpha \gamma \beta$$

donde la sustitución del símbolo gramatical $A$ por $\gamma$ depende del contexto en que aparezca $A$.

# La clasificación de Chomsky

## Gramáticas de tipo 2

- Son gramáticas independientes del contexto (CFG).
- Sus reglas son de la forma:

$$A \to \alpha$$

- La sustitución de $A$ por $\alpha$ se puede realizar independientemente de dónde aparezca el símbolo $A$.
- La mayor parte de los lenguajes de programación están generados por una CFG (aumentada con elementos contextuales necesarios para la semántica del lenguaje).

# La clasificación de Chomsky

## Gramáticas de tipo 3

- Son gramáticas regulares.
- Son las más simples.
- Las gramáticas regulares caracterizan a los lenguajes regulares: un lenguaje es regular si y sólo si tiene una gramática lineal por la izquierda o si y sólo si tiene una gramática lineal por la derecha.

# Árboles de Análisis Sintáctico (AAS)

## Un AAS es una representación gráfica de una derivación

El árbol se define de la manera siguiente:

# Árboles de Análisis Sintáctico (AAS)

## Un AAS es una representación gráfica de una derivación

El árbol se define de la manera siguiente:

- El nodo raíz está etiquetado con el símbolo de arranque de la gramática.

# Árboles de Análisis Sintáctico (AAS)

## Un AAS es una representación gráfica de una derivación

El árbol se define de la manera siguiente:

- El nodo raíz está etiquetado con el símbolo de arranque de la gramática.
- Los nodos internos del árbol se etiquetan con símbolos no terminales ($A \in V$).

# Árboles de Análisis Sintáctico (AAS)

## Un AAS es una representación gráfica de una derivación

El árbol se define de la manera siguiente:

- El nodo raíz está etiquetado con el símbolo de arranque de la gramática.
- Los nodos internos del árbol se etiquetan con símbolos no terminales ($A \in V$).
- Los nodos hoja del árbol se etiquetan con símbolos terminales o con $\varepsilon$ ($\Sigma \cup \{\varepsilon\}$)

# Árboles de Análisis Sintáctico (AAS)

## Un AAS es una representación gráfica de una derivación

El árbol se define de la manera siguiente:

- El nodo raíz está etiquetado con el símbolo de arranque de la gramática.
- Los nodos internos del árbol se etiquetan con símbolos no terminales ($A \in V$).
- Los nodos hoja del árbol se etiquetan con símbolos terminales o con $\varepsilon$ ($\Sigma \cup \{\varepsilon\}$)
- Si la producción $A \to X_1 X_2 ... X_n$ se ha utilizado en la derivación de la cadena, entonces en el AAS, el nodo A aparecerá teniendo como nodos hijos a $X_1, X_2, ..., X_n$

# Árboles de Análisis Sintáctico (AAS)

## Un AAS es una representación gráfica de una derivación

El árbol se define de la manera siguiente:

- El nodo raíz está etiquetado con el símbolo de arranque de la gramática.
- Los nodos internos del árbol se etiquetan con símbolos no terminales $(A \in V)$.
- Los nodos hoja del árbol se etiquetan con símbolos terminales o con $\varepsilon$ $(\Sigma \cup \{\varepsilon\})$
- Si la producción $A \to X_1 X_2 ... X_n$ se ha utilizado en la derivación de la cadena, entonces en el AAS, el nodo A aparecerá teniendo como nodos hijos a $X_1, X_2, ..., X_n$
- La cadena derivada puede leerse de izquierda a derecha en los nodos hoja del AAS.

# Árboles de Análisis Sintáctico (AAS)

## Ejemplo

$S \rightarrow AB$
$A \rightarrow aA|\varepsilon$
$B \rightarrow bB|\varepsilon$

# Árboles de Análisis Sintáctico (AAS)

## Ejemplo

$S \to AB$
$A \to aA|\varepsilon$
$B \to bB|\varepsilon$

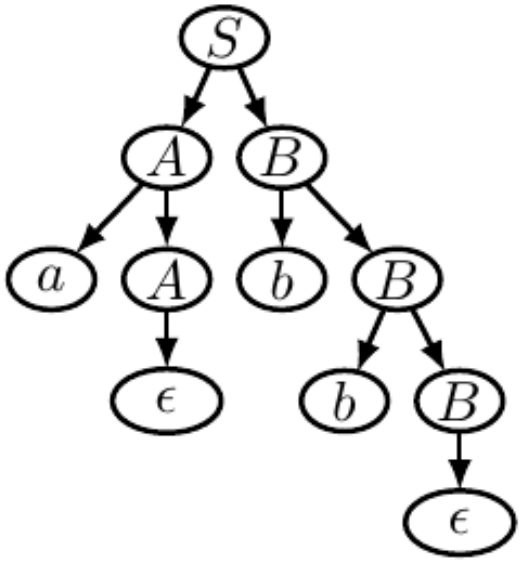
Podemos realizar la derivación siguiente:
$S \Rightarrow AB \Rightarrow aAB \Rightarrow aAbB \Rightarrow abB \Rightarrow abbB \Rightarrow abb$

# Árboles de Análisis Sintáctico (AAS)

## Ejemplo

$S \to AB$
$A \to aA|\varepsilon$
$B \to bB|\varepsilon$

Podemos realizar la derivación siguiente:
$S \Rightarrow AB \Rightarrow aAB \Rightarrow aAbB \Rightarrow abB \Rightarrow abbB \Rightarrow abb$



# Árboles de Análisis Sintáctico (AAS)

## Ejemplo

$S \to AB$
$A \to aA|a$
$B \to bB|b$

# Árboles de Análisis Sintáctico (AAS)

## Ejemplo

$S \to AB$
$A \to aA|a$
$B \to bB|b$

## Obtener el AAS correspondiente a la cadena $w = abbab$:

- No es posible obtener el AAS de esta cadena porque $w \notin L(G)$.
- Tampoco es posible hallar una derivación de $w$ partiendo de $S$.
- El lenguaje generado por una gramática $G$ se puede definir (alternativamente) como el conjunto de cadenas que se pueden obtener en los nodos hojas de los AAS de $G$.
- Si consideramos la cadena $w = aaaab$, entonces sí que podremos encontrar un AAS. Esto significará que $w \in L(G)$.

# Derivaciones canónicas

## Consideremos la siguiente gramática:

$S \to SbS|ScS|a$

# Derivaciones canónicas

## Consideremos la siguiente gramática:

$S \to SbS|ScS|a$

## Consideremos una derivación de la cadena $w = acaba$:

$S \Rightarrow SbS \Rightarrow ScSbS \Rightarrow acSbS \Rightarrow acabS \Rightarrow acaba$

# Derivaciones canónicas

## Consideremos la siguiente gramática:

$S \to SbS|ScS|a$

## Consideremos una derivación de la cadena $w = acaba$:

$S \Rightarrow SbS \Rightarrow ScSbS \Rightarrow acSbS \Rightarrow acabS \Rightarrow acaba$

## En cada paso de derivación se toman dos decisiones:

# Derivaciones canónicas

## Consideremos la siguiente gramática:

$S \to SbS|ScS|a$

## Consideremos una derivación de la cadena $w = acaba$:

$S \Rightarrow SbS \Rightarrow ScSbS \Rightarrow acSbS \Rightarrow acabS \Rightarrow acaba$

## En cada paso de derivación se toman dos decisiones:

1. ¿Qué símbolo no terminal de los que aparecen en la forma sentencial se sustituye?

# Derivaciones canónicas

## Consideremos la siguiente gramática:

$S \to SbS|ScS|a$

## Consideremos una derivación de la cadena $w = acaba$:

$S \Rightarrow SbS \Rightarrow ScSbS \Rightarrow acSbS \Rightarrow acabS \Rightarrow acaba$

## En cada paso de derivación se toman dos decisiones:

1. ¿Qué símbolo no terminal de los que aparecen en la forma sentencial se sustituye?
2. ¿Qué producción para ese símbolo no terminal se utiliza para sustituirlo?

# Derivaciones canónicas

## Definición

Se denomina *derivación más a la izquierda* (o más a la derecha) a una derivación en la que, en cada forma sentencial, el símbolo no terminal que se sustituye (por la parte derecha de una producción para ese símbolo) es el que está más a la izquierda (o más a la derecha) en la forma sentencial.

# Derivaciones canónicas

## Definición

Se denomina *derivación más a la izquierda* (o más a la derecha) a una derivación en la que, en cada forma sentencial, el símbolo no terminal que se sustituye (por la parte derecha de una producción para ese símbolo) es el que está más a la izquierda (o más a la derecha) en la forma sentencial.

## Definición

En una *derivación canónica*, todas las derivaciones son más a la izquierda o más a la derecha.

# Ambigüedad

## Definición

Se dice que una gramática $G$ es ambigua si existe alguna cadena para la cual hay más de un árbol de análisis sintáctico distinto.

# Ambigüedad

## Definición

Se dice que una gramática $G$ es ambigua si existe alguna cadena para la cual hay más de un árbol de análisis sintáctico distinto.

- Una gramática $G$ es ambigua si alguna cadena de $L(G)$ tiene más de una derivación a izquierdas.
- Una gramática $G$ es ambigua si alguna cadena de $L(G)$ tiene más de una derivación a derechas.
- Una gramática $G$ para la que toda cadena tiene un único AAS se dice que es no ambigua.

# Ambigüedad

## Ejemplo

Sea la gramática siguiente: $S \rightarrow SbS|ScS|a$

# Ambigüedad

## Ejemplo

Sea la gramática siguiente: $S \rightarrow SbS|ScS|a$

## Veamos dos derivaciones a izquierdas para la cadena $abaca$:

- $S \Rightarrow_{mi} ScS \Rightarrow_{mi} SbScS \Rightarrow_{mi} abScS \Rightarrow_{mi} abacS \Rightarrow_{mi} abaca$
- $S \Rightarrow_{mi} SbS \Rightarrow_{mi} abS \Rightarrow_{mi} abScS \Rightarrow_{mi} abacS \Rightarrow_{mi} abaca$

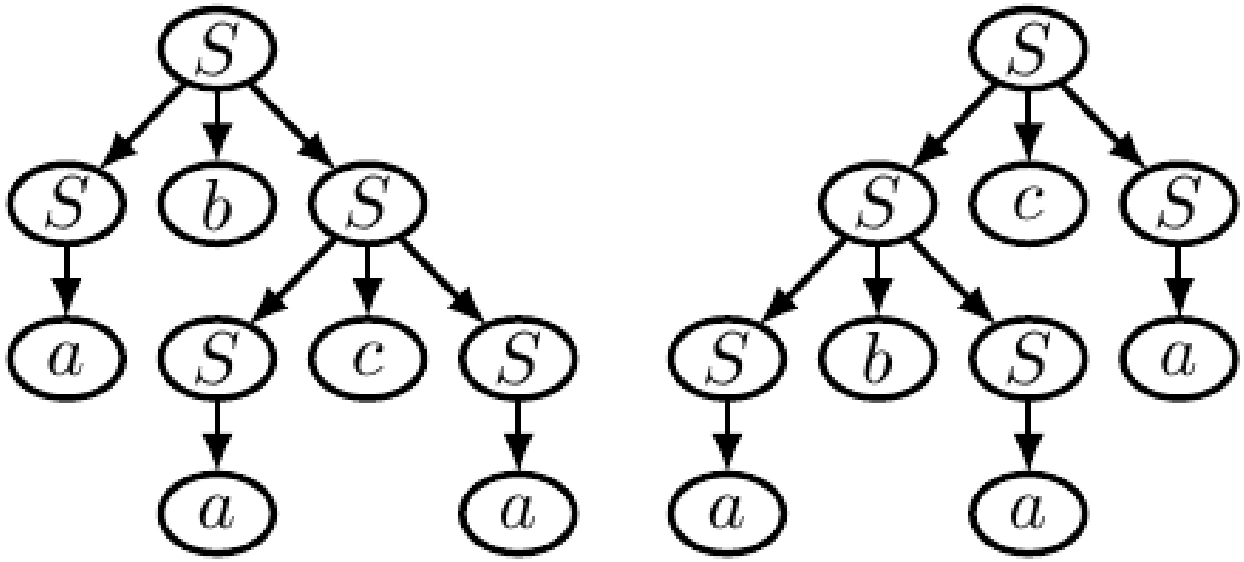
# Ambigüedad

## Ejemplo

Sea la gramática siguiente: $S \rightarrow SbS|ScS|a$

## Veamos dos derivaciones a izquierdas para la cadena $abaca$:

- $S \Rightarrow_{mi} ScS \Rightarrow_{mi} SbScS \Rightarrow_{mi} abScS \Rightarrow_{mi} abacS \Rightarrow_{mi} abaca$
- $S \Rightarrow_{mi} SbS \Rightarrow_{mi} abS \Rightarrow_{mi} abScS \Rightarrow_{mi} abacS \Rightarrow_{mi} abaca$



# Hay gramáticas que se pueden simplificar

## Ejemplo

Sea la gramática siguiente:

$S \to A$

$A \to B$

$B \to C$

$C \to D$

$D \to a|b$

# Hay gramáticas que se pueden simplificar

## Ejemplo

Sea la gramática siguiente:

$S \to A$

$A \to B$

$B \to C$

$C \to D$

$D \to a|b$

Es equivalente a:

$S \to a|b$

# Hay gramáticas que se pueden simplificar

## Ejemplo

Sea la gramática siguiente:

$S \rightarrow Aa|B|D$

$A \rightarrow aA|bA|B$

$B \rightarrow b$

$C \rightarrow abd$

# Hay gramáticas que se pueden simplificar

## Ejemplo

Sea la gramática siguiente:

$S \to Aa|B|D$

$A \to aA|bA|B$

$B \to b$

$C \to abd$

Es equivalente a:

$S \to Aa|b$

$A \to aA|bA|b$

# Símbolos útiles

## Definición

Un símbolo $X \in (\Sigma \cup V)$ se dice que es útil $\Leftrightarrow \exists$ una derivación de la forma: $S \Rightarrow^+ \alpha X \beta \Rightarrow^* w \in \Sigma^*$

# Símbolos útiles

## Definición

Un símbolo $X \in (\Sigma \cup V)$ se dice que es útil $\Leftrightarrow \exists$ una derivación de la forma: $S \Rightarrow^+ \alpha X \beta \Rightarrow^* w \in \Sigma^*$

Para que un símbolo $X$ sea útil ha de intervenir en la derivación de una cadena $u \in \Sigma^*$:

- Ha de ser "accesible" desde $S$: $S \Rightarrow^+ \alpha X \beta$
- Desde $X$ ha de poder derivarse una cadena de $\Sigma^*$: $X \Rightarrow^* u \in \Sigma^*$

# Símbolos útiles

## Definición

Un símbolo $X \in (\Sigma \cup V)$ se dice que es útil $\Leftrightarrow \exists$ una derivación de la forma: $S \Rightarrow^+ \alpha X \beta \Rightarrow^* w \in \Sigma^*$

Para que un símbolo $X$ sea útil ha de intervenir en la derivación de una cadena $u \in \Sigma^*$:

- Ha de ser "accesible" desde $S$: $S \Rightarrow^+ \alpha X \beta$
- Desde $X$ ha de poder derivarse una cadena de $\Sigma^*$: $X \Rightarrow^* u \in \Sigma^*$

Se dice que una producción es útil si lo son todos sus símbolos.

# Eliminación de símbolos y producciones inútiles

La eliminación de símbolos y producciones inútiles tiene dos etapas principales:

# Eliminación de símbolos y producciones inútiles

La eliminación de símbolos y producciones inútiles tiene dos etapas principales:

1. Eliminar las variables desde las que no se puede derivar una cadena de $\Sigma^*$ y las producciones en las que aparezcan dichas variables.

# Eliminación de símbolos y producciones inútiles

La eliminación de símbolos y producciones inútiles tiene dos etapas principales:

1. Eliminar las variables desde las que no se puede derivar una cadena de $\Sigma^*$ y las producciones en las que aparezcan dichas variables.
2. Eliminar aquellos símbolos que no se puedan derivar partiendo de $S$ y las producciones en las que aparecen esos símbolos.

# Eliminación de símbolos y producciones inútiles

## Etapa 1

$V' = \emptyset$;
**forall** (Producción de la forma $A \to w$) **do**
  $V' = V' \cup \{A\}$;
**while** ($V'$ cambie) **do**
  **forall** (Producción de la forma $B \to \alpha$) **do**
    **if** (todos los no-terminales de $\alpha$ pertenecen a $V'$) **then**
      $V' = V' \cup \{B\}$;

# Eliminación de símbolos y producciones inútiles

## Etapa 1

$V' = \emptyset$;
**forall** (Producción de la forma $A \to w$) **do**
  $V' = V' \cup \{A\}$;
**while** ($V'$ cambie) **do**
  **forall** (Producción de la forma $B \to \alpha$) **do**
    **if** (todos los no-terminales de $\alpha$ pertenecen a $V'$) **then**
      $V' = V' \cup \{B\}$;

- Eliminar todas las variables ($C \in V$) que estén en $V$ y no en $V'$.
- Eliminar todas las producciones donde aparezca una variable de las eliminadas en el paso anterior.

# Eliminación de símbolos y producciones inútiles

## Etapa 2

$J = \{S\}$;
$V' = \{S\}$;
$\Sigma' = \emptyset$;
**while** $(J \neq \emptyset)$ **do**
  Extraer un no terminal $A$ de $J$: $J = J - \{A\}$;
  **forall** (Producción de la forma $A \rightarrow \alpha$) **do**
    Poner todos los terminales de $\alpha$ en $\Sigma'$;
    **forall** (no-terminal $B$ en $\alpha$) **do**
      **if** $(B \notin V')$ **then**
        $J = J \cup \{B\}$;
        $V' = V' \cup \{B\}$;

# Eliminación de símbolos y producciones inútiles

## Etapa 2

$J = \{S\}$;
$V' = \{S\}$;
$\Sigma' = \emptyset$;
**while** $(J \neq \emptyset)$ **do**
  Extraer un no terminal $A$ de $J$: $J = J - \{A\}$;
  **forall** (Producción de la forma $A \rightarrow \alpha$) **do**
    Poner todos los terminales de $\alpha$ en $\Sigma'$;
    **forall** (no-terminal $B$ en $\alpha$) **do**
      **if** $(B \notin V')$ **then**
        $J = J \cup \{B\}$;
        $V' = V' \cup \{B\}$;

- Eliminar todos los no terminales de $V$ que no estén en $V'$ y todos los terminales que no estén en $\Sigma^*$.
- Eliminar todas las producciones donde aparezca un símbolo o variable de los eliminados.

# Eliminación de símbolos y producciones inútiles

## Ejemplo 1

$S \to Aa|B|D$
$A \to aA|bA|B$
$B \to b$
$C \to abd$

# Eliminación de símbolos y producciones inútiles

## Ejemplo 1

$S \to Aa|B|D$
$A \to aA|bA|B$
$B \to b$
$C \to abd$

## Después de la etapa 1

$S \to Aa|B$
$A \to aA|bA|B$
$B \to b$
$C \to abd$

# Eliminación de símbolos y producciones inútiles

## Ejemplo 1

$S \to Aa|B|D$
$A \to aA|bA|B$
$B \to b$
$C \to abd$

## Después de la etapa 1

$S \to Aa|B$
$A \to aA|bA|B$
$B \to b$
$C \to abd$

## Después de la etapa 2

$S \to Aa|B$
$A \to aA|bA|B$
$B \to b$

# Eliminación de símbolos y producciones inútiles

## Ejemplo 2

$S \to AB|a$

$A \to a$

# Eliminación de símbolos y producciones inútiles

## Ejemplo 2

$S \to AB|a$

$A \to a$

## Después de la etapa 1

$S \to a$

$A \to a$

# Eliminación de símbolos y producciones inútiles

## Ejemplo 2

$S \to AB|a$

$A \to a$

## Después de la etapa 1

$S \to a$

$A \to a$

## Después de la etapa 2

$S \to a$

# Eliminación de símbolos y producciones inútiles

## Ejemplo 2

$S \to AB|a$
$A \to a$

## Después de la etapa 1

$S \to a$
$A \to a$

## Después de la etapa 2

$S \to a$

¿Obtendríamos el mismo resultado si aplicáramos las etapas en orden inverso?

# Producciones vacías

## Definición

Una producción gramatical es vacía si es de la forma: $A \to \varepsilon$.

# Producciones vacías

## Definición

Una producción gramatical es vacía si es de la forma: $A \to \varepsilon$.

- Modificaremos la gramática para eliminar este tipo de producciones, sin cambiar el lenguaje generado.
- Si $\varepsilon \in L(G)$ no podremos eliminar todas las producciones vacías.
- En este caso dejaremos como única producción vacía: $S \to \varepsilon$.

# Cálculo del conjunto de variables anulables

$H = \emptyset$;
**forall** (Producción de la forma $A \to \varepsilon$)
**do**
  $H = H \cup \{A\}$;
**while** (H cambie)
**do**
  **forall** (Producción $B \to A_1 A_2 ... A_n$ donde $A_i \in H, \forall i = 1, 2, ...n$)
  **do**
    $H = H \cup \{B\}$;

# Eliminación de producciones vacías

Eliminar todas las producciones vacías ($A \to \varepsilon$);
**forall** (Producción de la forma $A \to X_1X_2...X_n, X_i \in (\Sigma \cup V)$)
**do**
  Eliminar la producción $A \to X_1X_2...X_n$;
  Añadir todas las producciones $A \to Y_1Y_2...Y_n$, donde:
    $Y_i = X_i$ si $X_i$ no es anulable;
    $(Y_i = X_i) \vee (Y_i = \varepsilon)$ si $X_i$ es anulable;
    $Y_i$ no es $\varepsilon$ $\forall i$ (no se introducen producciones vacías);

# Eliminación de producciones vacías

## Ejemplo 1

$S \to aA$

$A \to aA|\varepsilon$

# Eliminación de producciones vacías

## Ejemplo 1

$S \to aA$
$A \to aA|\varepsilon$

## Resultado

$S \to aA|a$
$A \to aA|a$

# Eliminación de producciones vacías

## Ejemplo 1

$S \to aA$
$A \to aA|\varepsilon$

## Resultado

$S \to aA|a$
$A \to aA|a$

Se ha eliminado $A \to \varepsilon$

# Eliminación de producciones vacías

## Ejemplo 2

$S \to ABb|ABC$
$A \to aA|\varepsilon$
$B \to bB|\varepsilon$
$C \to abC|AB$

# Eliminación de producciones vacías

## Ejemplo 2

$S \to ABb|ABC$
$A \to aA|\varepsilon$
$B \to bB|\varepsilon$
$C \to abC|AB$

## Resultado

$S \to \varepsilon|ABb|Ab|Bb|b$
$S \to ABC|AB|AC|BC|A|B|C$
$A \to aA|a$
$B \to bB|b$
$C \to abC|ab|AB|A|B$

# Eliminación de producciones vacías

## Ejemplo 2

$S \to ABb|ABC$
$A \to aA|\varepsilon$
$B \to bB|\varepsilon$
$C \to abC|AB$

## Resultado

$S \to \varepsilon|ABb|Ab|Bb|b$
$S \to ABC|AB|AC|BC|A|B|C$
$A \to aA|a$
$B \to bB|b$
$C \to abC|ab|AB|A|B$

Se ha eliminado $A \to \varepsilon$
Se ha eliminado $B \to \varepsilon$

# Producciones unitarias

**Definición**

Una producción es unitaria si tiene la forma $A \to B$ (con $A, B \in V$).

- Este tipo de producciones hacen la gramática innecesariamente compleja.
- Las producciones unitarias, simplemente renombran un símbolo no terminal.

# Eliminación de producciones unitarias

$H = \emptyset$; //$H \subseteq (V \times V)$ de modo que $(A, B) \in H \Leftrightarrow A \Rightarrow^* B$
**forall** (Producción de la forma $A \to B$) **do**
  $H = H \cup \{(A, B)\}$;
**while** (H cambie) **do**
  **forall** (par de parejas de la forma $(A, B)(B, C)$) **do**
    **if** $((A, C) \notin H)$ **then**
      $H = H \cup \{(A, C)\}$;
Eliminar todas las producciones unitarias;
**forall** (Producción $B \to \alpha$) **do**
  **forall** (pareja $(A, B) \in H$) **do**
    Añadir a la gramática una producción $A \to \alpha$;

# Eliminación de producciones unitarias

## Ejemplo

$S \to A|Aa$
$A \to B$
$B \to C|b$
$C \to D|ab$
$D \to b$

# Eliminación de producciones unitarias

## Ejemplo

$S \to A|Aa$
$A \to B$
$B \to C|b$
$C \to D|ab$
$D \to b$

## Resultado

$S \to Aa|ab|b$
$A \to ab|b$
$B \to ab|b$
$C \to ab|b$
$D \to b$

# Recursividad

## Definición

Una CFG $G$ es recursiva si tiene derivaciones de la forma: $A \Rightarrow^+ \alpha A \beta$

# Recursividad

## Definición

Una CFG $G$ es recursiva si tiene derivaciones de la forma: $A \Rightarrow^+ \alpha A \beta$

## Definición

Si $\alpha = \varepsilon$, la gramática tendrá derivaciones de la forma: $A \Rightarrow^+ A\beta$
En este caso se dice que $G$ es *recursiva por la izquierda*.

# Recursividad

## Definición

Una CFG $G$ es recursiva si tiene derivaciones de la forma: $A \Rightarrow^+ \alpha A \beta$

## Definición

Si $\alpha = \varepsilon$, la gramática tendrá derivaciones de la forma: $A \Rightarrow^+ A\beta$
En este caso se dice que $G$ es *recursiva por la izquierda*.

## Definición

Si $\beta = \varepsilon$, la gramática tendrá derivaciones de la forma: $A \Rightarrow^+ \alpha A$
En este caso se dice que $G$ es *recursiva por la derecha*.

# Forma normal (FN) de Chomsky

## Definición

Una CFG $G \equiv (V, \Sigma, S, P)$ está escrita en forma normal de Chomsky si:

- $V$ sólo contiene símbolos útiles.
- Todas las producciones de $G$ tienen una de las formas:
  $A \to BC$
  $A \to a$
  $S \to \varepsilon$

# Forma normal (FN) de Chomsky

## Definición

Una CFG $G \equiv (V, \Sigma, S, P)$ está escrita en forma normal de Chomsky si:

- $V$ sólo contiene símbolos útiles.
- Todas las producciones de $G$ tienen una de las formas:
  $A \to BC$
  $A \to a$
  $S \to \varepsilon$

- Para una CFG en FN de Chomsky, todo árbol de análisis sintáctico es un árbol binario.
- Toda CFG puede reescribirse en FN de Chomsky.

# Forma normal (FN) de Chomsky

Eliminar todas las producciones vacías;
Eliminar todas las producciones unitarias;
Eliminar todos los símbolos y producciones inútiles;
**forall** ($A \to X_1X_2...X_n$ (con $n \geq 2$, $X_i \in (\Sigma \cup V)$) **do**
  **forall** ($X_i$) **do**
    **if** ($X_1 = a \in \Sigma$) **then**
      Se añade la producción $C_a \to a$;
      Se cambia $X_i$ por $C_a$ en $A \to X_1X_2...X_n$;
**forall** ($A \to B_1B_2...B_m$ (con $m \geq 3$, $B_i \in V$) **do**
  Se añaden $m-2$ símbolos no-terminales $D_1D_2...D_{m-2}$;
  Se sustituye la producción $A \to B_1B_2...B_m$ por las producciones:
    $A \to B_1D_1$
    $D_1 \to B_2D_2$
    ...
    $D_{m-2} \to B_{m-1}B_m$

# Forma normal (FN) de Chomsky

## Ejemplo

$S \to bA|aB$
$A \to bAA|aS|a$
$B \to aBB|bS|b$

# Forma normal (FN) de Chomsky

## Ejemplo

$S \to bA|aB$
$A \to bAA|aS|a$
$B \to aBB|bS|b$

## Resultado

$S \to C_bA|C_aB$
$A \to C_bD_1|C_aS|a$
$B \to C_aD_2|C_bS|b$
$D_1 \to AA$
$D_2 \to BB$
$Ca \to a$
$Cb \to b$

# Forma normal (FN) de Chomsky y AAS

Sea $G \equiv (V, \Sigma, S, P)$ una CFG escrita en FN de Chomsky, y considerando un AAS para una cadena $w \in L(G)$:

- El árbol de análisis sintáctico de cualquier cadena es un árbol binario si $G$ está en FN de Chomsky.
- Supongamos que el camino más largo desde la raíz hasta las hojas en uno de estos AAS tiene $m + 2$ nodos (y por tanto $m + 1$ arcos).
- Los nodos hoja en estos AAS son "hijos únicos" de sus nodos padre.
- En el nivel $m$ hay a lo sumo $2^m$ nodos padre.
- Conclusión:
  - Si el camino más largo desde la raíz hasta las hojas en un AAS consta de $m + 2$ nodos, entonces $2^m$ es la longitud más larga posible de la cadena derivada $(w)$.
  - Si la cadena $w \in L(G)$ tiene $|w| > 2^m$ entonces el camino más largo "raíz-hojas" ha de tener más de $m + 2$ nodos.

# Lema del Bombeo

## Teorema

Sea $L$ un lenguaje independiente del contexto $(L = L(G), G \equiv (V, \Sigma, S, P))$ que no contiene a $\varepsilon$. Existe un entero $k \in \mathbb{N}$ para el cual si $z \in L$ y $|z| > k$ entonces:

# Lema del Bombeo

## Teorema

Sea $L$ un lenguaje independiente del contexto $(L = L(G), G \equiv (V, \Sigma, S, P))$ que no contiene a $\varepsilon$. Existe un entero $k \in \mathbb{N}$ para el cual si $z \in L$ y $|z| > k$ entonces:

1. $z = uvwxy \quad u, v, w, x, y \in \Sigma^*$

# Lema del Bombeo

## Teorema

Sea $L$ un lenguaje independiente del contexto $(L = L(G), G \equiv (V, \Sigma, S, P))$ que no contiene a $\varepsilon$. Existe un entero $k \in \mathbb{N}$ para el cual si $z \in L$ y $|z| > k$ entonces:

1. $z = uvwxy \quad u, v, w, x, y \in \Sigma^*$
2. $|vwx| \leq k$

# Lema del Bombeo

## Teorema

Sea $L$ un lenguaje independiente del contexto $(L = L(G), G \equiv (V, \Sigma, S, P))$ que no contiene a $\varepsilon$. Existe un entero $k \in \mathbb{N}$ para el cual si $z \in L$ y $|z| > k$ entonces:

1. $z = uvwxy \quad u, v, w, x, y \in \Sigma^*$
2. $|vwx| \leq k$
3. $uv^i wx^i y \in L \quad \forall i \geq 0$

# Lema del Bombeo

## Teorema

Sea $L$ un lenguaje independiente del contexto $(L = L(G), G \equiv (V, \Sigma, S, P))$ que no contiene a $\varepsilon$. Existe un entero $k \in \mathbb{N}$ para el cual si $z \in L$ y $|z| > k$ entonces:

1. $z = uvwxy \quad u, v, w, x, y \in \Sigma^*$
2. $|vwx| \leq k$
3. $uv^i wx^i y \in L \quad \forall i \geq 0$
4. $|vx| \geq 1$ (una de las dos, $v$ o $x$ no puede ser $\varepsilon$)

# Lema del Bombeo

## Demostración

Sea $G \equiv (V, \Sigma, S, P)$ una CFG en FN de Chomsky, tal que $L = L(G)$.
Sea $n = |V|$ y $k = 2^n$
Sea $z \in L$ con $|z| > k = 2^n$

- Puesto que $|z| > 2^n$, el camino más largo "raíz-hojas" en un AAS para $z$ ha de tener más de $n + 2$ nodos.
- Consideremos los $n + 2$ últimos nodos de ese camino (los más cercanos a las hojas). De esos $n + 2$ nodos, el último corresponderá a un símbolo terminal, y los $n + 1$ nodos restantes estarán etiquetados con símbolos no terminales.
- Puesto que $|V| = n$, algún no terminal se repetirá en ese camino.

# Lema del Bombeo

## Demostración

Sea $G \equiv (V, \Sigma, S, P)$ una CFG en FN de Chomsky, tal que $L = L(G)$.
Sea $n = |V|$ y $k = 2^n$
Sea $z \in L$ con $|z| > k = 2^n$

- Supongamos que $A \in V$ está repetido en el camino. Entonces:
  $S \Rightarrow^* uAy \Rightarrow^* uvAxy \Rightarrow^* uvwxy = z$
  para algunas subcadenas $u, v, w, x, y \in \Sigma^*$
- Puesto que hemos considerado sólo los $n + 2$ últimos nodos del camino de derivación de $z$, entonces el camino $A \Rightarrow^* vwx$ tendrá a lo sumo $n + 2$ nodos y por lo tanto $|vwx| < 2^n = k$.
- Dado que $A \Rightarrow^* vAx$ entonces $A \Rightarrow^* v^i Ax^i \ \forall i \geq 0$ por lo que $S \Rightarrow^* uv^i wx^i y \ \forall i \geq 0$

# Lema del Bombeo

## Demostración

Sea $G \equiv (V, \Sigma, S, P)$ una CFG en FN de Chomsky, tal que $L = L(G)$.
Sea $n = |V|$ y $k = 2^n$
Sea $z \in L$ con $|z| > k = 2^n$

- Finalmente, puesto que $A \Rightarrow^* vAx$ y $G$ están en FN de Chomsky, tendremos que la derivación tiene la forma:
  $A \Rightarrow BC \Rightarrow^* vAx \Rightarrow^* vwx$
- Puesto que no hay producciones vacías y $G$ está en FN de Chomsky, hay dos posibilidades:
  1. $B \Rightarrow^* v$
     $C \Rightarrow^* Ax$
     En cuyo caso $|v| \geq 1$, y $v \neq \varepsilon$
  2. $B \Rightarrow^* vA$
     $C \Rightarrow^* x$
     En cuyo caso $|x| \geq 1$, y $x \neq \varepsilon$

# Lema del Bombeo

## Ejemplo: $L = \{a^i b^j \mid j = i^2, i, j \geq 1\}$

- Sea $k$ la constante del lema.
- Consideremos $z = a^k b^{k^2} \in L, |z| > k$
- Debería poder descomponerse $z = uvwxy$ cumpliéndose $uv^i wx^i y \in L,\ \forall i \geq 0$ con $|vx| \geq 1$ y $|vwx| \leq k$.
- Si $v = a^r b^s$ entonces $v^i = (a^r b^s)^i$ y por lo tanto $uv^i wx^i y \notin L$ puesto que la cadena tendría "bes" antes de "aes".
- Lo mismo ocurre si $x = a^r b^s$
- Así pues, las posibilidades restantes son:
  1. $v = a^r \quad x = a^s$
  2. $v = b^r \quad x = b^s$
  3. $v = a^r \quad x = b^s$
- $r$ o $s$ ha de ser mayor que $1$ puesto que $|vx| \geq 1$.

# Lema del Bombeo

## Ejemplo: $L = \{a^i b^j \mid j = i^2, i, j \geq 1\}$

- $z = uvwxy = a^k b^{k^2} \in L, |z| > k$
  1. $v = a^r \quad x = a^s$
     $uv^2wx^2y = a^{k+r+s}b^{k^2} \notin L$ (al bombear sólo se añaden símbolos "a")
  2. $v = b^r \quad x = b^s$
     $uv^2wx^2y = a^k b^{k^2+r+s} \in L$ (al bombear sólo se añaden símbolos "b")
  3. $v = a^r \quad x = b^s$
     $uv^iwx^iy = a^{k-r+ir}b^{k^2-s+is} = a^{k+(i-1)r}b^{k^2+(i-1)s}$
     $uv^iwx^iy \notin L \ \forall i$ (sí pertenecen para algunos valores de $i$)

# Lema del Bombeo

## Ejemplo: $L = \{a^i b^i c^i \mid i \geq 1\}$

- Sea $n$ la constante del lema.
- Consideremos $z = a^n b^n c^n \in L, |z| = 3n > n$
- Supongamos que $z = uvwxy$ con $|vwx| \leq n$
- Puesto que $|vwx| \leq n$ la cadena $vx$ no puede contener simultáneamente símbolos $a$ y $c$ porque entre la última $a$ y la primera $c$ hay $n$ símbolos.

# Lema del Bombeo

## Ejemplo: $L = \{a^i b^i c^i \mid i \geq 1\}$

- En estas condiciones ($z = a^n b^n c^n = uvwxy$) se pueden dar los siguientes casos:
  1. Si $vx$ contiene exclusivamente símbolos $a$. En este caso, $uv^0wx^0y = uwy \notin L$ porque contiene $n$ símbolos $b$, $n$ símbolos $c$ y menos de $n$ símbolos $a$.
  2. Si $vx$ contiene solamente símbolos $b$ o sólo símbolos $c$ se llega a la misma conclusión con un razonamiento similar.
  3. Si $v$ contiene símbolos $a$ y $x$ contiene símbolos $b$, entonces $uv^0wx^0y = uwy$ tendrá más símbolos $c$ que "a's" y "b's".
  4. Si $v$ contiene "b's" y $x$ contiene "c's" entonces $uwy$ tendrá más símbolos $a$ que "b's" y "c's".
- Puesto que $L$ no cumple el LB, concluimos que no es un lenguaje independiente del contexto.

# Lenguaje vacío

**Teorema**

Existe un algoritmo para determinar si un lenguaje independiente del contexto es vacío.

# Lenguaje vacío

## Teorema

Existe un algoritmo para determinar si un lenguaje independiente del contexto es vacío.

## Demostración

- En la primera parte del algoritmo de eliminación de símbolos y variables inútiles se determinan los símbolos de $V$ que pueden derivar una frase $w \in \Sigma^*$
- $L(G) \neq \emptyset \Leftrightarrow S$ no es eliminada en ese proceso.

# Lenguaje finito o infinito

**Teorema**

Existe un algoritmo para determinar si un lenguaje independiente del contexto es finito.

# Lenguaje finito o infinito

## Teorema

Existe un algoritmo para determinar si un lenguaje independiente del contexto es finito.

## Demostración

Para determinar si $L(G)$ es finito o infinito:

- Se escribe $G$ en forma normal de Chomsky, sin símbolos ni producciones inútiles.
- En estas condiciones, todas las producciones tendrán la forma:
  $A \to BC$
  $A \to a$
- Se construye un grafo dirigido con los símbolos $A \in V$ como nodos.
- Para cada producción $A \to BC$ se añaden dos arcos: $A \to B$ y $A \to C$
- $L(G)$ es finito $\Leftrightarrow$ el grafo no tiene ciclos dirigidos.

# Lenguaje finito o infinito

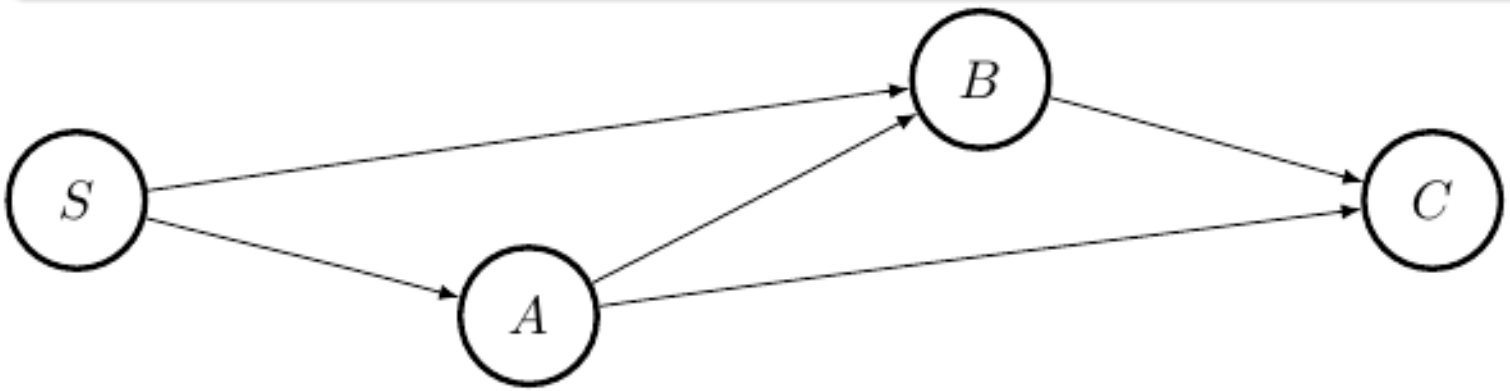
## Ejemplo

$S \to AB$
$A \to BC|a$
$B \to CC|b$
$C \to a$

# Lenguaje finito o infinito

## Ejemplo

$S \to AB$
$A \to BC|a$
$B \to CC|b$
$C \to a$



# Lenguaje finito o infinito

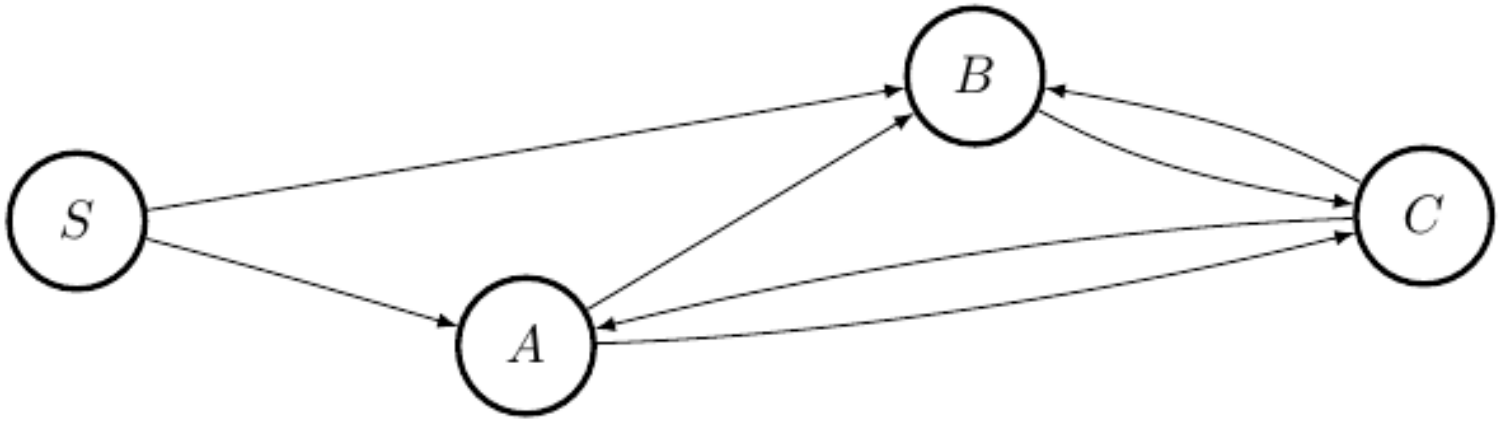
## Ejemplo

$S \to AB$
$A \to BC|a$
$B \to CC|b$
$C \to a|AB$

# Lenguaje finito o infinito

## Ejemplo

$S \to AB$
$A \to BC|a$
$B \to CC|b$
$C \to a|AB$



# Análisis sintáctico

## Definición

Sea $G \equiv (V, \Sigma, S, P)$ una CFG.
Sea $w \in \Sigma^*$.
El problema del análisis sintáctico es: ¿$w \in L(G)$?

# Análisis sintáctico

## Definición

Sea $G \equiv (V, \Sigma, S, P)$ una CFG.
Sea $w \in \Sigma^*$.
El problema del análisis sintáctico es: ¿$w \in L(G)$?

## Lema

Sea $G \equiv (V, \Sigma, S, P)$ una CFG en FN de Chomsky.
Sea $x \in \Sigma^*$:
$\forall A \in V$ y $\forall w$ subcadena de $x$ se puede determinar si $A \Rightarrow^* w$.

# Análisis sintáctico

## Definición

Sea $G \equiv (V, \Sigma, S, P)$ una CFG.
Sea $w \in \Sigma^*$.
El problema del análisis sintáctico es: ¿$w \in L(G)$?

## Lema

Sea $G \equiv (V, \Sigma, S, P)$ una CFG en FN de Chomsky.
Sea $x \in \Sigma^*$:
$\forall A \in V$ y $\forall w$ subcadena de $x$ se puede determinar si $A \Rightarrow^* w$.

- Sea $n = |x|$.
- Sea $w_{ij}$ la subcadena de $x$ que comienza en la posición $i$ y tiene longitud $j$.
- Demostraremos que el lema se cumple $\forall w_{ij}$ por inducción sobre la longitud de la cadena.

# Análisis sintáctico

## Demostración

- Si $j = 1$ entonces $|w_{ij}| = 1$, $w_{ij} \in \Sigma$ (es un terminal) y tenemos que: $A \Rightarrow^* w_{ij} \Leftrightarrow (A \to w_{ij}) \in P$
- Esto se puede determinar puesto que $P$ (las producciones de la gramática) es un conjunto finito.

# Análisis sintáctico

## Demostración

- Si $j = 1$ entonces $|w_{ij}| = 1$, $w_{ij} \in \Sigma$ (es un terminal) y tenemos que: $A \Rightarrow^* w_{ij} \Leftrightarrow (A \to w_{ij}) \in P$
- Esto se puede determinar puesto que $P$ (las producciones de la gramática) es un conjunto finito.
- Supongamos ahora que el lema se cumple $\forall w_{ih}$ tal que $|w_{ih}| = h < j$ (siendo $j > 1$).
- Considerando la derivación de una cadena $w_{ij}$ de longitud $j$:
  - $A \Rightarrow^* w_{ij} \Leftrightarrow A \to BC$ tal que $B \Rightarrow^* w_{ik}$ y $C \Rightarrow^* w_{i+k,j-k}$ para algún $1 \leq k \leq j-1$.
- Puesto que $w_{ik}$ y $w_{i+k,j-k}$ tienen longitud menor que $j$, entonces por hipótesis se puede determinar si $B \Rightarrow^* w_{ik}$ y también si $C \Rightarrow^* w_{i+k,j-k}$.

# Análisis sintáctico

## Demostración

- Así podemos determinar si $A \Rightarrow^* w_{ij}$ con $1 \leq i \leq n$, $1 \leq j \leq n - i + 1$.

# Análisis sintáctico

## Demostración

- Así podemos determinar si $A \Rightarrow^* w_{ij}$ con $1 \leq i \leq n$, $1 \leq j \leq n - i + 1$.
- Particularizando este lema para $A = S \in V$ y considerando la cadena $w_{1n} = x$.

# Análisis sintáctico

## Demostración

- Así podemos determinar si $A \Rightarrow^* w_{ij}$ con $1 \leq i \leq n$, $1 \leq j \leq n - i + 1$.
- Particularizando este lema para $A = S \in V$ y considerando la cadena $w_{1n} = x$.
- El lema indica que es posible determinar si $S \Rightarrow^* w_{1n} = x$.

# Análisis sintáctico

## Demostración

- Así podemos determinar si $A \Rightarrow^* w_{ij}$ con $1 \leq i \leq n$, $1 \leq j \leq n - i + 1$.
- Particularizando este lema para $A = S \in V$ y considerando la cadena $w_{1n} = x$.
- El lema indica que es posible determinar si $S \Rightarrow^* w_{1n} = x$.
- Dicho de otro modo, indica que es posible determinar si $x \in L(G)$, o lo que es lo mismo, que es posible resolver el problema del análisis sintáctico.

# Algoritmo de Cocke, Younger y Kasami (CYK)

El algoritmo calcula los conjuntos $V_{ij} = \{A \in V \mid A \Rightarrow^* w_{ij}\}$
**Entrada:** una cadena $w = a_1 a_2 ... a_n$ y una CFG escrita en FNC
**Salida:** el algoritmo determina si $w \in L(G) : w \in L(G) \Leftrightarrow S \in V_{1n}$

**for** (i := 1 to n) **do**
  $V_{i1} := \{A \mid (A \to a) \in P$ y el iésimo símbolo de $w$ es $a\}$
**for** (j := 2 to n) **do**
  **for** (i := 1 to n - j + 1) **do**
    $V_{ij} := \emptyset$;
    **for** (k := 1 to j - 1) **do**
      $V_{ij} := V_{ij} \cup \{A \mid (A \to BC) \in P, B \in V_{ik}, C \in V_{i+k,j-k}\}$;

# Algoritmo de Cocke, Younger y Kasami (CYK)

## Ejemplo

$S \to AB|BC$
$A \to BA|a$
$B \to CC|b$
$C \to AB|a$

# Algoritmo de Cocke, Younger y Kasami (CYK)

## Ejemplo

$S \to AB|BC$
$A \to BA|a$
$B \to CC|b$
$C \to AB|a$

Sea la cadena $w = baaba$

# Algoritmo de Cocke, Younger y Kasami (CYK)

## Ejemplo

$S \to AB|BC$
$A \to BA|a$
$B \to CC|b$
$C \to AB|a$

Sea la cadena $w = baaba$

| | b | a | a | b | a |
|---|---|---|---|---|---|
| 1 | B | A, C | A, C | B | A, C |
| 2 | S, A | B | S, C | S, A | |
| 3 | $\emptyset$ | B | B | | |
| 4 | $\emptyset$ | S, A, C | | | |
| 5 | S, A, C | | | | |

# Algoritmo de Cocke, Younger y Kasami (CYK)

## Ejemplo

$S \to AB|BC$
$A \to BA|a$
$B \to CC|b$
$C \to AB|a$

Sea la cadena $w = baaba$

| | b | a | a | b | a |
|---|---|---|---|---|---|
| 1 | B | A, C | A, C | B | A, C |
| 2 | S, A | B | S, C | S, A | |
| 3 | $\emptyset$ | B | B | | |
| 4 | $\emptyset$ | S, A, C | | | |
| 5 | S, A, C | | | | |

$w = baaba \in L(G)$

# Algoritmo de Cocke, Younger y Kasami (CYK)

## Ejemplo

$S \to AB|BC$
$A \to BA|a$
$B \to CC|b$
$C \to AB|a$

# Algoritmo de Cocke, Younger y Kasami (CYK)

## Ejemplo

$S \to AB|BC$
$A \to BA|a$
$B \to CC|b$
$C \to AB|a$

Sea la cadena $w = bbab$

# Algoritmo de Cocke, Younger y Kasami (CYK)

## Ejemplo

$S \to AB|BC$
$A \to BA|a$
$B \to CC|b$
$C \to AB|a$

Sea la cadena $w = bbab$

| | b | b | a | b |
|---|---|---|---|---|
| 1 | B | B | A, C | B |
| 2 | $\emptyset$ | A, S | S, C | |
| 3 | A | S, C | | |
| 4 | S, C | | | |

# Algoritmo de Cocke, Younger y Kasami (CYK)

## Ejemplo

$S \to AB|BC$
$A \to BA|a$
$B \to CC|b$
$C \to AB|a$

Sea la cadena $w = bbab$

| | b | b | a | b |
|---|---|---|---|---|
| 1 | B | B | A, C | B |
| 2 | $\emptyset$ | A, S | S, C | |
| 3 | A | S, C | | |
| 4 | S, C | | | |

$w = bbab \in L(G)$

# Unión

## Teorema

Si $L_1$ y $L_2$ son lenguajes independientes del contexto, entonces $L_1 \cup L_2$ también es un lenguaje independiente del contexto.

# Unión

## Teorema

Si $L_1$ y $L_2$ son lenguajes independientes del contexto, entonces $L_1 \cup L_2$ también es un lenguaje independiente del contexto.

## Demostración

- Si $L_1$ y $L_2$ son CFL's, entonces existen las gramáticas que los generan $L_1 = L(G_1)$ y $L_2 = L(G_2)$:
  $G_1 \equiv (V_1, \Sigma_1, S_1, P_1)$
  $G_2 \equiv (V_2, \Sigma_2, S_2, P_2)$
- Asumiremos (sin pérdida de generalidad) que $V_1 \cap V_2 = \emptyset$
- Construiremos otra gramática $G \equiv (V, \Sigma, S, P)$ tal que $L(G) = L_1 \cup L_2$:
  - $V = V_1 \cup V_2 \cup \{S\}$ siendo $S$ un nuevo símbolo $(S \notin V_1 \cup V_2)$
  - $\Sigma = \Sigma_1 \cup \Sigma_2$
  - $P = P_1 \cup P_2 \cup \{S \to S_1 | S_2\}$

# Unión

## Demostración

- Si $w \in L(G_1)$, entonces $\exists S_1 \Rightarrow^* w$ pero también: $S \Rightarrow S_1 \Rightarrow^* w$ de modo que $w \in L(G)$
- Si $w \in L(G_2)$, entonces $\exists S_2 \Rightarrow^* w$ pero también: $S \Rightarrow S_2 \Rightarrow^* w$ de modo que $w \in L(G)$
- Por lo tanto $L(G_1) \cup L(G_2) \subseteq L(G)$
- Por otra parte, si $w \in L(G)$, entonces $S \Rightarrow^* w$:
  $S \Rightarrow S_1 \Rightarrow^* w$ en cuyo caso $w \in L(G_1)$, o bien,
  $S \Rightarrow S_2 \Rightarrow^* w$ en cuyo caso $w \in L(G_2)$
- Téngase en cuenta que, puesto que $V_1 \cap V_2 = \emptyset$ sólo se pueden usar producciones de $P_i$ al derivar $S_i \Rightarrow^* w$
- Así pues, concluimos que $L(G) \subseteq L(G_1) \cup L(G_2)$ y finalmente tenemos que $L(G) = L(G_1) \cup L(G_2)$ y, por tanto, $L(G_1) \cup L(G_2)$ es independiente del contexto.

# Concatenación

## Teorema

Si $L_1$ y $L_2$ son lenguajes independientes del contexto, entonces $L_1L_2$ también es un lenguaje independiente del contexto.

# Concatenación

## Teorema

Si $L_1$ y $L_2$ son lenguajes independientes del contexto, entonces $L_1 L_2$ también es un lenguaje independiente del contexto.

## Demostración

- Si $L_1$ y $L_2$ son CFL's, entonces existen las gramáticas que los generan $L_1 = L(G_1)$ y $L_2 = L(G_2)$:
  $G_1 \equiv (V_1, \Sigma_1, S_1, P_1)$
  $G_2 \equiv (V_2, \Sigma_2, S_2, P_2)$
- Asumiremos (sin pérdida de generalidad) que $V_1 \cap V_2 = \emptyset$
- Construiremos otra gramática $G \equiv (V, \Sigma, S, P)$ tal que $L(G) = L_1 \cup L_2$:
  - $V = V_1 \cup V_2 \cup \{S\}$ siendo $S$ un nuevo símbolo $(S \notin V_1 \cup V_2)$
  - $\Sigma = \Sigma_1 \cup \Sigma_2$
  - $P = P_1 \cup P_2 \cup \{S \to S_1 S_2\}$
- Ejercicio: demostrar que $L(G) = L(G_1)L(G_2)$

# Cierre de Kleene

## Teorema

Si $L$ es un lenguaje independiente del contexto, entonces $L^*$ también es un lenguaje independiente del contexto.

# Cierre de Kleene

## Teorema

Si $L$ es un lenguaje independiente del contexto, entonces $L^*$ también es un lenguaje independiente del contexto.

## Demostración

- Si $L$ es un CFL, entonces existe una gramática que lo genera $(L = L(G))$:
  $G \equiv (V, \Sigma, S, P)$
- Construiremos otra gramática $G' \equiv (V', \Sigma', S', P')$ tal que $L(G') = L^*$:
  - $V' = V \cup \{S'\}$ siendo $S'$ un nuevo símbolo $(S' \notin V)$
  - $\Sigma' = \Sigma$
  - $P' = P \cup \{S' \to SS' | \varepsilon\}$
- Ejercicio: demostrar que $L(G') = L^*$

# Intersección

## Teorema

Los lenguajes independientes del contexto son cerrados respecto a unión, concatenación y cierre de Kleene, pero no lo son respecto a la intersección.

# Intersección

## Teorema

Los lenguajes independientes del contexto son cerrados respecto a unión, concatenación y cierre de Kleene, pero no lo son respecto a la intersección.

## Demostración a través de un contraejemplo

- Sea $L_1 = \{a^i b^j c^j \mid i, j \geq 1\}$, la siguiente gramática genera $L_1$:
  $S \to AC$
  $A \to aA|a$
  $C \to bCc|bc$
- Sea $L_2 = \{a^i b^i c^j \mid i, j \geq 1\}$, la siguiente gramática genera $L_2$:
  $S \to AC$
  $A \to aAb|ab$
  $C \to cC|c$
- Sin embargo, $L = \{a^i b^i c^i \mid i \geq 1\} = L_1 \cap L_2$ sabemos (lema del bombeo) que no es un lenguaje independiente del contexto.

# Complementación

**Teorema**

Los lenguajes independientes del contexto son cerrados respecto a unión, concatenación y cierre de Kleene, pero no lo son respecto a la complementación.

# Complementación

## Teorema

Los lenguajes independientes del contexto son cerrados respecto a unión, concatenación y cierre de Kleene, pero no lo son respecto a la complementación.

## Demostración

$$L_1 \cap L_2 = \overline{\overline{L_1} \cup \overline{L_2}}$$

## IMPORTANTE

Estas transparencias se utilizan ÚNICAMENTE como guía para el profesorado durante las clases.

Estas transparencias NO son un material completo y autocontenido para el uso del alumnado.