

# AUTÓMATAS Y LENGUAJES FORMALES

## LENGUAJES REGULARES II

### LENGUAJES INDEPENDIENTES DEL CONTEXTO

Francisco Hernández Quiroz

Departamento de Matemáticas  
 Facultad de Ciencias, UNAM  
 E-mail: fhq@ciencias.unam.mx  
 Página Web: www.matematicas.unam.mx/fhq

Posgrado en Ciencia e Ingeniería de la Computación

## Autómata cociente

Sea  $A \in \text{DFA}$ ,  $A = (Q, \Sigma, \delta, s, F)$ . Definimos una relación de equivalencia entre estados de  $Q$ . Sean  $q, r \in Q$ . Entonces:

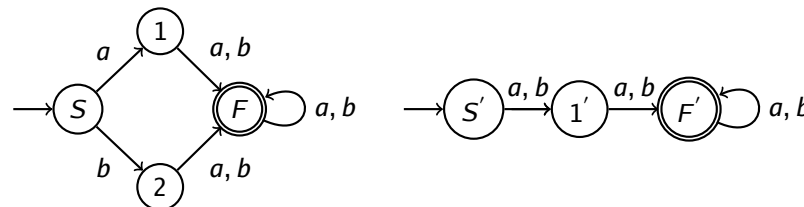
$$q \approx r \quad \text{sii} \quad \forall \alpha. \delta^*(q, \alpha) \in F \text{ sii } \delta^*(r, \alpha) \in F.$$

Dado que  $\approx$  es una relación de equivalencia, designaremos con  $[q]$  la clase de equivalencia a la que pertenece el estado  $q$ . El autómata cociente de  $A$ , que denotaremos con  $A/\approx = (Q_\approx, \Sigma, \delta_\approx, s_\approx, F_\approx)$ , se define así:

$$\begin{aligned} Q_\approx &= \{[q] \mid q \in Q\} \\ \delta_\approx([q], a) &= [\delta(q, a)] \\ s_\approx &= [s] \\ F_\approx &= \{[f] \mid f \in F\}. \end{aligned}$$

## Minimalización de estados

En ocasiones, un autómata  $A$  puede reemplazarse con un autómata  $A'$  que reconoce el mismo lenguaje pero que tiene un conjunto de estados menor:



De hecho, para cada lenguaje regular existe un único DFA (salvo isomorfismo) con un número mínimo de estados que lo reconoce.

## Algoritmo de minimalización I

El siguiente algoritmo permite construir el autómata cociente a partir del autómata  $A = (Q, \Sigma, \delta, s, F)$ , con  $Q = \{q_1, \dots, q_k\}$ :

- 1 Se construye una tabla
 

Estado	$q_1$	$\dots$	$q_k$
$q_1$	$m$	$\dots$	
$\vdots$			
$q_k$			

 Donde  $m = s$  si el estado en el  $i$ -ésimo renglón es final y el estado en la  $j$ -ésima columna no es final o viceversa. En caso contrario,  $m = n$ .
- 2 Se repite lo siguiente hasta que ya no haya cambios:  
 Si  $(q_i, q_j) = n$  y  $(\delta(q_i, a), \delta(q_j, a)) = s$ , para algún  $a \in \Sigma$ , entonces  $(q_i, q_j) := s$ .
- 3 Al terminar el paso anterior,  $q_i \approx q_j$  sii  $(q_i, q_j) = n$ .

## Algoritmo de minimalización II

El algoritmo toma cuando mucho

$$\binom{n}{2} = \frac{n!}{2!(n-2)!}$$

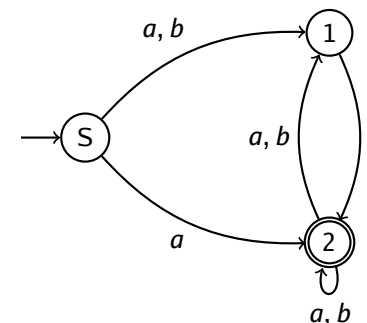
pasos, donde  $n$  es el número de estados.

## Autómatas finitos alternantes (AFA)

- Un autómata alternante generaliza la idea anterior a otras funciones booleanas. Por ejemplo, la aceptación de la cadena  $aa$  podría ocurrir si la expresión  $(\Delta(1, a) \cap F) \wedge (\neg \Delta(2, a) \cap F)$  es verdadera.
- En cada estado de un autómata alternante, la lectura de un símbolo lleva a la aplicación de una función booleana a los posibles resultados de procesar el resto de la cadena de entrada.

## Autómatas no deterministas y disyunción

Considérese el siguiente NFA:



En el estado inicial, la cadena  $aa$  será aceptada si la cadena  $a$  lleva a un estado de aceptación a partir del estado 1 o el estado 2. Es decir, si la disyunción  $(\Delta(1, a) \cap F) \vee (\Delta(2, a) \cap F)$  es verdadera.

## Definiciones preliminares

Sea  $Q$  un conjunto finito de estados. Una evaluación es una función  $e : Q \rightarrow \{V, F\}$ . Sea  $B^Q = \{e : Q \rightarrow \{V, F\}\}$ . El conjunto de funciones booleanas con argumentos en  $Q$  es  $\{b : B^Q \rightarrow \{V, F\}\}$ .  
*Ejemplo.* Sea  $Q = \{s, 1, 2\}$ . Una función booleana es  $b = s \wedge (1 \vee 2)$ . Sea  $e$  la evaluación siguiente

$$e(s) = V \quad e(1) = F \quad e(2) = V.$$

Entonces,  $b(e) = V$ . Si  $b' = s \wedge 1$  entonces  $b'(e) = F$ .

**Nota:** La función booleana no necesita utilizar todos sus argumentos, como en el ejemplo de  $b'$ .

## Definición formal

Un autómata alternante es un cuarteto  $A = (Q, \Sigma, \delta, s, F)$ , donde  $\delta : Q \times \Sigma \rightarrow \{b : B^Q \rightarrow \{V, F\}\}$ .

La función  $\delta$  se extiende a  $\delta^* : Q \times \Sigma^* \rightarrow \{b : B^Q \rightarrow \{V, F\}\}$ :

$$\begin{aligned} \delta^*(q, \varepsilon) &= q \\ \delta^*(q, a\alpha) &= \delta(q, a) \cdot (\delta^*(q_1, \alpha), \dots, \delta^*(q_n, \alpha)) \end{aligned}$$

Sea  $f : Q \rightarrow \{V, F\}$  la evaluación siguiente:

$$f(q) = V \quad \text{sii} \quad q \in F.$$

Entonces

$$L(A) = \{\alpha \mid \delta^*(s, \alpha)(f) = V\}$$

## Sistemas de ecuaciones

Una forma concisa de definir AFA es por medio de sistemas de ecuaciones. Sea  $\Gamma$  el alfabeto de entrada. Entonces, las ecuaciones tienen el formato:

$$\begin{aligned} X_1 &= \sum_{a \in \Gamma} a \cdot b_1(X_1, \dots, X_n) + c_1 \\ &\dots \\ X_n &= \sum_{a \in \Gamma} a \cdot b_n(X_1, \dots, X_n) + c_n \end{aligned}$$

donde las  $b_i$  son funciones booleanas de  $n$  argumentos y las  $c_i$  pueden ser  $\varepsilon$  (en caso de que sea un estado final) o  $\emptyset$ . Los operadores aritméticos y booleanos se interpretan como en las expresiones regulares o de manera tradicional.

## Ejemplo

Sea  $A = (\{s, 1, 2\}, \{a, b\}, \delta, s, \{2\})$ , donde  $\delta$  está definida así:

Estado	$a$	$b$
$s$	$1 \vee \neg 2$	$1 \wedge 2$
$1$	$\neg 2$	$1$
$2$	$2$	$\neg 1 \vee 2$

Entonces

$$\begin{aligned} \delta^*(s, baa)(f) &= \delta^*(1, aa) \wedge \delta^*(2, aa) \\ &= \neg \delta^*(2, a) \wedge \delta^*(2, a) \\ &= \neg \delta^*(2, \varepsilon) \wedge \delta^*(2, \varepsilon) \\ &= \neg 2 \wedge 2 \end{aligned}$$

y finalmente la cadena es rechazada, pues

$$\neg 2 \wedge 2(f) = F$$

## Ejemplo

El AFA del ejemplo anterior quedaría descrito por las siguientes ecuaciones:

$$\begin{aligned} X_0 &= a \cdot (X_1 \vee \neg X_2) + b \cdot (X_1 \wedge X_2) + \emptyset \\ X_1 &= a \cdot (\neg X_2) + b \cdot X_1 + \emptyset \\ X_2 &= a \cdot X_2 + b \cdot (\neg X_1 \vee X_2) + \varepsilon \end{aligned}$$

Un resultado interesante es que todo sistema de ecuaciones corresponde a un AFA y viceversa. La regularidad de los AFA se deduce de que los sistemas de ecuaciones pueden transformarse en expresiones regulares.

## Una aplicación: reconocimiento de cadenas

- Supongamos que se tiene un conjunto de cadenas  $X \subseteq \Sigma^*$  y una cadena  $\alpha \in \Sigma^*$  y el objetivo es verificar si alguna cadena de  $X$  aparece en  $\alpha$ , es decir si

$$\exists \beta, \gamma, \eta \in \Sigma^*. \alpha = \beta\gamma\eta \wedge \gamma \in X.$$

- El algoritmo ingenuo sigue el método de la *ventana corrediza*: para cada cadena en  $X$  se recorre  $\alpha$  carácter por carácter y se verifica si corresponde al segmento de  $\alpha$  que inicia ahí.
- No es difícil ver que este algoritmo es muy ineficiente.
- Hay algoritmos más eficientes que éste, pero requieren ciertas operaciones iniciales también muy costosas.
- Si se realizarán búsquedas frecuentes de cadenas de  $X$  en cadenas arbitrarias, el costo se podrá justificar.

## Autómata *trie* de un diccionario

Dado un un diccionario  $X$ , su autómata *trie*  $T(X)$  está formado por

- Un conjunto de estados que comprende un estado por cada prefijo de las cadenas de  $X$  (si más de una cadena tienen el mismo prefijo,  $T(X)$  sólo incluirá un estado para todas).
- El estado inicial corresponde a  $\epsilon$ .
- Los estados finales son las cadenas de  $X$ .
- La función  $\delta$  se define así. Sean  $p$  y  $q$  los estados correspondientes a las prefijos  $\alpha$  y  $\beta$ , respectivamente, y sea  $a \in \Sigma$ . Entonces

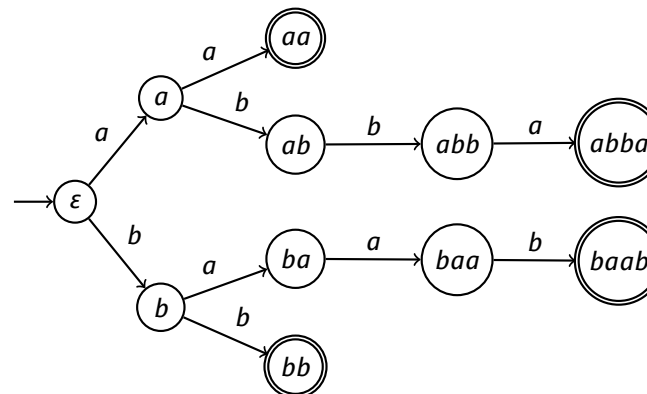
$$\delta(p, a) = q \quad \text{sii} \quad \beta = \alpha a.$$

## Diccionarios

- Sea  $\Sigma$  un alfabeto. Un diccionario es un subconjunto finito de  $\Sigma^*$  que no contiene a  $\epsilon$ .
- Buscar apariciones de cadenas de un diccionario  $X$  en cadenas arbitrarias de  $\Sigma^*$  equivale a reconocer el lenguaje  $\Sigma^*X$ .
- Sea  $\alpha \in \Sigma^*$  una cadena no vacía. Denotaremos con  $\text{Pre}(\alpha)$  los prefijos de  $\alpha$ . Si  $X$  es un conjunto de cadenas,  $\text{Pre}(X)$  es el conjunto de prefijos de las cadenas de  $X$ .
- Ejemplo. Si  $\alpha = abab$ ,  $\text{Pre}(\alpha) = \{\epsilon, a, ab, aba, abab\}$ .

## Ejemplo de autómata *trie*

Considérese el siguiente conjunto de palabras:  $\{aa, bb, abba, baab\}$ .



## Autómata de un diccionario I

- Sea  $X$  un diccionario. Con base en  $T(X)$  se puede construir el autómata del diccionario  $X$  que aceptará el lenguaje  $\Sigma^*X$ .
- Sea  $h : \Sigma^* \times X \rightarrow \text{Pre}(X)$  la función siguiente
 
$$h(\alpha, X) = \text{el sufijo de } \alpha \text{ de mayor longitud que pertenece a } \text{Pre}(X).$$
- El autómata del diccionario de  $X$ , denotado por  $D(X)$ , constará de los elementos
  - Los mismos estados de  $T(X)$ .
  - El mismo estado inicial.
  - El conjunto de estados finales es el conjunto de estados que corresponden a las cadenas en  $\text{Pre}(X) \cap \Sigma^*X$ .

## Autómata de un diccionario II

- La función  $\delta$  definida por

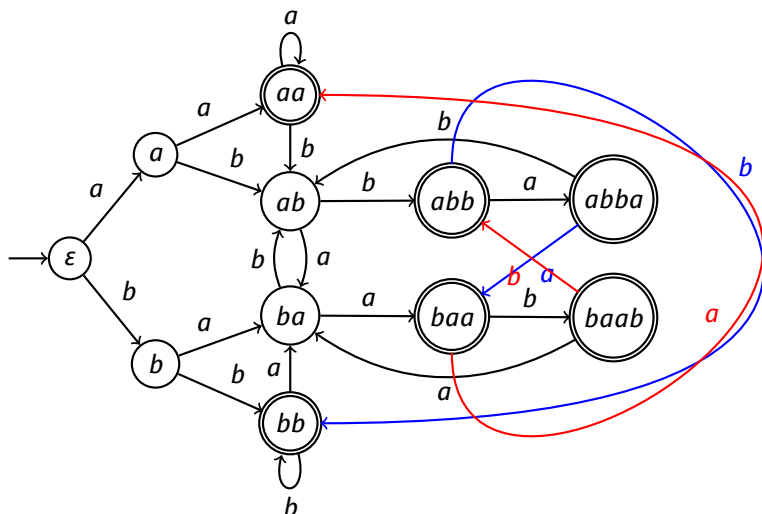
$$\delta(p, a) = q \quad \text{sii} \quad \beta = h(\alpha a, X),$$

donde  $p$  y  $q$  los estados correspondientes a las prefijos  $\alpha$  y  $\beta$ , respectivamente, y  $a \in \Sigma$ .

$D(X)$  es un autómata completo que acepta el lenguaje  $\Sigma^*X$ .

## Ejemplo de autómata de diccionario

El autómata de diccionario del ejemplo anterior:



## Gramáticas independientes del contexto

- Una gramática  $G = \langle \Sigma, \Gamma, S, \rightarrow \rangle$  es *independiente del contexto* sii todas las reglas de producción tienen una de las dos siguientes formas

$$A \rightarrow \alpha \quad A \rightarrow \epsilon,$$

donde  $A \in \Gamma$ .

- CFG designará el conjunto de gramáticas independientes del contexto.
- El lenguaje generado por  $G$  se denotará con  $L(G)$ .
- Un lenguaje  $L$  es *independiente del contexto* sii existe una gramática  $G \in \text{CFG}$  tal que

$$L = L(G).$$

- CFL es el conjunto de lenguajes independientes del contexto.

## Ejemplos

- 1 **Lenguaje de palindromas.** Sea  $G_P = \langle \{a, b\}, \{S\}, S, \rightarrow_P \rangle$ , donde

$$S \rightarrow_P aSa \mid bSb \mid \varepsilon.$$

- 2 **Lenguaje de paréntesis.** Sea  $G_D = \langle \{(, )\}, \{S\}, S, \rightarrow_D \rangle$ , con las siguientes reglas

$$S \rightarrow_D (S) \mid SS \mid \varepsilon.$$

## Otras definiciones II

y será *por la derecha* sii

$$|\eta_i| \leq |\eta_{i+1}| \quad \eta_i \in \Sigma^*.$$

Una gramática es *no ambigua* sii  $\forall \alpha \in L(G)$ ,  $\alpha$  tiene exactamente una derivación por la izquierda (o por la derecha).

Un lenguaje es *no ambiguo* sii existe una gramática no ambigua que lo genere. De lo contrario, será *inherentemente ambiguo*.

Una gramática es *pulcra* sii

- (a)  $\forall A \in \Gamma. L(A) \neq \emptyset$ ;
- (b)  $\forall A \in \Gamma. \exists \alpha, \beta \in \Sigma^*. S \rightarrow \alpha A \beta$  o  $S = A$ .

## Otras definiciones I

Considérese una derivación

$$\alpha_0 \Rightarrow \alpha_1 \Rightarrow \dots \Rightarrow \alpha_n.$$

Esta derivación presupone reglas de producción

$$A_1 \rightarrow \beta_1, \dots, A_n \rightarrow \beta_n$$

tales que

$$\alpha_i = \gamma_i A_i \eta_i \quad \alpha_{i+1} = \gamma_i \beta_i \eta_i.$$

Esta derivación será *por la izquierda* sii para todo  $i \leq n$

$$|\gamma_i| \leq |\gamma_{i+1}| \quad \gamma_i \in \Sigma^*$$

## Otras definiciones III

Una producción es *terminal* sii no contiene símbolos no terminales del lado derecho.

Una producción es una *regla-ε* sii su lado derecho es  $\varepsilon$ .

Una producción es *unitaria* sii es de la forma  $A \rightarrow B$ , con  $A, B \in \Gamma$ .

Una gramática es *propia* sii no contiene regla-ε ni reglas unitarias.

## Paréntesis balanceados

La gramática  $G_D$  genera el lenguaje de paréntesis balanceados, es decir,  $\alpha \in L(G_D)$  sii

- 1  $A(\alpha) = C(\alpha)$ ;
- 2  $C(\beta) \leq A(\beta) \quad \forall \beta \in Pre(\alpha)$ ;

donde  $A(\alpha)$  y  $C(\alpha)$  son el número de paréntesis que abren y cierran que aparecen en  $\alpha$ , respectivamente.

*Demostración.* Por inducción en la derivación de  $S \Rightarrow \alpha$ .

## Lenguajes de Dyck

El lenguaje de paréntesis se puede generalizar a los lenguajes de Dyck. Sea  $A$  un alfabeto de símbolos terminales y sea  $\bar{A}$  una copia de  $A$ , es decir:

- (i)  $A \cap \bar{A} = \emptyset$
- (ii) existe una biyección de  $A$  a  $\bar{A}$ .

Por ejemplo, si  $A = \{\{\}$ , entonces  $\bar{A} = \{\}\}$ .

El lenguaje de Dyck sobre  $A$  es el lenguaje generado por las reglas de producción:

$$S \rightarrow a_1 S \bar{a}_1 \mid \dots \mid a_n S \bar{a}_n \mid SS \mid \varepsilon,$$

donde  $A = \{a_1, \dots, a_n\}$  y  $\bar{a}_i \in \bar{A}$  es la copia de  $a_i$ . Este lenguaje se denotará con  $D_A^*$ .

Si  $|A| = n$ , una notación alternativa es  $D_n^*$ .

## Formas normales de Chomsky y de Greibach

Sea  $G = \langle \Sigma, \Gamma, S, \rightarrow \rangle$  un gramática.  $G$  tiene *forma normal de Chomsky* (CNF) sii todas las producciones tiene la forma

$$A \rightarrow BC \quad A \rightarrow a,$$

donde  $A, B, C \in \Gamma$  y  $a \in \Sigma$ .

En cambio,  $G$  tiene *forma normal de Greibach* (GNF) sii todas las producciones tienen la forma

$$A \rightarrow bB_1 \dots B_k$$

con  $0 \leq k, b \in \Sigma$  y  $B_1, \dots, B_k \in \Gamma$ .

## Ejemplos

Los conjuntos de producciones siguientes tienen forma normal de Chomsky y de Greibach (respectivamente)

$$S \rightarrow AB \mid AC \mid SS \quad C \rightarrow SB \quad A \rightarrow ( \quad B \rightarrow ) \\ S \rightarrow (B \mid (SB \mid (BS \mid (SBS \quad B \rightarrow )$$

y generan el lenguaje de paréntesis balanceados.

Las producciones siguientes generan el lenguaje de palíndromas

$$S \rightarrow AA \mid BB \mid AC \mid BD \quad C \rightarrow SA \quad D \rightarrow SB \quad A \rightarrow a \quad B \rightarrow b \\ S \rightarrow aA \mid aSA \mid bB \mid bSB \quad A \rightarrow a \quad B \rightarrow b$$

Con excepción en ambos casos de  $\varepsilon$ .

## Todas las CFG tienen equivalente en CNF o GNF

*Teorema.* Sea  $G \in \text{CFG}$ . Entonces existen  $G_C, G_G \in \text{CFG}$  tales que tienen forma normal de Chomsky y de Greibach, respectivamente, y

$$L(G_C) = L(G_G) = L(G) - \{\epsilon\}.$$

## Lema de gramáticas propias II

Y sea  $G' \in \text{CFG}$  como  $G$ , pero con las reglas de producción  $\rightarrow'$ . Entonces

$$L(G) = L(G')$$

Obviamente,  $L(G) \subseteq L(G')$ . Para  $L(G') \subseteq L(G)$ , baste observar que toda derivación en  $G'$  se puede realizar en  $G$ , salvo tal vez en más pasos. Sea ahora  $G_P$  como  $G'$  pero con la relación  $\rightarrow''$  que prescinde de las producciones unitarias y  $-\epsilon$  que existan en  $\rightarrow'$ . Entonces

$$L(G_P) = L(G') - \{\epsilon\} = L(G) - \{\epsilon\}.$$

Esta última afirmación se demuestra considerando que las derivaciones de longitud mínima en  $G'$  prescinden de producciones  $-\epsilon$  y unitarias.

## Lema de gramáticas propias I

Sea  $G \in \text{CFG}$ . Entonces, existe una gramática propia  $G_P$  tal que

$$L(G_P) = L(G) - \{\epsilon\}.$$

*Demostración.* Sea  $\rightarrow'$  la relación mínima que

- (i) contiene a  $\rightarrow$ ;
- (ii) si  $A \rightarrow' \alpha B \beta$  y  $B \rightarrow \epsilon$ , entonces  $A \rightarrow' \alpha \beta$
- (iii) si  $A \rightarrow' B$  y  $B \rightarrow' \gamma$ , entonces  $A \rightarrow' \gamma$ .

## Transformación a forma normal de Chomsky

Una vez que contamos con una gramática propia  $G_P$ , podemos construir la CNF:

- 1 Para cada  $a \in \Sigma$ , introducimos nuevos no terminales  $A_a$  y reglas de producción  $A_a \rightarrow a$ .
- 2 Sustituimos los terminales por los no terminales del punto anterior en todas las reglas de  $G_P$  que no tengan CNF todavía.
- 3 Dada una regla  $A \rightarrow B_1 \dots B_k$  ( $k > 2$ ), introducimos un nuevo no terminal  $C_1$  y las reglas

$$A \rightarrow B_1 C_1 \quad C_1 \rightarrow B_2 \dots B_k.$$

- 4 Repetimos este procedimiento hasta tener sólo reglas con dos no terminales del lado derecho.
- 5 Eliminamos las reglas del punto anterior que no tengan CNF.



## Transformación a forma normal de Greibach

Sea  $G_C$  una gramática en CNF. Para convertirla a GNF:

- 1 Para cada  $X \in \Gamma$ , considérese el lenguaje

$$R_{X,x} = \{\beta \in \Gamma^* \mid X \Rightarrow_L x\beta\}.$$

Este lenguaje es regular. Sea  $G_{X,x}$  una gramática lineal por la derecha que lo genere. Sea  $S_{X,x}$  el símbolo inicial de esta gramática.

- 2 Entonces, sustituimos las reglas

$$A \rightarrow XY$$

por reglas

$$A \rightarrow xS_{X,x}Y,$$

y agregamos las producciones y símbolos no terminales de  $G_{X,x}$  a nuestra gramática.

- 3 Eliminamos las reglas- $\epsilon$  y las reglas unitarias.

## Ejemplos I

Lenguaje de paréntesis. Empezamos con forma normal de Chomsky:

$$S \rightarrow AB \mid AC \mid SS \quad C \rightarrow SB \quad A \rightarrow ( \quad B \rightarrow ) .$$

Tenemos ahora los siguientes lenguajes  $R_{X,x}$

$$R_{S,(} = (B + C)S^*$$

$$R_{C,(} = (B + C)S^*B$$

$$R_{A,(} = \{\epsilon\}$$

$$R_{B,)} = \{\epsilon\}$$

Estas producciones nos generan los lenguajes anteriores:

$$\begin{aligned} S_{S,(} &\rightarrow BX \mid CX & X &\rightarrow SX \mid \epsilon \\ S_{C,(} &\rightarrow BY \mid CY & Y &\rightarrow SY \mid BZ & Z &\rightarrow \epsilon \\ S_{A,(} &\rightarrow \epsilon \\ S_{B,)} &\rightarrow \epsilon \end{aligned}$$

## Ejemplos II

Entonces, tenemos las reglas nuevas

$$\begin{aligned} S &\rightarrow (S_{A,(}B \mid (S_{A,(}C \mid (S_{S,(}S & C &\rightarrow (S_{S,(}B & A &\rightarrow ( \\ S_{S,(} &\rightarrow S_{B,)}X \mid (S_{C,(}X & X &\rightarrow (S_{S,(}X \mid \epsilon & B &\rightarrow ) \\ S_{C,(} &\rightarrow S_{B,)}Y \mid (S_{C,(}Y & Y &\rightarrow (S_{S,(}Y \mid S_{B,)}Z & Z &\rightarrow \epsilon \\ S_{A,(} &\rightarrow \epsilon & S_{B,)} &\rightarrow \epsilon \end{aligned}$$

Finalmente, eliminamos reglas- $\epsilon$  y símbolos superfluos:

$$\begin{aligned} S &\rightarrow (B \mid (C \mid (S_{S,(}S & C &\rightarrow (S_{S,(}B & A &\rightarrow ( \\ S_{S,(} &\rightarrow X \mid (S_{C,(}X \mid (S_{C,(} & X &\rightarrow (S_{S,(}X \mid (S_{S,(} & B &\rightarrow ) \\ S_{C,(} &\rightarrow Y \mid (S_{C,(}Y & Y &\rightarrow (S_{S,(}Y \mid ) \end{aligned}$$

## Teorema del bombeo I

Sea  $L \in CFL$ . Entonces, existe  $k \in \mathbb{N}$  tal que para toda  $\alpha \in L$ , si  $k \leq |\alpha|$ , entonces

$$\alpha = \beta\gamma\eta\theta\varphi, \quad \gamma\theta \neq \epsilon \quad \gamma \quad |\gamma\eta\theta| \leq k$$

y para toda  $i \in \mathbb{N}$

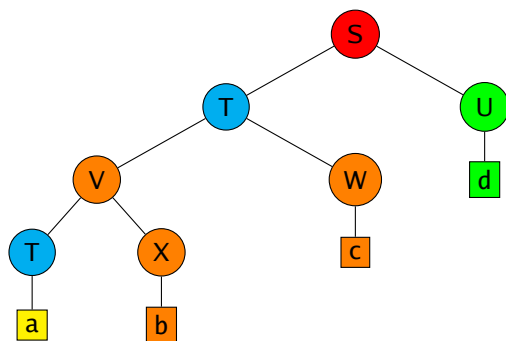
$$\beta\gamma^i\eta\theta^i\varphi \in L.$$

Demostración.

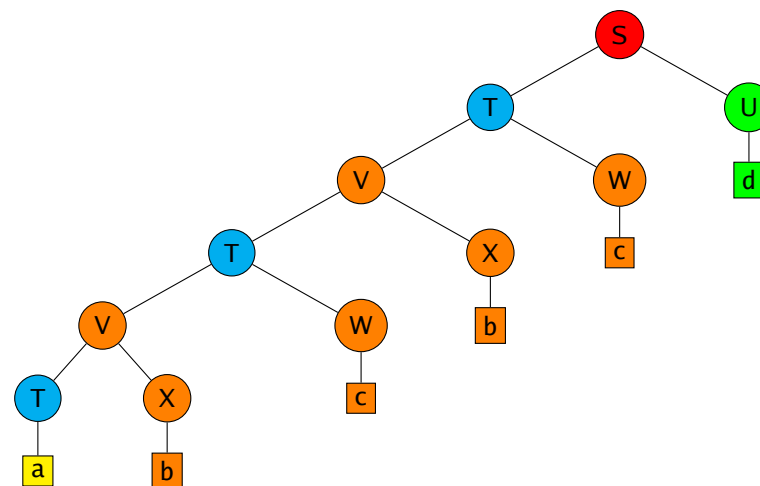
- Sea  $G \in CFG$  en CNF y  $L(G) = L - \epsilon$ .
- Sea  $\Gamma$  el alfabeto de símbolos no terminales de  $G$ , con  $|\Gamma| = n$  y sea  $k = 2^{n+1}$ .
- Entonces, el árbol sintáctico de toda cadena de longitud igual o superior a  $k$  debe tener profundidad de  $n + 1$  al menos.
- Entonces en un camino de longitud máxima de la raíz a las hojas debe aparecer dos veces, al menos, el mismo no terminal.
- Este no terminal puede usarse para definir las cadenas  $\gamma$  y  $\theta$ .

## Ejemplo

Considérense los siguientes árboles de derivación:



Árbol de la cadena  $\alpha = abcd$



Árbol de la cadena  $\beta\gamma^2\eta\theta^2\varphi = abc bcd$   
con  $\beta = a, \gamma = bc, \eta = \theta = \varepsilon \gamma \varphi = d$

## Aplicaciones

Como con los lenguajes regulares, este teorema se puede utilizar para demostrar que un lenguaje dado no es independiente del contexto.

Ejemplos:

$$\{a^n b^n c^n\},$$

por una aplicación directa del teorema, y

$$\{\alpha\alpha\}$$

pues los CFL y los regulares son cerrados bajo intersección y

$$\{a^n b^m a^n b^m\} = \{\alpha\alpha\} \cap L(a^* b^* a^* b^*)$$

y, obviamente,  $\{a^n b^m a^n b^m\}$  no es independiente del contexto

## Autómatas de pila

Un autómata de pila no determinista  $A$  está formado por

- un conjunto de estados  $Q$ ;
- un alfabeto de entrada  $\Sigma$ ;
- un alfabeto de pila  $\Gamma$ ;
- una relación de transición  $\delta \subseteq (Q \times (\Sigma \cup \{\varepsilon\}) \times \Gamma) \times (Q \times \Gamma^*)$
- un estado inicial  $s$ ;
- un símbolo inicial de la pila  $\perp$ ;
- un subconjunto de estados finales  $F \subseteq Q$ .

Llamaremos NPDA a los autómatas de pila no deterministas.

## Lenguajes aceptados por NPDA

Una configuración de un autómata  $A \in \text{NPDA}$  es una terna  $(q, \alpha, \beta)$  donde

$$\begin{aligned} q &\in Q \\ \alpha &\in \Sigma^* \\ \beta &\in \Gamma^* \end{aligned}$$

La configuración inicial es  $(s, \alpha, \perp)$ . Definiremos ahora una relación entre configuraciones:

$$\begin{aligned} \text{si } \delta((q, a, B), (r, \eta)) \text{ entonces } (q, a\alpha, B\beta) &\Rightarrow (r, \alpha, \eta\beta) \\ \text{si } \delta((q, \varepsilon, B), (r, \eta)) \text{ entonces } (q, \alpha, B\beta) &\Rightarrow (r, \alpha, \eta\beta) \end{aligned}$$

El lenguaje aceptado por un  $A \in \text{NPDA}$  es

$$L(A) = \{\alpha \in \Sigma^* \mid \exists f \in F. (s, \alpha, \perp) \Rightarrow^* (f, \varepsilon, \beta)\}$$

## Teorema de equivalencia entre CFG y NPDA I

(a) Sea  $G \in \text{CFG}$ . Entonces, existe  $A \in \text{NPDA}$  tal que

$$L(G) = L(A).$$

(b) Sea  $A \in \text{NPDA}$ . Entonces, existe  $G \in \text{CFG}$  tal que

$$L(G) = L(A).$$

*Demostración*

(a) Sea  $G = \langle \Sigma, \Gamma, S \rightarrow \rangle$  en FNG. Sea

$$A = (\{q\}, \Sigma, \Gamma, \delta, q, S, q),$$

donde

$$\delta((q, a, A), (q, B_1 \cdots B_k)) \quad \text{sii} \quad A \rightarrow aB_1 \cdots B_k.$$

## Aceptación por pila vacía

Una definición alternativa del lenguaje aceptado por un  $A \in \text{NPDA}$  es

$$L(A) = \{\alpha \in \Sigma^* \mid (s, \alpha, \perp) \Rightarrow^* (q, \varepsilon, \varepsilon)\}$$

Ambas definiciones son equivalentes y puede construirse un  $A' \in \text{NPDA}$  que acepte por pila vacía el mismo lenguaje que un  $A \in \text{NPDA}$  que acepte por estado final (y viceversa).

## Teorema de equivalencia entre CFG y NPDA II

Si puede demostrar por inducción en la longitud de las derivaciones (por la izquierda) que

$$(q, \alpha\beta, A) \Rightarrow_A^n (q, \beta, \gamma) \quad \text{sii} \quad A \Rightarrow_G^n \alpha\gamma.$$

Entonces

$$\begin{aligned} \alpha \in L(G) \quad \text{sii} \quad S &\Rightarrow_G^* \alpha \\ \text{sii} \quad (q, \alpha, S) &\Rightarrow_A^* (q, \varepsilon, \varepsilon) \\ \text{sii} \quad \alpha &\in L(A). \end{aligned}$$

(b) Se demuestra primero que para todo  $A \in \text{NPDA}$  existe un  $A' \in \text{NPDA}$  con un solo estado tal que

$$L(A) = L(A')$$

y una vez construido  $A'$ , los argumentos del caso (a) se pueden aplicar en sentido inverso.

## Determinismo

- A diferencia de REG, la clase CFL no determinista es estrictamente mayor que la determinista (DCFL).
- Un autómata determinista de pila  $D = (Q, \Sigma, \Gamma, \delta, \perp, s, F)$  incluye el símbolo especial  $\perp$  que sirve para indicar el final de la cadena de entrada.
- Las transiciones están indicadas por la función

$$\delta : Q \times \Sigma \cup \{\varepsilon, \perp\} \rightarrow Q \times \Gamma^*$$

- $\delta$  no puede eliminar de la pila a  $\perp$ , es decir

$$\delta(q, a, \perp) = (r, a\perp).$$

- La aceptación es por estados finales. La aceptación por pila vacía define un conjunto de lenguajes diferente.
- Un ejemplo de un lenguaje en CFL pero no en DCFL es

$$\{a, b\}^* - \{\alpha\alpha \mid \alpha \in \{a, b\}^*\}$$

## El teorema de Chomsky y Schützenberger I

Sea  $L \in CFL$ . Entonces existen  $n \in \mathbb{N}$ ,  $R \in REG$  y un homomorfismo  $h$  tal que

$$L = h(D_n^* \cap R).$$

Nota: un homomorfismo  $h : \Sigma^* \rightarrow \Gamma^*$  es una función tal que

$$h(\alpha\beta) = h(\alpha)h(\beta).$$

*Demostración.* Sea  $G = \langle \Sigma, \Gamma, S, \rightarrow \rangle$  en CNF. Asignaremos letras griegas minúsculas a las producciones de  $G$ :

$$\pi, \rho, \sigma, \dots$$

Definiremos nuevas producciones

$$\pi' = \begin{cases} A \rightarrow \left[ \begin{smallmatrix} 1 & 1 \\ \pi & \pi \end{smallmatrix} \right] \left[ \begin{smallmatrix} 2 & 2 \\ \pi & \pi \end{smallmatrix} \right] & \text{si } \pi = A \rightarrow BC \\ A \rightarrow \left[ \begin{smallmatrix} 1 & 1 & 2 & 2 \\ \pi & \pi & \pi & \pi \end{smallmatrix} \right] & \text{si } \pi = A \rightarrow a \end{cases}$$

## El teorema de Chomsky y Schützenberger II

y sean

$$\Sigma' = \left\{ \left[ \begin{smallmatrix} 1 & 1 \\ \pi & \pi \end{smallmatrix} \right], \left[ \begin{smallmatrix} 2 & 2 \\ \pi & \pi \end{smallmatrix} \right] \mid \pi \in \rightarrow \right\}$$

$$\rightarrow' = \{ \pi' \mid \pi \in \rightarrow \}$$

$$G' = \langle \Sigma', \Gamma, S, \rightarrow' \rangle$$

Se puede definir fácilmente un isomorfismo con un lenguaje  $D_n^*$  (con un par de tipos de paréntesis por cada regla en  $\rightarrow$ ). Obviando el isomorfismo, es claro que

$$L(G') \subseteq D_n^*.$$

$L(G')$  cumple además con las siguientes restricciones adicionales:

- 1 Para toda  $\pi \in \rightarrow$ , el símbolo  $\left[ \begin{smallmatrix} 1 & 1 \\ \pi & \pi \end{smallmatrix} \right]$  está seguido de  $\left[ \begin{smallmatrix} 2 & 2 \\ \pi & \pi \end{smallmatrix} \right]$ .
- 2 Ningún  $\left[ \begin{smallmatrix} 2 & 2 \\ \pi & \pi \end{smallmatrix} \right]$  está seguido de  $\left[ \begin{smallmatrix} 1 & 1 \\ \pi & \pi \end{smallmatrix} \right]$ .

## El teorema de Chomsky y Schützenberger III

- 3 Si  $\pi = A \rightarrow BC$  entonces  $\left[ \begin{smallmatrix} 1 & 1 \\ \pi & \pi \end{smallmatrix} \right]$  está seguido de  $\left[ \begin{smallmatrix} 1 & 1 \\ \rho & \rho \end{smallmatrix} \right]$ , con  $\rho = B \rightarrow \beta$ .

Análogamente para  $\left[ \begin{smallmatrix} 2 & 2 \\ \pi & \pi \end{smallmatrix} \right]$ .

- 4 Si  $\pi = A \rightarrow a$ , entonces  $\left[ \begin{smallmatrix} 1 & 1 \\ \pi & \pi \end{smallmatrix} \right]$  está seguido por la cadena  $\left[ \begin{smallmatrix} 1 & 2 & 2 \\ \pi & \pi & \pi \end{smallmatrix} \right]$ .

- 5 Si  $A \Rightarrow_G^* \alpha$ , entonces  $\alpha$  comienza con  $\left[ \begin{smallmatrix} 1 & 1 \\ \pi & \pi \end{smallmatrix} \right]$  para alguna  $\pi = A \rightarrow \gamma$ .

Considérese el siguiente lenguaje regular:

$$R_A = \{ \alpha \in \Sigma'^* \mid \alpha \text{ satisface 1-5} \}$$

*Lema.*  $A \Rightarrow_G^* \alpha$  sii  $\alpha \in D_n^* \cap R_A$ .

Suponiendo que el lema anterior es verdadero, entonces definimos el homomorfismo

$$h : \Sigma'^* \rightarrow \Sigma^*.$$

## El teorema de Chomsky y Schützenberger IV

Si  $\pi = A \rightarrow BC$  entonces

$$h\left(\begin{matrix} 1 \\ \pi \end{matrix}\right) = h\left(\begin{matrix} 1 \\ \pi \end{matrix}\right) = h\left(\begin{matrix} 2 \\ \pi \end{matrix}\right) = h\left(\begin{matrix} 2 \\ \pi \end{matrix}\right) = \varepsilon.$$

Si  $\pi = A \rightarrow a$  entonces

$$h\left(\begin{matrix} 1 \\ \pi \end{matrix}\right) = a \quad \text{y} \quad h\left(\begin{matrix} 1 \\ \pi \end{matrix}\right) = h\left(\begin{matrix} 2 \\ \pi \end{matrix}\right) = h\left(\begin{matrix} 2 \\ \pi \end{matrix}\right) = \varepsilon.$$

El teorema se sigue directamente de esta definición.

## Algoritmo de Cocke, Kasami y Younger

- Es un algoritmo para *decidir* de manera eficiente si una cadena pertenece a un CFL.
- Sea  $L$  un CFL y sea  $G = \langle \Sigma, \Gamma, S, \rightarrow \rangle$  una CFG en forma normal de Chomsky tal que

$$L(G) = L.$$

- Sea  $\alpha \in L$  con longitud  $n$ . Sean  $0 \leq i < j \leq n$  y sea

$$\alpha_{i,j}$$

la subcadena de  $\alpha$  que va de la posición  $i + 1$  a la  $j$ .

- Sea  $T_{i,j} \subseteq \Gamma$  el conjunto de no terminales que generaron  $\alpha_{i,j}$ .
- El algoritmo calcula el valor de todos los  $T_{i,j}$  de manera inductiva. En particular,  $S \in T_{0,n}$  si  $S \Rightarrow_G^* \alpha$ .
- El algoritmo es  $O(pn^3)$ ,  $p$  el número de producciones en  $G$ .

## Lema $A \Rightarrow_G^* \alpha$ sii $\alpha \in D_n^* \cap R_A$

*Demostración.* Por inducción en  $n$  para la dirección  $\Rightarrow$ .

Para el caso inverso, por inducción en la longitud de las cadenas en  $D_n^* \cap R_A$ . Basta considerar que

$$\alpha = \begin{matrix} 1 & 2 \\ \beta & \gamma \end{matrix}$$

y analizar los dos casos posibles para  $\pi$ , a saber,  $A \rightarrow BC$  o bien  $A \rightarrow a$ .

En el primer caso, por hipótesis inductiva,

$$B \Rightarrow_G^* \beta \quad \text{y} \quad C \Rightarrow_G^* \gamma$$

y además  $\beta$  y  $\gamma$  satisfacen las condiciones 1-5. El segundo caso es aún más simple.

**for**  $i := 0$  **to**  $n - 1$  **do**

$T_{i,i+1} := \emptyset;$

**for all**  $A \rightarrow a \in G$  **do**

**if**  $a = \alpha_{i,i+1}$  **then**

$T_{i,i+1} := T_{i,i+1} \cup \{A\}$

**for**  $m := 2$  **to**  $n$  **do**

**for**  $i := 0$  **to**  $n - m$  **do**

$T_{i,i+m} := \emptyset;$

**for**  $j := i + 1$  **to**  $i + m - 1$  **do**

**for all**  $A \rightarrow BC \in G$  **do**

**if**  $B \in T_{i,j} \wedge C \in T_{j,i+m}$  **then**

$T_{i,i+m} := T_{i,i+m} \cup \{A\}$

## Un ejemplo I

Reconocimiento de  $abba$  en el lenguaje de palindromos (p. 28).

Inicialmente agregamos al menos un no terminal a los  $T_{i,i+1}$

$$T_{0,1} = \{A\} \quad T_{1,2} = \{B\} \quad T_{2,3} = \{B\} \quad T_{3,4} = \{A\}.$$

Después consideramos los segmentos de longitud 2.

$S \in T_{1,3}$  porque  $B \in T_{1,2}$  y  $B \in T_{2,3}$  y por la regla  $S \rightarrow BB$

$$T_{0,2} = \emptyset \quad T_{1,3} = \{S\} \quad T_{2,4} = \emptyset$$

Hay más de una forma de construir los segmentos de longitud 3. En particular,

$$\alpha_{1,4} = \alpha_{1,2} \cdot \alpha_{2,4} = \alpha_{1,3} \cdot \alpha_{3,4}.$$

La última es el que nos da  $C$  con la regla  $C \rightarrow SA$

$$T_{0,3} = \emptyset \quad T_{1,4} = \{C\}.$$

## Un ejemplo II

Finalmente,  $S \rightarrow AC$  y  $T_{0,1}$  y  $T_{1,4}$  nos dan

$$T_{0,4} = \{S\}.$$

*Resultado:* aceptación.

Reconocimiento de  $bba$ .

$$\begin{aligned} T_{0,1} &= \{B\} & T_{1,2} &= \{B\} & T_{2,3} &= \{A\} \\ T_{0,2} &= \{S\} & T_{1,3} &= \emptyset \\ T_{0,3} &= \{C\}. \end{aligned}$$

*Resultado:* rechazo, pues  $S \notin T_{0,3}$ .