Gramáticas Libres de Contexto

Pedro J. Álvarez Pérez-Aradros
Rubén Béjar Hernández

Departamento de Informática e Ingeniería de Sistemas
C.P.S. Universidad de Zaragoza

Gramáticas Regulares

- Una ***gramática regular*** es una 4-tupla $G=(\Sigma,N,S,P)$, donde Σ es el *alfabeto*, N es una colección de *símbolos no terminales*, S es un símbolo no terminal llamado *símbolo inicial* y P es un conjunto de reglas de sustitución, llamadas *producciones* de la forma $A \rightarrow w$ donde $A \in N$ y $w \in (\Sigma \cup N \cup \{\varepsilon\})$ satisfaciendo:

  1) w contiene un no terminal como máximo.
  2) Si w contiene un no terminal, entonces es el símbolo que está en el extremo derecho de w.

- El lenguaje generado por la gramática se representa como L(G).

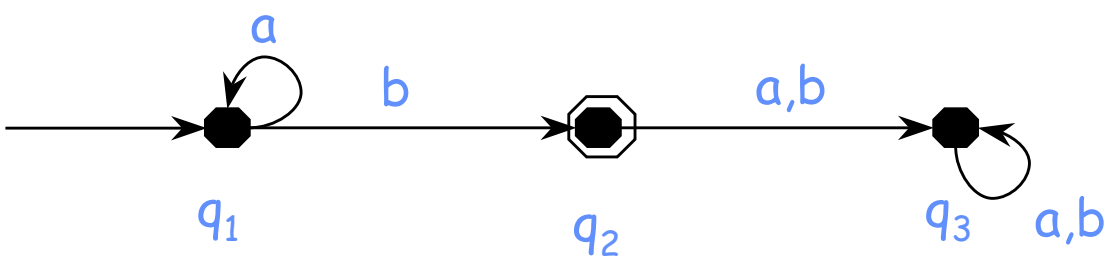
Gramáticas Regulares (II)

- Sea L el lenguaje regular reconocido por un AFD $M = (\Sigma, Q, s, F, \delta)$. La gramática regular que genera el lenguaje L será $G = (\Sigma, N, S, P)$ tal que:
  - $\Sigma = \Sigma$
  - $N = Q$
  - $S = s$
  - $P = \{q \rightarrow ap \mid \delta(q, a) = p\} \cup \{q \rightarrow \varepsilon \mid q \in F\}$

- Todas aquellas palabras que reconoce el AFD M, pueden ser generadas a partir de las producciones de G.

Gramáticas Regulares (III)

- Ejemplo: Sea el AFD M,



¿Cuál es la gramática G que genera L(M)?

Gramáticas Regulares (IV)

- Sea L el lenguaje generado por la gramática regular G = (Σ, N, S, P). El AFN que reconoce el lenguaje L será M = (Σ, Q, s, F, Δ) tal que:
  - Q = N ∪ {f}, siendo f un símbolo nuevo
  - s = S
  - F = {f}
  - Δ se definen:
    - Si $A \rightarrow \sigma_1 \ldots \sigma_n B$ con A y B no terminales, entonces se añadirán a Q los nuevos estados q_1, q_2, $\ldots q_{n-1}$ y las transformaciones siguientes:

      $$\Delta(A, \sigma_1 \ldots \sigma_n) = \Delta(q_1, \sigma_2 \ldots \sigma_n) = \ldots = \Delta(q_{n-1}, \sigma_n) = B$$

    - Si $A \rightarrow \sigma_1 \ldots \sigma_n$ con A no terminal, entonces se añadirán a Q los nuevos estados q_1, q_2, $\ldots q_{n-1}$ y las transformaciones siguientes:

      $$\Delta(A, \sigma_1 \ldots \sigma_n) = \Delta(q_1, \sigma_2 \ldots \sigma_n) = \ldots = \Delta(q_{n-1}, \sigma_n) = f$$

Gramáticas Regulares (V)

- Ejemplo: Sea la gramática regular G,

$$S \rightarrow aB \mid bA \mid \varepsilon$$

$$A \rightarrow abaS$$

$$B \rightarrow babS$$

¿Cuál el AFN que reconoce L(G)?

Gramáticas Independientes del Contexto

- Una ***Gramática Independiente del Contexto*** (GIC) es una 4-tupla

$$\mathbf{G= (\Sigma, N, S, P)}$$

donde Σ es el *alfabeto* (conjunto de terminales), N es la colección finita de los *no terminales*, S es un no terminal llamado *símbolo inicial* y $P \subseteq N \times (N \cup \Sigma)^*$ es un conjunto de *producciones*.

- El lenguaje generado por la GIC se denota L(G) y se llama ***Lenguaje Independiente del Contexto*** (LIC).

- Relación GR y GIC:

$$GR \subseteq GIC$$

$$LR \subseteq LIC$$

Árboles de derivación o de análisis de ambigüedad

- En las GIC puede haber, en el lado derecho de sus producciones, más de un no terminal y además, éste puede aparecer en cualquier lugar de la cadena.

$$P \subseteq N \times (N \cup \Sigma)^*$$

- Objetivo: Realizar un gráfico de derivación, que indique de qué manera ha contribuido cada no terminal a formar la cadena final de símbolos terminales (***árbol de derivación***).

Árboles de derivación o de análisis de ambigüedad (II)

- Características de los árboles de derivación:
  - El nodo raíz se etiqueta con el símbolo inicial S.
  - Los nodos hijo del nodo raíz se etiquetan con los símbolos que aparecen en la parte derecha de la producción del símbolo inicial.
  - Todo nodo etiquetado con un no terminal, tiene como nodos hijos los etiquetados con los símbolos del lado derecho de la producción usada para sustituir dicho no terminal.
  - Los nodos que no tienen hijos deben estar etiquetados con símbolos terminales.

Árboles de derivación o de análisis de ambigüedad (III)

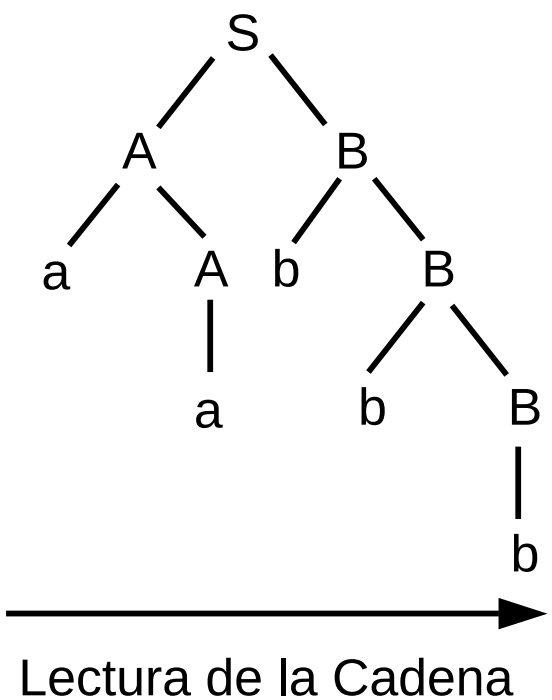
- Ejemplo: Sea la GIC

$$S \rightarrow AB$$

$$A \rightarrow aA \mid a$$

$$B \rightarrow bB \mid b$$

la cadena w = aabbb puede ser derivada



Árboles de derivación o de análisis de ambigüedad (IV)

- Otra representación proceso de derivación para la cadena w:

  $S \Rightarrow AB \Rightarrow AbB \Rightarrow AbbB \Rightarrow Abbb \Rightarrow aAbbb \Rightarrow aabbb$

  $S \Rightarrow AB \Rightarrow aAB \Rightarrow aaB \Rightarrow aabB \Rightarrow aabbB \Rightarrow aabbb$

  $S \Rightarrow AB \Rightarrow aAB \Rightarrow aAbB \Rightarrow aAbbB \Rightarrow aAbbb \Rightarrow aabbb$

  Preguntas:

  ¿Cuál sería el árbol de derivación para cada una de las derivaciones anteriores de la cadena?

  ¿Qué tienen en común los tres árboles de derivación que se obtienen?

Árboles de derivación o de análisis de ambigüedad (V)

- Ejemplo: Sea la GIC

$$S \rightarrow SbS \mid ScS \mid a$$

se pretende derivar la cadena w = abaca.

Preguntas:

¿Qué posibles derivaciones de la cadena w podemos realizar?

¿Cuál es el árbol de derivación de cada una de las derivaciones previas?

¿Cómo son los árboles de derivación obtenidos?

Árboles de derivación o de análisis de ambigüedad (VI)

- Una gramática en la cual, para toda cadena w, todas las derivaciones de w tienen el mismo árbol de derivación, es ***no ambigua***.
- En algunos casos, si la gramática es ambigua, se puede encontrar otra gramática que produzca el mismo lenguaje y no lo sea.

$S \rightarrow A \mid B$

$A \rightarrow a$

$B \rightarrow a$

Gramática Ambigua

$S \rightarrow a$

Gramática No Ambigua

- Si todas las GIC para un lenguaje son ambiguas, se dice que el lenguaje es un ***Lenguaje Independiente del Contexto inherentemente ambiguo.***

Árboles de derivación o de análisis de ambigüedad (VII)

- Formas de derivar cadenas:
  - Derivación por la izquierda: el no terminal que se expande es, siempre, el del extremo izquierdo. Para la gramática

$$S \rightarrow SbS \mid ScS \mid a$$

la derivación por la izquierda de la cadena w = abaca

$$S \Rightarrow ScS \Rightarrow SbScS \Rightarrow abScS \Rightarrow abacS \Rightarrow abaca$$

  - Derivación por la derecha: el no terminal que se expande es, siempre, el del extremo derecho. Para la gramática

$$S \rightarrow SbS \mid ScS \mid a$$

la derivación por la derecha de la cadena w = abaca

$$S \Rightarrow ScS \Rightarrow Sca \Rightarrow SbSca \Rightarrow Sbaca \Rightarrow abaca$$

Simplificación de las GIC

- Objetivo: Encontrar un modelo formal estándar para las producciones (forma normal).
  - El primer paso para el desarrollo del modelo, es limpiar las gramáticas eliminado aquellas producciones y símbolos inútiles.
  - La eliminación de éstos problemas no afecta a la capacidad de generación de las gramáticas.
- Ejemplo: Sea la GIC

$$S \rightarrow Aa \mid B \mid D$$

$$A \rightarrow aA \mid bA \mid B$$

$$B \rightarrow b$$

$$C \rightarrow abd$$

¿Qué problemas podemos encontrar en esta gramática?

Simplificación: Transformación 1

- Objetivo transformación: Eliminar los no terminales que no deriven cadenas de terminales.
- Sea $G=(\Sigma, N, S, P)$ una GIC. Transformamos G en $G'=(\Sigma, N', S, P')$ de forma que $L(G)=L(G')$ y para todo $A \in N'$, se obtenga $A \stackrel{*}{\Rightarrow} w$ tal que $w \in \Sigma^*$. Para lograrlo construimos N' y P' de manera iterativa:

1. Inicializar N' con todos los no terminales A para los que $A \rightarrow w$, es una producción de G, con $w \in \Sigma^*$.
2. Inicializar P' con todas las producciones $A \rightarrow w$ para $A \in N'$ y $w \in \Sigma^*$.
3. **Repetir**

   Añadir a N' todos los no terminales A para los cuales $A \rightarrow w$, para algún $w \in (N' \cup \Sigma)^*$, que sea una producción de P y añadirla a P'.

   Hasta que no se puedan añadir más no terminales a N'

Nota: Este algoritmo trata ε como una cadena de un terminal (ε-producciones)

Simplificación: Transformación 2

- Objetivo transformación: Eliminar aquellos terminales y no terminales que no aparezcan en las cadenas que se deriven a partir de S.
- Sea $G=(\Sigma, N, S, P)$ una GIC. Transformamos G en $G'=(\Sigma', N', S, P')$ de forma que $L(G)=L(G')$ y para todo $X \in N' \cup \Sigma'$, se tenga $S \overset{*}{\Rightarrow} \alpha X \beta$ tal que $\alpha, \beta \in (N' \cup \Sigma')^*$. Para lograrlo construimos Σ', N' y P' de forma iterativa:

1. Inicializar $N'=\{S\}$, $P'=\phi$ y $\Sigma'=\phi$
2. **Repetir**

   Para $A \in N'$, si $A \rightarrow w$ es una producción de P:

   1. Introducir $A \rightarrow w$ en P'.
   2. Para todo no terminal B de w, introducir B en N'
   3. Para todo terminal α de w, introducir α en Σ'

   Hasta que no se puedan añadir nuevas producciones

Simplificación: Transformación 1 y 2

- Problema: Dependiendo del orden en que se apliquen los dos algoritmos previos el resultado es distinto.
- Ejemplo: Sea la GIC

$$S \rightarrow AB \mid a$$

$$A \rightarrow a$$

Aplicando Algoritmo 1 y Algoritmo 2:

$$S \rightarrow a$$

Aplicando Algoritmo 2 y Algoritmo 1:

$$S \rightarrow a$$

$$A \rightarrow a$$

Simplificación: Transformación 3

- Objetivo transformación: Eliminación de las ε-producciones.
  - Si $\varepsilon \in L(G)$ entonces tales producciones no pueden ser eliminadas para que ε pueda se generada por la gramática.
  - Si $\varepsilon \notin L(G)$ entonces las ε-pueden ser eliminadas.

- Un no terminal A es **anulable** si $A \stackrel{*}{\Rightarrow} \varepsilon$.
  - Para eliminar las ε-producciones es crucial identificar los no terminales que son anulables.

- Ejemplo: Sea la GIC

$$S \rightarrow aA$$

$$A \rightarrow aA \mid \varepsilon$$

Simplificación: Transformación 3 (II)

- Sea G= (Σ, N, S, P) una GIC. El siguiente algoritmo identifica Ɗ como el conjunto de todos los no terminales anulables de G:

1. Inicializar Ɗ con todos los no terminales A de G para los que existe una producción $A \rightarrow \varepsilon$.
2. **Repetir**

   Si $B \rightarrow w$ para algún $w \in (N \cup \Sigma)^*$ y todos los símbolos de w están en Ɗ, añadir B a Ɗ

   Hasta que no se puedan añadir más no terminales a Ɗ

Simplificación: Transformación 3 (III)

- Transformación para GIC donde $\varepsilon \notin L(G)$:
  Identificados los no terminales anulables, se modifican las reglas de producción con el fin de poder eliminar las ε-producciones.
- El nuevo conjunto P' se crea a partir de:

  Si $B \rightarrow X_1 X_2 \ldots X_n \in P$, entonces en P' introducimos todas las producciones de la forma $B \rightarrow Y_1 Y_2 \ldots Y_n$ donde las Y_i satisfacen:

  $Y_i = X_i$, si X_i es no anulable

  $Y_i = X_i$ ó ε , si X_i es anulable

  Y_i no es ε para todo i (no se introducen en P' producciones de la forma $B \rightarrow \varepsilon$)

Simplificación: Transformación 3 (IV)

- Transformación para GIC donde $\varepsilon \in L(G)$: Se pueden eliminar todas las ε-producciones de G menos una.
- El nuevo conjunto P' se crea:
  1. Eliminar todas las ε-producciones de G , transformando G en G' tal que $L(G')=L(G)-\{\varepsilon\}$
  2. Añadir la producción $S \rightarrow \varepsilon$ que restituye ε al lenguaje generado por la gramática.

Simplificación: Transformación 4

- Objetivo transformación: Eliminación de las producciones unitarias o no generativas (aumentan la complejidad de las gramáticas).
- Las producciones de la forma A→B donde A y B son no terminales, se llaman producciones unitarias o no generativas.
- Primera aproximación: Renombrar no terminales y añadir un paso más a la derivación.

  $$A \rightarrow B$$

  $$B \rightarrow w1 \mid C$$

  lo transformaríamos a

  $$A \rightarrow w1 \mid C$$

- Problema Primera Aproximación: Al eliminar producciones unitarias aparecen nuevas producciones unitarias.

Simplificación: Transformación 4 (II)

- Sea G=(Σ, N, S, P) una GIC que no contenga ε-producciones. Construimos una GIC G'=(Σ, N, S, P') equivalente que no contenga producciones unitarias:

1. Inicializar P' de forma que contenga todos los elementos de P.
2. Para cada no terminal $A \in N$ obtener el conjunto Unitario(A)

   $Unitario(A)=\{B \in N \mid A \overset{*}{\Rightarrow} B$ usando solamente producciones unitarias$\}$

   Nota: $A \in Unitario(A)$ porque $A \overset{0}{\Rightarrow} A$
3. Para cada A para el cual $Unitario(A) \neq \{A\}$

   Para cada $B \in Unitario(A)$

   Para cada producción no unitaria $B \rightarrow w$ de P añadir $A \rightarrow w$ a P'
4. Eliminar todas las producciones unitarias de P'

Simplificación: Transformación 4 (III)

- Ejemplo: Sea la GIC

$$S \rightarrow A \mid Aa$$

$$A \rightarrow B$$

$$B \rightarrow C \mid b$$

$$C \rightarrow D \mid ab$$

$$D \rightarrow b$$

Eliminación de las producciones unitarias o no generativas de G.

Forma Normal de Chomsky

- Es un modelo o forma normal para las producciones.
- Se dice que una GIC está en **Forma Normal de Chomsky**, si no contiene ε-producciones y si todas las producciones son de la forma:
  - $A \rightarrow a$, para $a \in \Sigma$
  - $A \rightarrow BC$, con B y C no terminales

- ¿Toda GIC puede ser transformada en un GIC en Forma Normal de Chomsky?
  - Sea G una GIC tal que $\varepsilon \notin L(G)$
  - Sea G una GIC tal que $\varepsilon \in L(G)$

Forma Normal de Chomsky (II)

- Sea G una GIC y $\varepsilon \notin L(G)$, G puede ser transformada en una GIC en F.N. de Chomsky.

1. Eliminar las ε-producciones (transformación 3).
2. Eliminar las producciones unitarias (transformación 4).
3. Eliminar símbolos inútiles (primero transformación 1, luego transformación 2).
4. Para las producciones de la forma $A \rightarrow w$ y $|w| > 1$, donde $w = X_1 X_2 \ldots X_n$, si X_i es un terminal, llamado σ, sustituimos X_i por un no terminal llamado C_σ y añadimos la producción $C_\sigma \rightarrow \sigma$.
5. Transformar aquellas producciones con más de dos no terminales que se encuentren en el lado derecho de la misma. Si $A \rightarrow B_1 B_2 \ldots B_n$, $n \geq 2$ se reemplaza por (n-1) producciones

$$A \rightarrow B_1 D_1$$

$$D_1 \rightarrow B_2 D_2$$

$$\ldots$$

$$D_{n-2} \rightarrow B_{n-1} B_n$$

Donde los D_i son nuevos no terminales.

Forma Normal de Chomsky (III)

- Ejemplo: Sea la GIC

$$S \rightarrow bA \mid aB$$

$$A \rightarrow bAA \mid aS \mid a$$

$$B \rightarrow aBB \mid bS \mid b$$

Transformarla a una GIC en Forma Normal de Chomsky.

Forma Normal de Chomsky (IV)

- Sea G una GIC y $\varepsilon \in L(G)$, G puede ser transformada en una GIC en F.N. de Chomsky.
  1. Se obtiene la gramática en Forma Normal de Chomsky que genere el lenguaje $L(G)-\varepsilon$
  2. Añadir a la gramática resultante la producción $S \rightarrow \varepsilon$

- ¿Qué problema hay con la GIC obtenida de la transformación previa?

Forma Normal de Greibach

- Definición: Una producción de la forma $A \rightarrow \alpha A$ con $\alpha \in (N \cup \Sigma)^*$ se conoce como recursiva por la derecha. De la misma forma, una producción de la forma $A \rightarrow A\alpha$ se conoce como recursiva por la izquierda.

- Problema: Cuando hay recursividad por la izquierda, los árboles de derivación se expanden por la izquierda. Para algunas aplicaciones de las GIC no es deseable la recursividad por la izquierda.

Forma Normal de Greibach (II)

- Definición : Una GIC está en Forma Normal de Greibach (FNG) si todas las producciones son de la forma $A \rightarrow a\alpha$ donde a es un símbolo terminal y $\alpha \in (N \cup \Sigma)^*$.
  - No puede tener producciones recursivas a la izquierda
  - Cada producción requiere un símbolo del alfabeto, con lo que una GIC en FNG sólo puede generar lenguajes no vacíos y que no contengan ε

Forma Normal de Greibach (III)

- Teorema: Todo Lenguaje LIC no vacio, que no contiene la palabra vacia ε, se puede generar por medio de una gramática GIC en forma Normal de Greibach.
- Toda GIC que genera un lenguaje no vacio que no contine la palabra vacia, se puede transformar en una GIC en FN de Greibach.
- Entonces ¿True/False?:
  - Una gramática regular nunca se puede poner en Forma Normal de Chomsky.
  - Una gramática G con $\varepsilon \notin L(G) \neq \phi$ que esté en Forma Normal de Chomsky nunca es una gramática regular.
  - Si se tiene una gramática cuyo conjunto de producciones es P:{S→a} entonces G es regular y además está en Forma Normal de Chomsky.
  - Toda gramática en Forma Normal de Greibach es una gramática regular.

Propiedades de los LIC

- Objetivo: Encontrar alguna propiedad que caracterice a los LIC para poder determinar si un lenguaje es LIC.
- *Lema de Bombeo*: Sea L un LIC que no contiene ε. Entonces existe un entero *k* para el cual, si $z \in L$ y $|z|>k$, entonces *z* se puede escribir como $z=uvwxy$ con las propiedades siguientes:
  1. $|vwx| \leq k$
  2. Al menos o *v* o *x* no es ε
  3. $uv^iwx^iy \in L$ para todo $i \geq 0$
- Ejemplo: Sea el lenguaje

  $$L=\{a^ib^j \mid j = i^2\}$$

  ¿Es un Lenguaje Independiente del Contexto?

Propiedades de Cierre de los LIC

- Teorema: Si L_1 y L_2 son LIC, entonces $L_1 \cup L_2$ es un LIC.
- Sea $G_1=(\Sigma_1,N_1,S_1,P_1)$ y $G_2=(\Sigma_2,N_2,S_2,P_2)$ dos GIC para $N_1 \cap N_2=\phi$. Definimos $G=(\Sigma,N,S,P)$ tal que,
  - $\Sigma = \Sigma_1 \cup \Sigma_2$
  - $N = N_1 \cup N_2 \cup \{S\}$
  - S, nuevo símbolo inicial
  - $P = P_1 \cup P_2 \cup \{(S,S_1), (S,S_2)\}$

  siendo $L(G) = L(G_1) \cup L(G_2)$.

Propiedades de Cierre de los LIC (II)

- Teorema: Si Les LIC, entonces L* es un LIC.
- Sea G=(Σ,N,S,P) una GIC. Definimos G'=(Σ',N',S',P') tal que,
  - Σ' = Σ
  - N' = N ∪ {S', T}
  - S', nuevo símbolo inicial
  - P' = P ∪ {(S', ST), (S',ε), (T, ST), (T, ε)}

  siendo L(G') = L(G)*.

Propiedades de Cierre de los LIC (III)

- Teorema: Si L_1 y L_2 son LIC, entonces L_1L_2 es un LIC.
- Ejercicio: Definir la GIC G, en función de las GIC G_1 Y G_2, que genere el lenguaje $L(G)=L(G_1)L(G_2)$.

- Teorema: Los LIC no son cerrados respecto de las operaciones de intersección y complementación.

Autómatas de Pila

- Sean los LIC,

$L_1 = \{a^n b^n \mid n \geq 0\}$

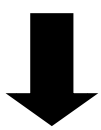
$L_2 = \{wcw^I \mid w \in \{a,b,c\}^*\}$

¿qué problemas surgen si tratamos de reconocer estos lenguajes con un Autómata Finito?

  - Necesitamos "recordar" lo que ya hemos reconocido
  - Almacenamiento ilimitado

- Solución: Los Autómatas de Pila son capaces de reconocer todo lenguaje independiente del contexto.

(estado actual + cima pila + símbolo de entrada)



(nuevo estado actual, nueva pila)

Autómatas de Pila (II)

- Un **Autómata de Pila No Determinista** (ADPND) es una 7-tupla $M=(\Sigma, Q, \Gamma, \Delta, S, F, z)$ donde,
  - Σ es el alfabeto de entrada
  - Q es el conjunto finito de estados
  - Γ es el alfabeto de la pila
  - Δ es la regla de transición

$$\Delta \subseteq Q \times (\Sigma \cup \{\varepsilon\}) \times \Gamma \times Q \times \Gamma^*$$

  - S es el estado inicial o de partida ($S \in Q$)
  - F es el conjunto de estados de aceptación ($F \subseteq Q$)
  - z es el símbolo inicial de la pila ($z \in \Gamma$)

Autómatas de Pila (III)

- Algunas consideraciones:
  - Dado que el autómata de pila es no determinista, Δ es una regla de transición. ¿Cuántos pares resultantes puedo tener para cada terna de entrada?.
  - Podemos tener el par resultante (p, ε). ¿Qué sucede?.
  - De la definición del autómata, siempre se consume un símbolo de la pila. ¿Qué sucede si la pila está vacía?.
  - ¿Qué sucede en esta transición $\Delta(q, \varepsilon, a)=(p, aa)$?
  - ¿Qué sucede en esta transición $\Delta(q, \sigma, \gamma) = \phi$?

Autómatas de Pila (IV)

- Ejemplo: Sea el ADPND,

$\Sigma = \{a, b\}$ $\quad Q = \{q_1, q_2, q_3, q_4\}$

$\Gamma = \{A, B\}$ $\quad z = A$

$F = \{q_4\}$ $\quad s = q_1$

Δ	(a,A)	(b,A)	(ε,A)	(a,B)	(b,B)	(ε,B)
q_1	$\{(q_2,BA), (q_4,A)\}$	ϕ	$\{(q_4, \varepsilon)\}$	ϕ	ϕ	ϕ
q_2	ϕ	ϕ	ϕ	$\{(q_2,BB)\}$	$\{(q3, \varepsilon)\}$	ϕ
q_3	ϕ	ϕ	$\{(q_4,A)\}$	ϕ	$\{(q3, \varepsilon)\}$	ϕ
q_4	ϕ	ϕ	ϕ	ϕ	ϕ	ϕ

¿Qué lenguaje reconoce el autómata de pila?

Autómatas de Pila (V)

- La **Descripción Instantánea** (DI) de un autómata de pila, es una configuración concreta, de las sucesivas, por las que para el autómata durante el procesamiento de una cadena.
- **Notación**: DI :: (q, w, u) donde,

  q es el estado actual

  w es los símbolos de entrada aún no leídos

  u es el contenido de la pila
- El símbolo $\vdash$ indica el movimiento de una DI a otra,

$$(q_1, aw, bx) \vdash (q_2, w, yx)$$

representa el movimiento que resulta de $(q_2,y) \in (q_1, a, b)$

Autómatas de Pila (VI)

- Sea $M=(\Sigma, Q, \Gamma, \Delta, S, F, z)$ un ADPND. El lenguaje aceptado por M se denota L(M) y es el conjunto,

  $L=\{w \in \Sigma^* \mid (S,w,z) \vdash^* (p,\varepsilon,u)$ para $p \in F$ y $u \in \Gamma^*\}$

  Nota: La pila puede acabar vacía o no.

- Para cada lenguaje independiente del contexto existe un ADPND que lo acepta.

  Dado el lenguaje,

  $$L=\{wcw^I \mid w \in \{a,b\}^*\}$$

  ¿Cómo podría construir el ADPND que lo reconozca?

Algoritmo CYK

- Objetivo: Dado un LIC L sobre Σ y $w \in \Sigma^*$, determinar si ¿$w \in L$?.
- *Lema*: Sea $G=(\Sigma,N,S,P)$ una GIC que no tiene ε producciones y está en Forma Normal de Chomsky. Sea $x \in \Sigma^*$, se puede determinar para cada $A \in N$ y para cada subcadena w de x, si $A \overset{*}{\Rightarrow} w$.
  - El objetivo es aplicar el Lema anterior a $S \in N$ y a $w=x$, para determinar si $S \overset{*}{\Rightarrow} x$

Algoritmo CYK (II)

- Algoritmo de CYK (Cocke, Younger, Kasami):

  Puesto que hay muchas subcadenas de x, las nombraremos mediante su posición inicial y su longitud.

  > *Ejemplo*: La w_{ij} es la subcadena de x que comienza en la posición i y tiene una longitud j.

  El algoritmo construye conjunto de no terminales N_{ij} que generan las subcadenas w_{ij} de x. Si $S \in N_{1n}$, entonces $x \in L(G)$

  Nota: Darse cuenta que se elimina mucho trabajo por el hecho de que no pueden existir subcadenas de longitud mayor que n-i+1 que empiecen en la posición i.

  Requisitos: La gramática G es una GIC expresada en Forma Normal de Chomsky.

Algoritmo CYK (III)

- Algoritmo de CYK (Cocke, Younger, Kasami):

1. Para i = 1..n hacer

   $N_{i1} = \{A \mid A \rightarrow w_{i1}\}$

2. Para j = 2..n hacer

   Para i = 1..(n-j+1) hacer

   a. Inicializa $N_{ij} = \phi$

   b. Para K=1..(j-1) añadir a N_{ij} todos los no terminales A para los que $A \rightarrow BC$, con $B \in N_{ik}$ y $C \in N_{i+k,j-k}$

3. Si $S \in N_{1n}$ entonces $x \in L(G)$

Algoritmo CYK (IV)

- Ejemplo: Sea la GIC G,

$S \rightarrow AB \mid BC$

$A \rightarrow BA \mid a$

$B \rightarrow CC \mid b$

$C \rightarrow AB \mid a$

¿pertenece la cadena x=bbab al lenguaje L(G)?