Propiedades de los lenguajes regulares

Curso de preparación en línea para el ingreso a la Maestría en Ciencias Computacionales

Plan

Section 1

Autómatas de Pila

Pushdown Automata

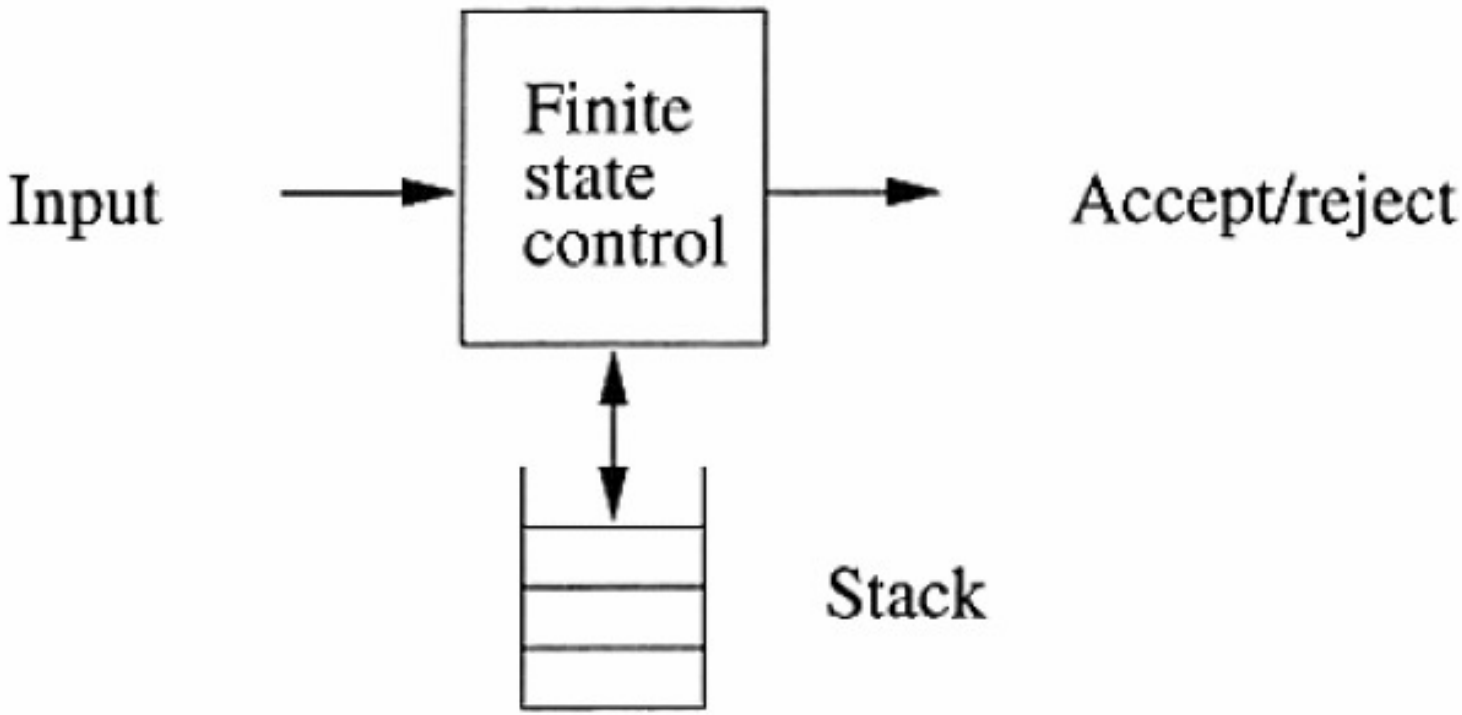
- Las gramáticas libres de contexto tienen un tipo de autómata que las define llamado *pushdown automata*.
- Un *pushdown automata* (PDA) es básicamente un ϵ-NFA con un *stack*, en donde se puede almacenar una cadena y por lo tanto se puede recordar información.
- Sin embargo, sólo puede acceder a esta información en forma LIFO por lo que existen lenguajes reconocidos por una computadora pero no por un PDA, por ejemplo: $\{0^n 1^n 2^n | n \geq 1\}$.

Autómatas No-Deterministas

En una transición el PDA:

1. Consume un símbolo de entrada o hace una transición vacía (ϵ)
2. Se va a un nuevo estado (o se queda en el mismo)
3. Reemplaza el primer elemento del *stack* por alguna cadena (puede ser el mismo símbolo de arriba del *stack*, lo que corresponde con no hacer nada, hace *pop*, lo que corresponde con ϵ, o hace *push* de una cadena al *stack*)

Autómatas No-Deterministas



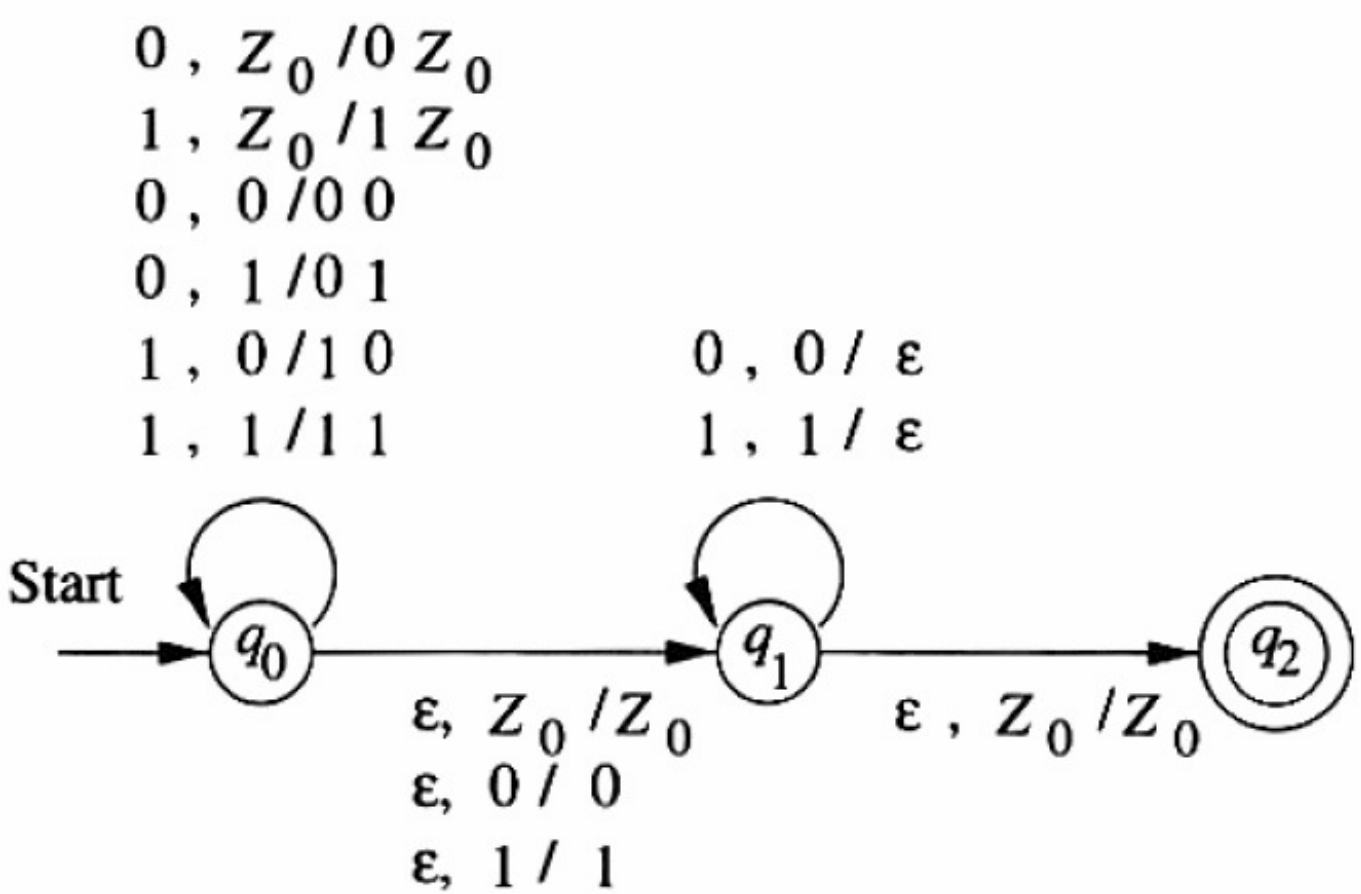
Ejemplo

Sea $L_w = \{ww^R : w \in \{0,1\}^*\}$ con una gramática: $P \to 0P0, P \to 1P1, P \to \epsilon$. Un PDA para L_{ww^r} tiene 3 estados y opera de la siguiente manera:

1. Empieza en q_0 y mientras está ahí lee los símbolos y los almacena en el *stack*.
2. En cualquier momento adivina que está en medio de la cadena (ww^R) y se mueve espontáneamente al estado q_1.
3. En q_1, está leyendo w^R y compara el valor de la cadena con el valor de hasta arriba del *stack*. Si son iguales hace un *pop* y se queda en el estado q_1.
4. Si se vacía el *stack*, se va al estado 2 y acepta.

Autómatas No-Deterministas

El PDA para L_{ww^r} es el siguiente diagrama de transición:



Autómatas No-Deterministas

- Los nodos, nodo inicial y final, son como los hemos visto antes. La diferencia principal es que en las transiciones (arcos) la etiqueta $a, X/\alpha$ significa que $\delta(q, a, X)$ tiene el par (p, α). Osea nos dice la entrada (a) y cómo estaba (X) y cómo queda (α) la parte superior del *stack*.
- Lo único que no nos dice es cuál es el símbolo inicial del *stack*. Por convención se usa Z_0.

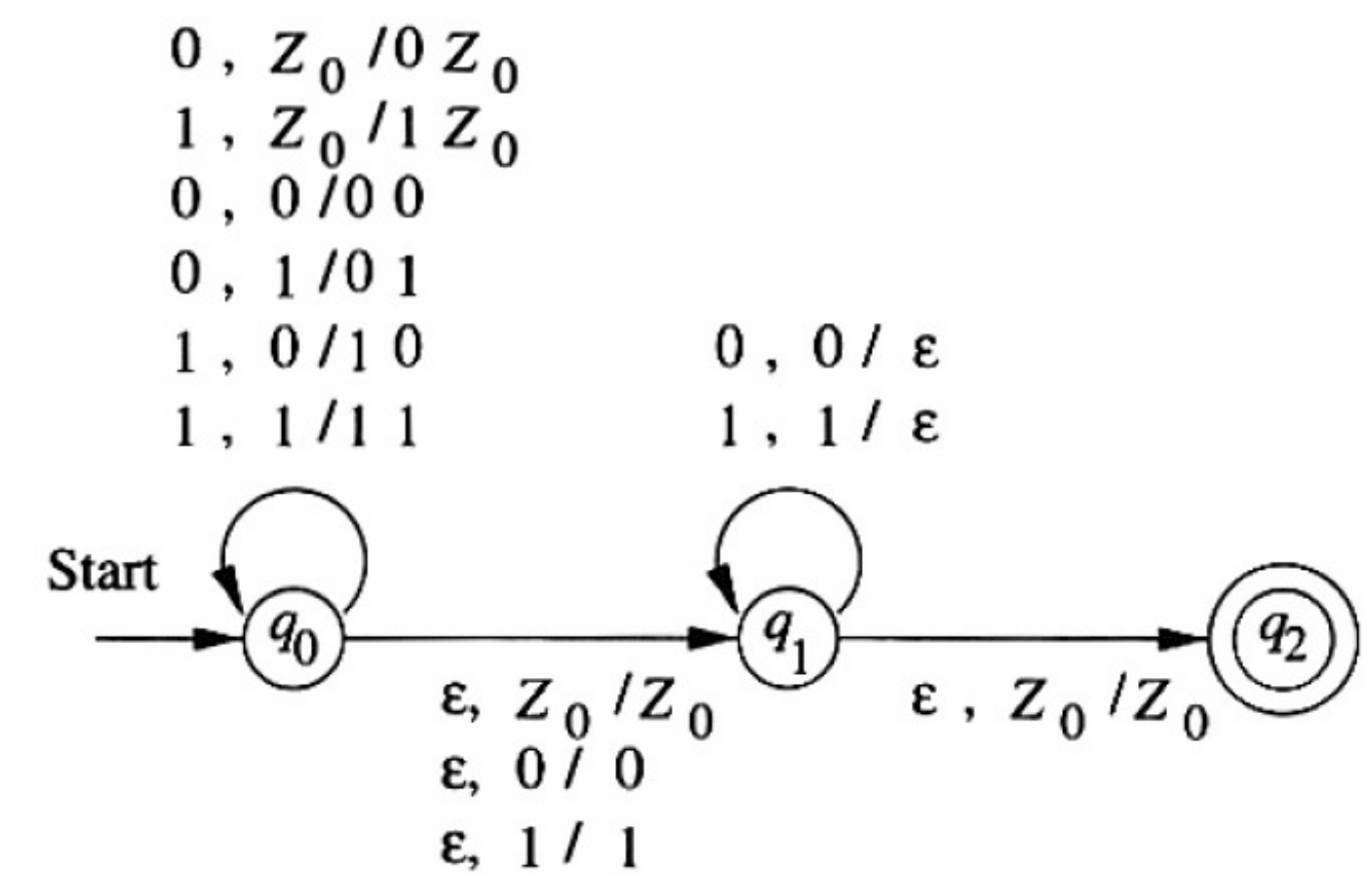
Autómatas No-Deterministas

Formalmente, un PDA es una séptupla: $P = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$, donde:

- Q: es un conjunto finito de estados
- Σ: es un alfabeto finito de entradas
- Γ: es un alfabeto finito del *stack*
- $\delta : Q \times \Sigma \cup \{\epsilon\} \times \Gamma \rightarrow 2^{Q \times \Gamma^*}$ es la función de transición. $\delta(q, a, X) \rightarrow (p, \gamma), \{p, q\} \in Q, a \in \Sigma \cup \{\epsilon\}, X \in \Gamma$ y γ reemplaza a X.
- q_0: es el estado inicial
- $Z_0 \in \Gamma$: es el símbolo inicial del *stack*
- $F \subseteq Q$: es el conjunto de estados finales o de aceptación

Ejemplo

El PDA:



Ejemplo

Es la siguiente tupla:

$P = (\{q_0, q_1, q_2\}, \{0, 1\}, \{0, 1, Z_0\}, \delta, q_0, Z_0, \{q_2\})$.

Donde δ está definida por la siguiente tabla:

	$0, Z_0$	$1, Z_0$	$0, 0$	$0, 1$	$1, 0$
$\rightarrow q_0$	$q_0, 0Z_0$	$q_0, 1Z_0$	$q_0, 00$	$q_0, 01$	$q_0, 10$
q_1			q_1, ϵ		
$*q_2$					

cont...

	$1, 1$	ϵ, Z_0	$\epsilon, 0$	$\epsilon, 1$
$\rightarrow q_0$	$q_0, 11$	q_1, Z_0	$q_1, 0$	$q_1, 1$
q_1		q_1, ϵ	q_2, Z_0	
$*q_2$				

Section 2

Descripciones instantáneas o IDs

Descripciones instantáneas o IDs

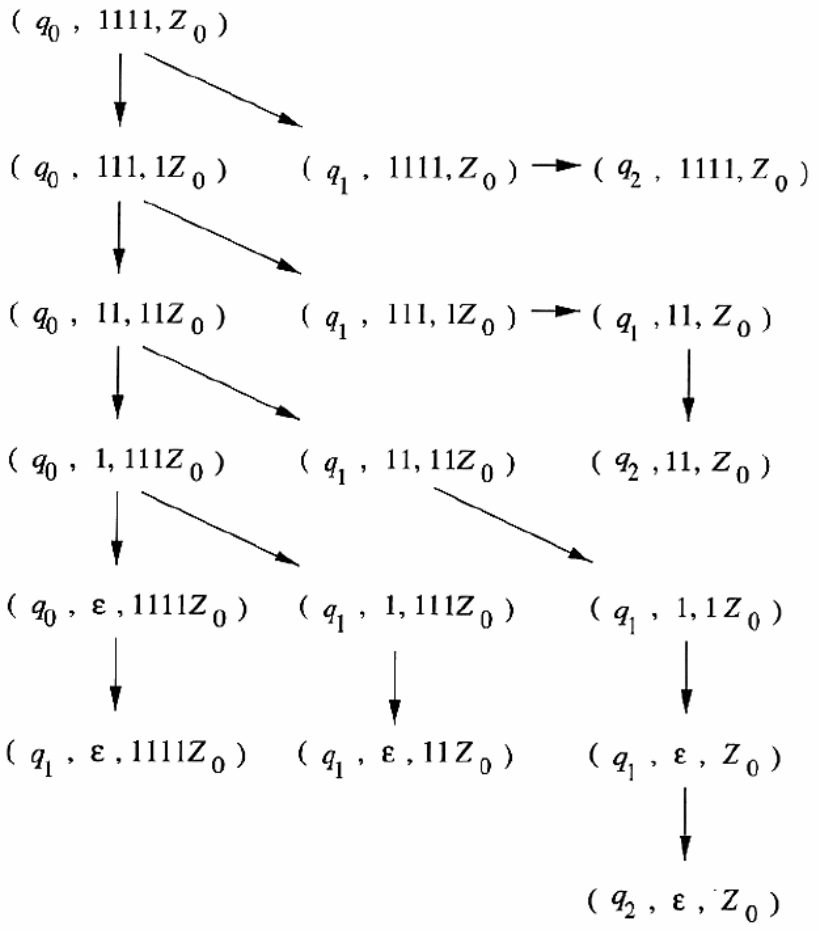
- Una ID es una tripleta (q, w, γ) donde q es un estado, w es lo que falta de la entrada y γ es el contenido del *stack*. Esto es lo que para los FA es el $\hat{\delta}$.
- Usamos $\vdash$ para representar un movimiento en un PDA.
- Sea $P = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ un PDA. Entonces $\forall w \in \Sigma^*, \beta \in \Gamma^*$: $(p, \alpha) \in \delta(q, a, X)$
- $(q, aw, X\beta) \vdash (p, w, \alpha\beta)$. Esto consume a, reemplaza X de arriba del *stack* por α y se va del estado q al estado p.

Cerradura de $\vdash$

- Definimos como la cerradura reflexiva-transitiva de $\vdash$ a $\overset{*}{\vdash}$, para representar cero o más movimientos del PDA.
- $I \overset{*}{\vdash} J$. Si $I \overset{*}{\vdash} J$ y $I \vdash K$, entonces $K \overset{*}{\vdash} J$

Ejemplo

El PDA anterior con la entrada 1111 nos da la siguiente secuencia de transiciones:



Propiedades

Se cumplen las siguientes propiedades. Si una secuencia de ID (computación) es legal para un PDA:

1. Entonces también es legal la secuencia que se obtiene al añadir una cadena igual al final de la entrada de cada ID (segundo componente).
2. Entonces también es legal la secuencia que se obtiene al añadir una cadena igual hasta abajo del *stack* en cada ID (tercer componente).
3. Y si el final de una entrada no es consumida, al eliminar ese final de todos los ID's también resulta en una secuencia legal.

Autómatas No-Deterministas

Teoremas: $\forall w \in \Sigma^*, \gamma \in \Gamma^*$
$(q, x, \alpha) \overset{*}{\vdash} (p, y, \beta) \Rightarrow (q, xw, \alpha\gamma) \overset{*}{\vdash} (p, yw, \beta\gamma)$ que cubre el primer principio si $\gamma = \epsilon$ y el segundo si $w = \epsilon$, y
$(q, xw, \alpha) \overset{*}{\vdash} (p, yw, \beta) \Rightarrow (q, x, \alpha) \overset{*}{\vdash} (p, y, \beta)$ que cubre el tercer principio.

Ejemplo

Suponga que el PDA $P = (\{q, p\}, \{0, 1\}, \{Z_0, X\}, \delta, q, Z_0, \{p\})$ tiene la siguiente función de transición:

$\delta(q, 0, Z_0) = \{(q, XZ_0)\}$
$\delta(q, 0, X) = \{(q, XX)\}$
$\delta(q, 1, X) = \{(q, X)\}$
$\delta(q, \epsilon, X) = \{(p, \epsilon)\}$
$\delta(p, \epsilon, X) = \{(p, \epsilon)\}$
$\delta(p, 1, X) = \{(p, XX)\}$
$\delta(p, 1, Z_0) = \{(p, \epsilon)\}$

Empezando del ID inicial (q, w, Z_0) muestre todos los ID's alcanzables cuando w es: 01

$(q, 01, Z_0) \vdash (q, 1, XZ_0) \vdash (q, \epsilon, XZ_0) \vdash (p, \epsilon, Z_0) \vdash (p, 1, Z_0) \vdash (p, \epsilon, \epsilon)$

Section 3

El Lenguaje de PDA

El Lenguaje de PDA

Existen dos formas equivalentes de PDA que aceptan un cierto lenguaje L:

1. Aceptar por el estado final:

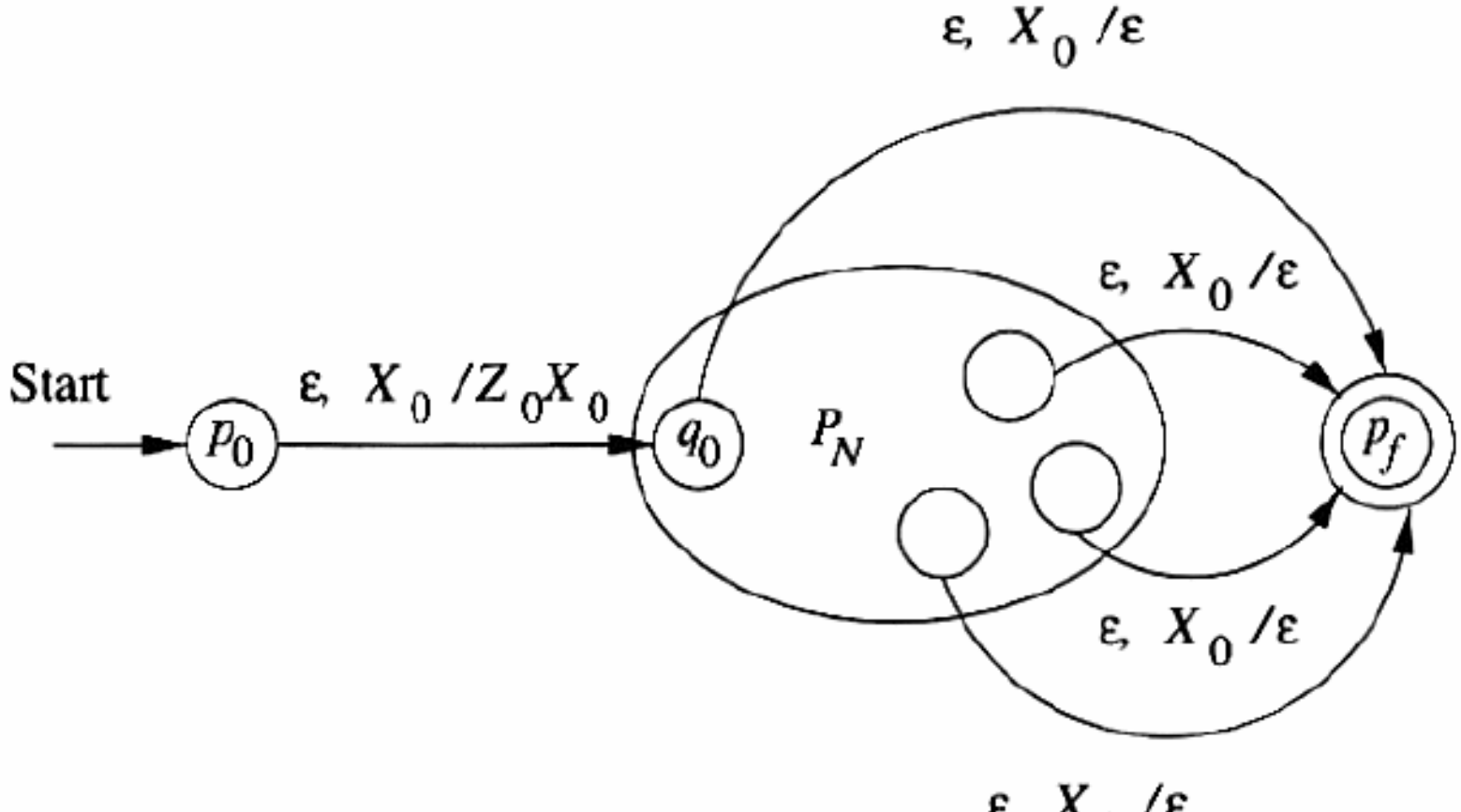
   $L(P) = \{w : (q_0, w, Z_0) \overset{*}{\vdash} (q, \epsilon, \alpha), q \in F\}$. Por ejemplo, podemos ver una secuencia legal para aceptar por estado final un palíndrome: $(q_0, ww^R, Z_0) \overset{*}{\vdash} (q_0, w^R, w^R Z_0) \vdash$ $(q_1, w^R, w^R Z_0) \overset{*}{\vdash} (q_1, \epsilon, Z_0) \vdash (q_2, \epsilon, Z_0)$
2. Aceptar por el *stack* vacío:

   $N(P) = \{w : (q_0, w, Z_0) \overset{*}{\vdash} (q, \epsilon, \epsilon)\}$. El ejemplo anterior no acepta por *stack* vacío, pero con una pequeña modificación lo puede hacer si en lugar de $\delta(q, \epsilon, Z_0) = \{(q_2, Z_0)\}$ ponemos $\delta(q, \epsilon, Z_0) = \{(q_2, \epsilon)\}$

Las dos representaciones son equivalentes y se puede pasar de una a la otra

De *stack* vacío a estado final

Se puede mostrar que si $L = N(P_N)$ para un PDA P_N, entonces existe un PDA P_F tal que $L = L(P_F)$. Se ilustra en la siguiente figura:



De *stack* vacío a estado final

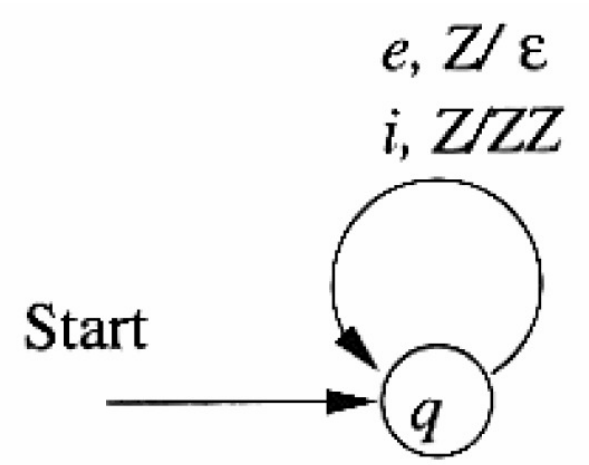
- La idea, es usar un símbolo nuevo ($X_0 \notin \Gamma$) como marca para señalizar el final del *stack* y un nuevo estado cuyo único objetivo es poner Z_0 arriba de este símbolo especial.
- Después se simula la misma transición de estados hasta que se vacíe el *stack*.
- Finalmente, añadimos un nuevo estado que es el de aceptación al que se llega cada vez que se vacía el *stack*.
- También necesitamos un nuevo estado inicial (p_0) que lo único que hace es hacer un *push* del símbolo inicial Z_0 e irse al estado inicial de $P_N(q_0)$.

De *stack* vacío a estado final

Lo anterior lo podemos expresar como sigue:

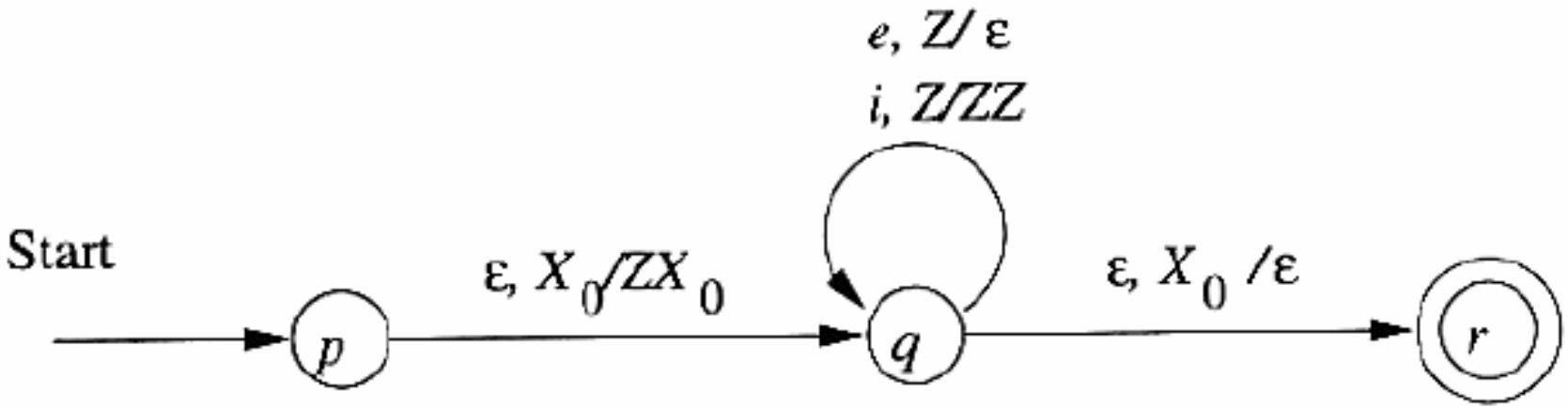
$(q_0, w, X_0) \vdash_F (q_0, w, Z_0X_0) \overset{*}{\vdash}_F (q, \epsilon, X_0) \vdash_F (p_f, \epsilon, \epsilon)$

Ejemplo: podemos pasar del siguiente PDA que acepta por *stack* vacío a un PDA que acepta por estado final:



Ejemplo (cont.)

$P_N = (\{q\}, \{i, e\}, \{Z\}, \delta_N, q, Z)$, donde δ_N es:
$\delta_N(q, i, Z) = \{(q, ZZ)\}$
$\delta_N(q, e, Z) = \{(q, \epsilon)\}$

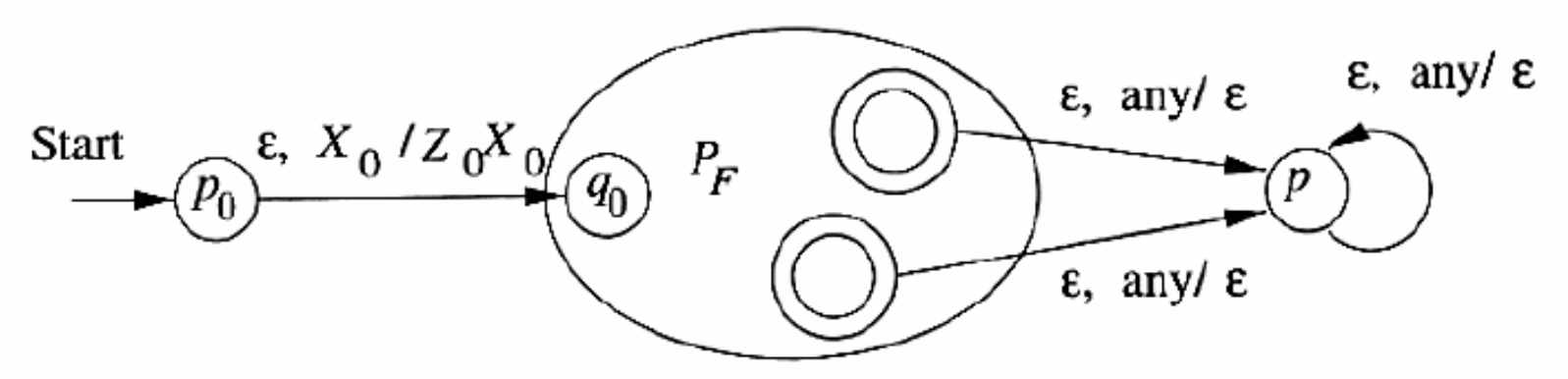


Ejemplo (cont.)

$P_F = (\{p, q, r\}, \{i, e\}, \{Z, X_0\}, \delta_F, p, X_0, r)$, donde δ_F es:
$\delta_F(p, \epsilon, X0) = \{(q, ZX_0)\}$
$\delta_F(q, i, Z) = \{(q, ZZ)\}$
$\delta_F(q, e, Z) = \{(q, \epsilon)\}$
$\delta_F(q, \epsilon, X_0) = \{(r, \epsilon)\}$

De un estado final a un *stack* vacío

Se puede mostrar que si $L = L(P_F)$ para un PDA P_F, entonces existe un PDA P_N tal que $L = N(P_N)$. Se ilustra en la siguiente figura:



De un estado final a un *stack* vacío

- La idea es la siguiente: Cada vez que se llega a un estado final, se hace una transición vacía a un nuevo estado en donde se vacía el *stack* sin consumir símbolos de entrada.
- Para evitar simular una situación en donde se vacía el *stack* sin aceptar la cadena, de nuevo se introduce al principio un nuevo símbolo al *stack*.
- Lo anterior lo podemos expresar como sigue:

$$(p_0, w, X_0) \vdash_N (q_0, w, Z_0X_0) \overset{*}{\vdash}_N (q, \epsilon, \alpha X_0) \overset{*}{\vdash}_N (p, \epsilon, \epsilon)$$

Ejemplos

- Diseñe PDAs que acepten los siguientes lenguajes:
  1. $\{0^n 1^n | n \geq 1\}$
  2. $\{a^i b^j c^k | i = j \text{ o } j = k\}$
- Para el siguiente PDA genere las trazas de IDs para la cadena

$$\begin{array}{ll} \delta(q_0, a, Z_0) = (q_1, AAZ_0) & \delta(q_0, b, Z_0) = (q_2, BZ_0) \\ \delta(q_0, \epsilon, Z_0) = (f, \epsilon) & \delta(q_1, a, A) = (q_1, AAA) \\ \delta(q_1, b, A) = (q_1, \epsilon) & \delta(q_1, \epsilon, Z_0) = (q_0, Z_0) \\ \delta(q_1, a, B) = (q_3, \epsilon) & \delta(q_2, b, B) = (q_2, BB) \\ \delta(q_2, \epsilon, Z_0) = (q_0, Z_0) & \delta(q_3, \epsilon, B) = (q_2, \epsilon) \\ \delta(q_3, \epsilon, Z_0) = (q_1, AZ_0) & \end{array}$$

bab:

Soluciones

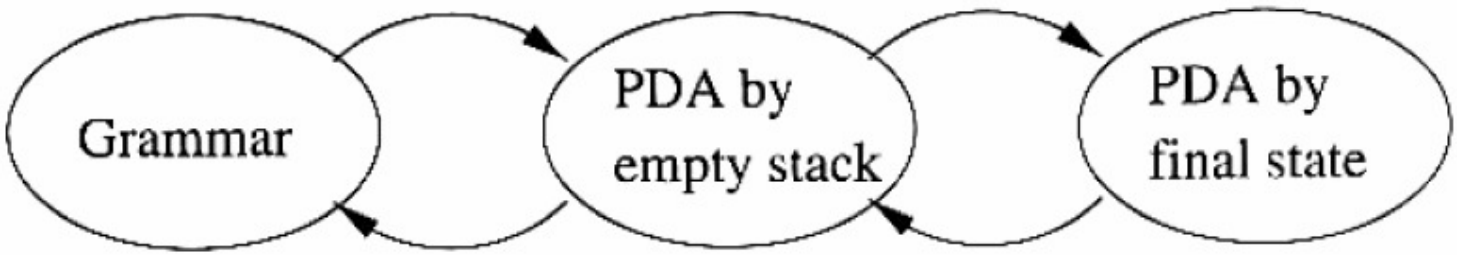
- Aceptando por stack vacío:
  $\delta(q,0,Z_0) = \{(q,X)\}$ $\delta(q,0,X) = \{(q,XX)\}$
  $\delta(q,1,X) = \{(p,\epsilon)\}$ $\delta(q,1,X) = \{(p,\epsilon)\}$
- Adivinar: (i) $i = j = 0$ (q_1), (ii) $i = j > 0$ (q_2) y (iii) $j = k$ (q_3):
  $\delta(q_0,\epsilon,Z_0) = \{(q_1,Z_0),(q_2,Z_0),(q_3,Z_0)\}$
  $\delta(q_1,c,Z_0) = \{(q_1,Z_0)\}$
  $\delta(q_2,a,Z_0) = \{(q_2,XZ_0)\}$ y $\delta(q_2,a,X) = \{(q_2,XX)\}$
  $\delta(q_2,b,X) = \delta(q_4,b,X) = \{(q_4,\epsilon)\}$
  $\delta(q_3,a,Z_0) = \{(q_3,Z_0)\}$
  $\delta(q_3,b,Z_0) = \{(q_5,XZ_0)\}$
  $\delta(q_5,b,X) = \{(q_5,XX)\}$
  $\delta(q_5,c,X) = \delta(q_6,c,X) = \{(q_6,\epsilon)\}$
  $\delta(q_6,\epsilon,Z_0) = \{(q_3,\epsilon)\}$
- $(q_0,bab,Z_0) \vdash (q_2,ab,BZ_0) \vdash (q_3,b,Z_0) \vdash (q_1,b.AZ_0) \vdash$
  $(q_1,b,AZ_0) \vdash (q_1,\epsilon,Z_0) \vdash (q_0,\epsilon,Z_0) \vdash (f,\epsilon,\epsilon)$

Section 4

Equivalencia entre PDAs y CFGs

Equivalencia entre PDAs y CFGs

- Los lenguajes definidos por los PDA son los lenguajes libres de contexto.
- Un lenguaje es generado por un CFG si y solo si es aceptado por un PDA con *stack* vacío si y solo si es aceptado por un PDA con estado final.



La última parte ya la sabemos y sólo nos falta demostrar lo primero.

De un CFG a un PDA

- Dada una gramática G vamos a construir un PDA que simula $\Rightarrow^*_{lm}$
- Cualquier forma de sentencia izquierda que no es una cadena terminal la podemos escribir como $xA\alpha$ donde A es la variable más a la izquierda, x son los símbolos terminales que aparecen a la izquierda, y α es la cadena de símbolos terminales y variables que aparecen a la derecha de A.
- La idea para construir un PDA a partir de una gramática, es que el PDA simula las formas de sentencia izquierdas que usa la gramática para generar una cadena w.

De un CFG a un PDA

- $A\alpha$ que es la "cola" de la forma de sentencia izquierda va a aparecer en el *stack* con A como primer elemento.
- Sea $xA\alpha \Rightarrow_{lm} x\beta\alpha$. Usa o adivina que se usa la producción $A \to \beta$.
- Esto corresponde a un PDA que consume primero a x con $A\alpha$ en el *stack* y luego con ϵ saca (*pops*) A y mete (*pushes*) β.
- O de otra forma, sea $w = xy$, entonces el PDA va en forma no-determinísta de la configuración $(q, y, A\alpha)$ a la configuración $(q, y, \beta\alpha)$.

De un CFG a un PDA

- En $(q, y, \beta\alpha)$ el PDA se comporta como antes, a menos que sean símbolos terminales en el prefijo de β, en cuyo caso el PDA los saca (*pops*) del *stack*, si puede consumir símbolos de entrada.
- Si todo se adivina bien, el PDA acaba con un *stack* vacío y con la cadena de entrada. Nótese que el PDA tiene un solo estado.
- Formalmente, sea $G = (V, T, Q, S)$ un CFG. Se define un P_G como $(\{q\}, T, V \cup T, \delta, q, S)$ donde:
  $\delta(q, \epsilon, A) = \{(q, \beta) : A \to \beta \in Q\}$ para $A \in V$, y
  $\delta(q, a, a) = \{(q, \epsilon)\}$ para $a \in T$.

Ejemplo

- Convirtamos la siguiente gramática en un PDA:
  $I \to a|b|Ia|Ib|I0|I1$
  $E \to I|E*E|E+E|(E)$
- Los símbolos terminales del PDA son: $\{a, b, 0, 1, (,), +, *\}$
- La función de transición del PDA es:
  $\delta(q, \epsilon, I) = \{(q, a), (q, b), (q, Ia), (q, Ib), (q, I0), (q, I1)\}$
  $\delta(q, \epsilon, E) = \{(q, I), (q, E+E), (q, E*E), (q, (E))\}$
  $\delta(q, a, a) = \{(q, \epsilon)\}, \delta(q, b, b) = \{(q, \epsilon)\},$
  $\delta(q, 0, 0) = \{(q, \epsilon)\}, \delta(q, 1, 1) = \{(q, \epsilon)\},$
  $\delta(q, (, () = \{(q, \epsilon)\}, \delta(q,),)) = \{(q, \epsilon)\},$
  $\delta(q, +, +) = \{(q, \epsilon)\}, \delta(q, *, *) = \{(q, \epsilon)\}.$

Teorema: $N(P) = L(G)$

- Sea $w \in L(G)$, entonces, $S = \gamma_1 \Rightarrow_{lm} \gamma_2 \Rightarrow_{lm} \cdots \gamma_n = w$
- Sea $\gamma_i = x_i \alpha_i$. Vamos a probar por inducción en i que si: $S \overset{*}{\Rightarrow}_{lm} \gamma_i$, entonces: $(q, w, S) \overset{*}{\vdash} (q, y_i, \alpha_i)$, donde $w = x_i y_i$. El caso base es fácil, donde $i = 1, \gamma_1 = \epsilon$, y $y_1 = w$.

Teorema: $N(P) = L(G)$

- Dado $(q, w, S) \overset{*}{\vdash} (q, y_i, \alpha_i)$ queremos probar para el siguiente paso, o sea: $(q, y_i, \alpha_i) \vdash (q, y_{i+1}, \alpha_{i+1})$
- α_i empieza con una variable A y tenemos lo siguiente: $\underbrace{x_i A \chi}_{\gamma_i} \Rightarrow_{lm} \underbrace{x_{i+1} \beta \chi}_{\gamma_{i+1}}$
- Por la hipótesis inductiva, $A\chi$ está en el *stack* y y_i no ha sido consumida. De la construcción de P_G se sigue que podemos hacer el siguiente movimiento: $(q, y_i, A\chi) \vdash (q, y_i, \beta\chi)$.

Teorema: $N(P) = L(G)$

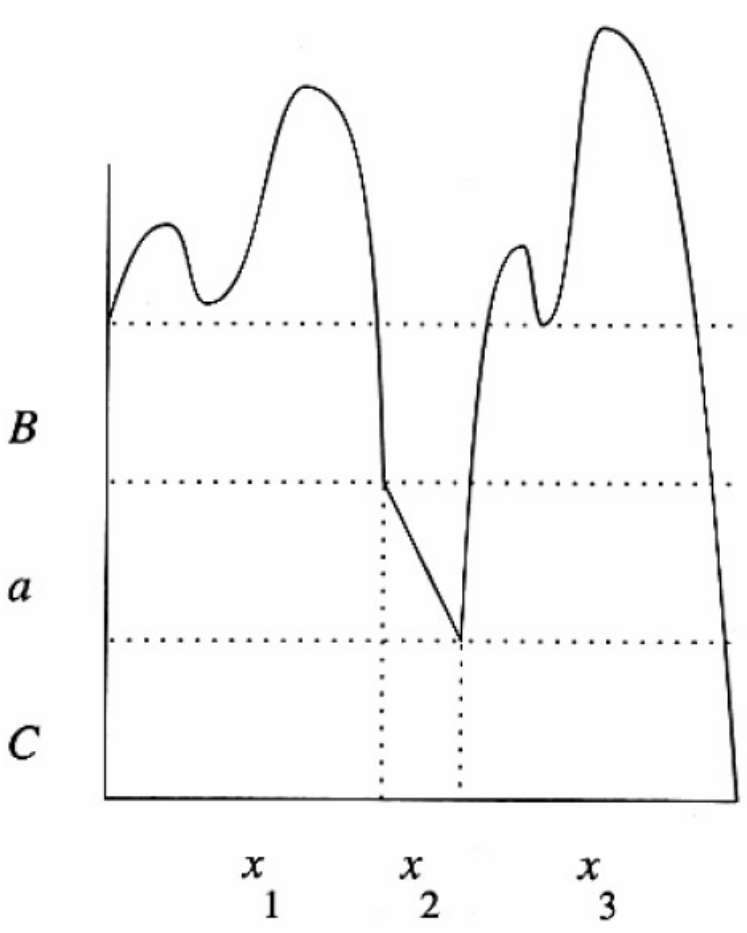
- Si β tiene un prefijo de terminales, podemos sacarlos (*pop*) con terminales correspondientes en un prefijo de y_i, acabando con la configuración $(q, y_{i+1}, \alpha_{i+1})$, donde $\alpha_{i+1} = \beta\chi$ que es la parte final de: $x_{i+1}\beta\chi = \gamma_{i+1}$.
- Finalmente, como $\gamma_n = w$, tenemos que $\alpha_n = \epsilon$, y $y_n = \epsilon$, y por lo tanto: $(q, w, S) \overset{*}{\vdash} (q, \epsilon, \epsilon)$, por lo que $w \in N(P_G)$, osea que P acepta w con *stack* vacío.
- Para la otra parte de la prueba, tenemos que probar que si: $(q, x, A) \overset{*}{\vdash} (q, \epsilon, \epsilon)$ entonces: $A \overset{*}{\Rightarrow} x$.

Teorema: $N(P) = L(G)$

- El caso base es simple, con $x = \epsilon$.
- Para el caso inductivo, tenemos que, como A es una variable, tenemos en general:
  $(q, x, A) \vdash (q, x, Y_1 Y_2 \ldots Y_k) \vdash \ldots \vdash (q, \epsilon, \epsilon)$
  donde $A \to Y_1 Y_2 \ldots Y_k$ está en G.
- Si escribimos x como $x_1 x_2 \ldots x_k$, x_1 es la parte de la entrada que es consumida hasta que Y_1 es sacada (*popped*) del *stack*.
- Luego consumimos x_2 sacando a Y_2 del *stack*, y así sucesivamente.

Teorema: $N(P) = L(G)$

Podemos imaginarnoslo de la siguiente figura con $Y_1 = B$, $Y_2 = a$ y $Y_3 = C$.

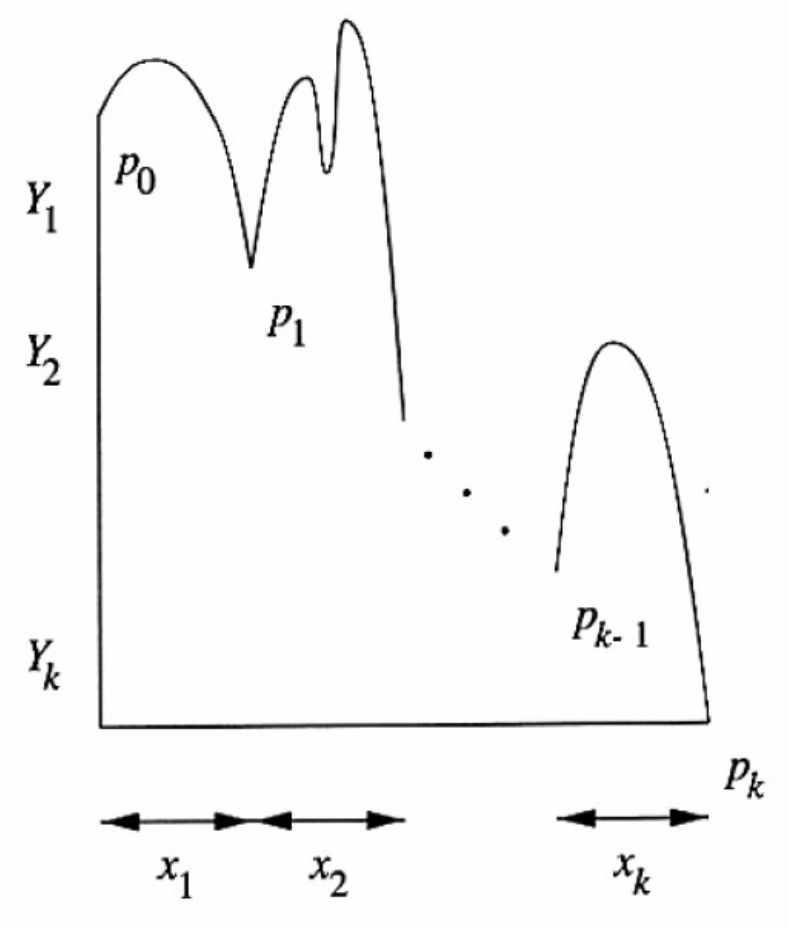


Teorema: $N(P) = L(G)$

- Aquí x es dividió en 3 partes.
  1. La hipótesis inductiva nos dice que si $(q, x, A) \overset{*}{\vdash} (q, \epsilon, \epsilon)$ entonces $A \overset{*}{\Rightarrow} x$.
  2. Si ahora tenemos una x que la podemos descomponer en varias x_i, entonces podemos aplicar la hipótesis inductiva a cada una de las partes y demostrar que:
     $A \Rightarrow Y_1 Y_2 \ldots Y_k \overset{*}{\Rightarrow} x_1 Y_2 \ldots Y_k \overset{*}{\Rightarrow} \ldots \overset{*}{\Rightarrow} x_1 x_2 \ldots x_k$.
  3. Para completar la prueba suponemos que $A = S$ y que $x = w$.

De PDA's a Gramáticas

Se puede también demostrar como pasar de PDA's a CFG's. Osea como consumir una cadena $x = x_1 x_2 \ldots x_n$ y vaciar el *stack*.



De PDA's a Gramáticas

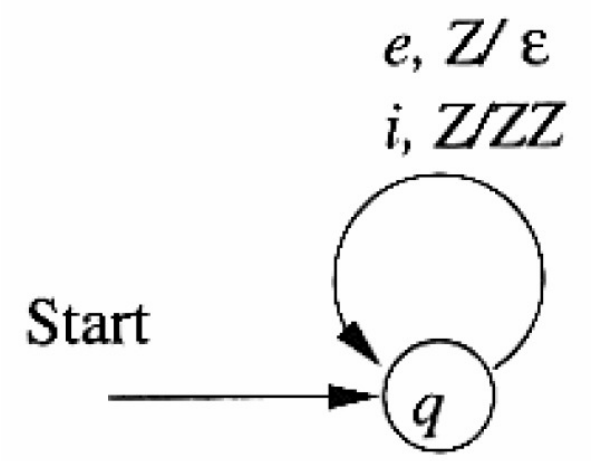
- La idea es definir una gramática con variables de la forma $[p_{i-1} Y_i p_i]$ que representan ir de p_{i-1} a p_i haciendo un *pop* de Y_i.
- Formalmente, sea $P = (Q, \Sigma, \Gamma, \delta, q_0, Z_0)$ un PDA. Definimos una gramática $G = (V, \Sigma, R, S)$, donde:
  $V = \{[pXq] : \{p, q\} \subseteq Q, X \in \Gamma\} \cup \{S\}$
  $R = \{S \to [q_0 Z_0 p] : p \in Q \cup \{[qXr_k] \to$
  $a[rY_1 r_1] \ldots [r_{k-1} Y_k r_k] : a \in \Sigma \cup \{\epsilon\}, \{r_1, \ldots, r_k\} \subseteq$
  $Q, (r, Y_1 Y_2 \ldots Y_k) \in \delta(q, a, X)\}$

De PDA's a Gramáticas

- Osea las variables son el símbolo inicial S junto con todos los símbolos formados por pXq donde p y q son estados de Q y X es un símbolo del *stack*.
- Las producciones se forman primero para todos los estados p, G tiene la producción $S \rightarrow [q_0 Z_0 p]$, que genera todas las cadenas w que causan a P sacar (pop) a Z_0 del *stack* mientras se va del estado q_0 al estado p.
- Esto es $(q_0, w, Z_0) \overset{*}{\vdash} (p, \epsilon, \epsilon)$.
- Por otro lado, se tienen las producciones que dicen que una forma de sacar (pop) a X e ir de un estado q a un estado r_k, es leer a a (que puede ser ϵ), usar algo de entrada para sacar (pop) a Y_1 del *stack*, mientras vamos del estado r al estado r_1, leer más entrada que saca Y_2 del *stack* y así sucesivamente.

Ejemplo

Supongamos el siguiente PDA:



De PDA's a Gramáticas

- $P_N = (\{q\}, \{i, e\}, Z, \delta_N, q, Z)$, donde $\delta_N(q, i, Z) = \{(q, ZZ)\}$, y $\delta_N(q, e, Z) = \{(q, \epsilon)\}$.
- Podemos definir la siguiente gramática equivalente: $G = (V, \{i, e\}, R, S)$, donde $V = \{[qZq], S\}$ y $R = \{S \to [qZq], [qZq] \to i[qZq][qZq], [qZq] \to e\}$.
- Asumimos que S es el símbolo de entrada para toda gramática. $[qZq]$ es la única tripleta que podemos formar con símbolos de entrada y símbolos del *stack*.
- A partir de S la única producción entonces que tenemos es: $S \to [qZq]$.
- Debido que tenemos la transición: $\delta_N(q, i, Z) = \{(q, ZZ)\}$ generamos la producción: $[qZq] \to i[qZq][qZq]$, y con $\delta_N(q, e, Z) = \{(q, \epsilon)\}$, generamos la producción: $[qZq] \to e$.

De PDA's a Gramáticas

- Si remplazamos a $[qZq]$ por A, nos queda: $S \rightarrow A$ y $A \rightarrow iAA|e$. De hecho podemos poner simplemente $S \rightarrow iSS|e$.
- **Ejemplo2:** Sea $P_N = (\{p, q\}, \{0, 1\}, \{X, Z_0\}, \delta, q, Z_0)$, donde δ está dada por:
  $\delta(q, 1, Z_0) = \{(q, XZ_0)\}$
  $\delta(q, 1, X) = \{(q, XX)\}$
  $\delta(q, 0, X) = \{(p, X)\}$
  $\delta(q, \epsilon, X) = \{(q, \epsilon)\}$
  $\delta(p, 1, X) = \{(p, \epsilon)\}$
  $\delta(p, 0, Z_0) = \{(q, Z_0)\}$

Ejemplo (cont.)

El CFG equivalente es: $G(V, \{0,1\}, R, S)$, donde:
$V = \{[pXp], [pXq], [pZ_0p], [pZ_0q], [qXq], [qXp], [qZ_0q], [qZ_0p], S\}$
y las reglas de producción R son:

- $S \rightarrow [qZ_0q] | [qZ_0p]$
- De la primera regla de transición:
  $[qZ_0q] \rightarrow 1[qXq], [qZ_0q]$
  $[qZ_0q] \rightarrow 1[qXp], [pZ_0q]$
  $[qZ_0p] \rightarrow 1[qXq], [qZ_0p]$
  $[qZ_0p] \rightarrow 1[qXp], [pZ_0q]$
- De la 2:
  $[qXq] \rightarrow 1[qXq], [qXq]$
  $[qXq] \rightarrow 1[qXp], [pXq]$
  $[qXp] \rightarrow 1[qXq], [qXp]$
  $[qXp] \rightarrow 1[qXp], [pXp]$

Ejemplo (cont.)

- De la 3:
  $[qXq] \to 0[pXq]$
  $[qXp] \to 0[pXp]$
- De la 4:
  $[qXq] \to \epsilon$
- De la 5:
  $[pXp] \to 1$
- De la 6:
  $[pZ_0q] \to 0[qZ_0q]$
  $[pZ_0p] \to 0[qZ_0p]$

De PDA's a Gramáticas

- Se puede probar que si G se construye como arriba a partir de un PDA P, entonces, $L(G) = N(P)$.
- La prueba se hace por inducción sobre las derivaciones. Se quiere probar que: Si $(q, w, X) \overset{*}{\vdash} (p, \epsilon, \epsilon)$ entonces: $[qXp] \overset{*}{\Rightarrow} w$
- El caso base, es sencillo cuando w es a o ϵ y $(p, \epsilon) \in \delta(q, w, X)$, por lo que por construcción de G tenemos $[qXp] \to w$ y por lo tanto $[qXp] \overset{*}{\Rightarrow} w$.

De PDA's a Gramáticas

- Para la parte de inducción tenemos que: $(q, w, X) \vdash (r_0, x, Y_1 Y_2 \dots Y_k) \vdash \dots \vdash (p, \epsilon, \epsilon)$. Donde $w = ax$ o $w = \epsilon x$, por lo que $(r_0, x, Y_1 Y_2 \dots Y_k) \in \delta(q, a, X)$.
- Entonces tenemos la siguiente producción: $[qXr_k] \to a[r_0 Y_1 r_1] \dots [r_{k-1} Y_k r_k]$ para toda $\{r_1, \dots, r_k\} \subset Q$.
- Podemos escoger r_i ser el estado en la secuencia cuando Y_i sale del *stack* (*popped*). Sea $w = w_1 w_2 \dots w_k$, donde w_i es consumido cuando Y_i es sacado (*popped*).
- Entonces: $(r_{i-1}, w_i, Y_i) \overset{*}{\vdash} (r_i, \epsilon, \epsilon)$

De PDA's a Gramáticas

- Por la hipótesis de inducción tenemos: $[r_{i-1}, Y, r_i] \stackrel{*}{\Rightarrow} w_i$.
- Por lo que obtenemos la siguiente secuencia de derivación:
  $[qXr_k] \Rightarrow a[r_0 Y_1 r_1] \ldots [r_{k-1} Y_k r_k] \stackrel{*}{\Rightarrow}$
  $aw_1[r_1 Y_2 r_2][r_2 Y_3 r_3] \ldots [r_{k-1} Y_k r_k] \stackrel{*}{\Rightarrow}$
  $aw_1 w_2[r_2 Y_3 r_3] \ldots [r_{k-1} Y_k r_k] \stackrel{*}{\Rightarrow}$
  $\ldots$
  $aw_1 w_2 \ldots w_k = w$
- Para el *only-if* queremos probar que si: $[qXp] \stackrel{*}{\Rightarrow} w$ entonces:
  $(q, w, X) \stackrel{*}{\vdash} (p, \epsilon, \epsilon)$.
- La prueba sigue las mismas ideas que hemos estado viendo.

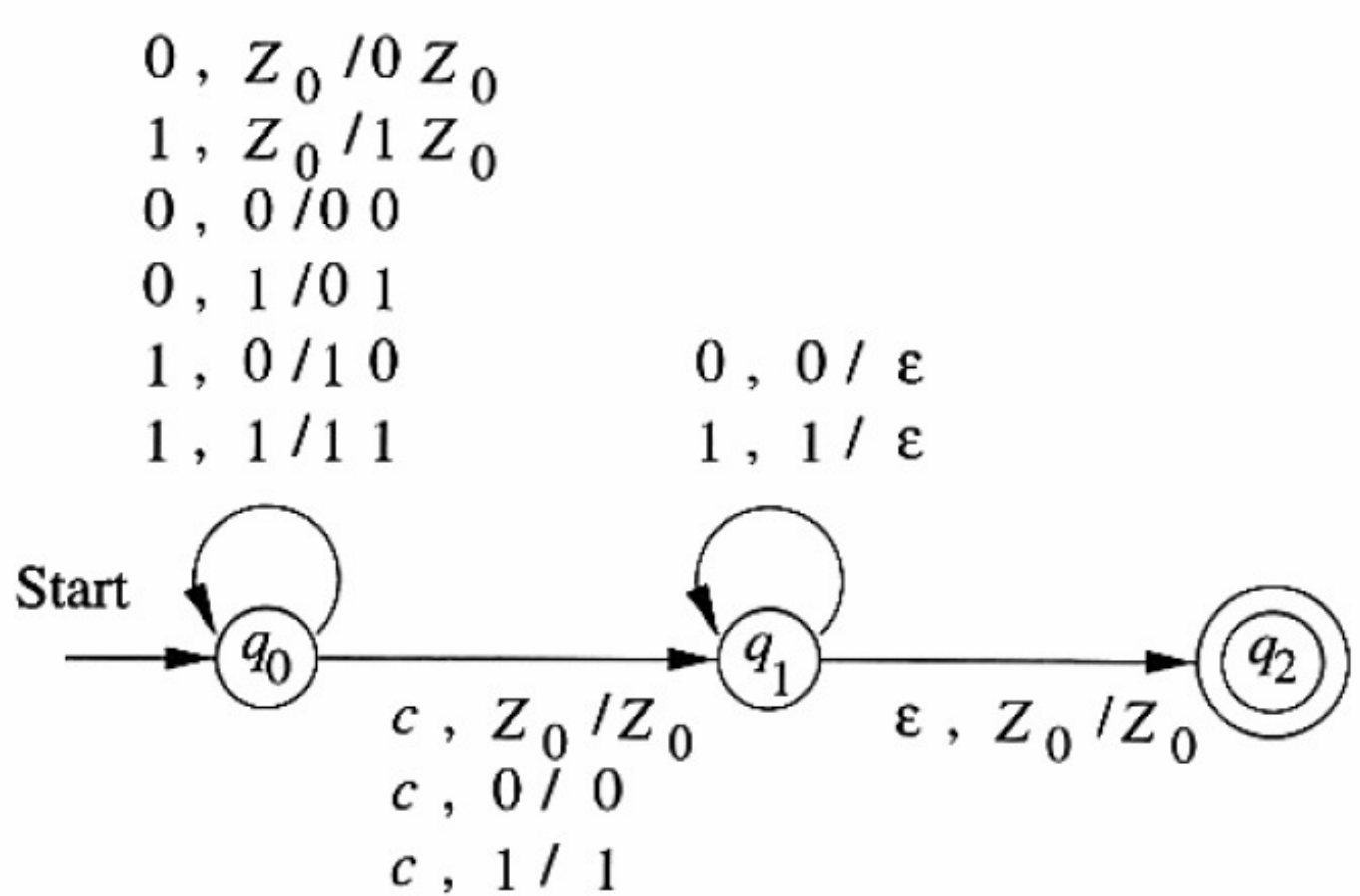
PDA's determinísticos

Un PDA $P = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ es *determinístico* si y solo si:

1. $\delta(q, a, X)$ es siempre vacío o es un *singleton*.
2. Si $\delta(q, a, X)$ no es vacío, entonces $\delta(q, \epsilon, X)$ debe de ser vacío.

Ejemplo

Sea $L_{wcw^r} = \{wcw^R : w \in \{0,1\}^*\}$. Entonces L_{wcw^r} es reconocido por el siguiente DPDA:



Ejemplo

Covertir el siguiente PDA a un CFG:

1. $\delta(q, 1, Z_0) = \{(q, XZ_0)\}$
2. $\delta(q, 1, X) = \{(q, XX)\}$
3. $\delta(q, 0, X) = \{(p, X)\}$
4. $\delta(q, \epsilon, X) = \{(q, \epsilon)\}$
5. $\delta(p, 1, X) = \{(p, \epsilon)\}$
6. $\delta(p, 0, Z_0) = \{(q, Z_0)\}$

Ejemplo (cont.)

$S \to [qZq]|[qZp]$ (S símbolo de entrada, e cadena vacía, y Z en lugar de Z_0)

De la regla (1):

$[qZq] \to 1[qZq][qZq]$
$[qZq] \to 1[qZp][pZq]$
$[qZp] \to 1[qZq][qZp]$
$[qZp] \to 1[qZp][pZq]$

De la regla (2):

$[qXq] \to 1[qXq][qXq]$
$[qXq] \to 1[qXp][pXq]$
$[qXp] \to 1[qXq][qXp]$
$[qXp] \to 1[qXp][pXq]$

De la regla (3):

$[qXq] \to 0[pXq]$
$[qXp] \to 0[pXp]$

Ejemplo (cont.)

De la regla (4):
$[qZq] \rightarrow \epsilon$
De la regla (5):
$[pXp] \rightarrow 1$
De la regla (6):
$[pZq] \rightarrow 0[qZq]$
$[pZp] \rightarrow 0[qZp]$

PDA's determinísticos

- Se puede demostrar que $RE \subset L(DPDA) \subset CFL$.
- La primera parte es fácil. Si es RE se puede construir un DFA, por lo que construir un DPDA es trivial a partir de este. De hecho podemos ignorar el *stack*.

Algunas propiedades

- $RE \subset L(DPDA) \subset CFL$
- Por ejemplo: L_{wcw^r} es reconocido por $L(DPDA)$ pero no por RE.
- L_{ww^r} es reconocido por un lenguaje de un CFG pero no por $L(DPDA)$.
- Si $L = L(P)$ para algún DPDA P, entonces L tiene un CFG no ambiguo.

Es el PDA del primer ejemplo (acetato 17) determinista?
Material basado en el libro Introduction to Automata Theory Languages and Computation [Hopcroft et al., 2006].

Bibliography I


Hopcroft, J. E., Motwani, R., and Ullman, J. D. (2006).
Automata theory, languages, and computation.
International Edition, 24.