# Autómatas de Pila

INAOE

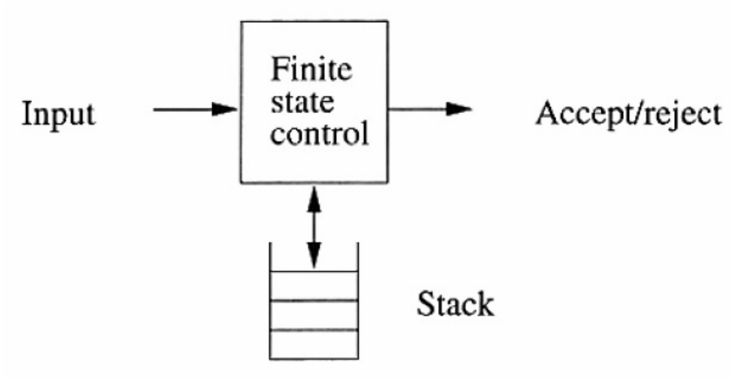
# Contenido

# 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 $\epsilon$-NFA con un *stack*, en donde se puede almacenar una cadena y por lo tanto se puede recordar información.
- Los *pushdown automata*s reconocen todas y solamente las gramáticas libres de contexto.
- Como sólo pueden acceder a la información almacenada en forma LIFO, existen lenguajes reconocidos por una computadora pero no por un PDA, por ejemplo: $\{0^n1^n2^n | n \geq 1\}$.

# Pushdown Automata



# Pushdown Automata

En una transición el PDA:

1. Consume un símbolo de entrada (si se usa $\epsilon$ entonces no se consume ningún símbolo)
2. Se va a un nuevo estado (que puede ser el mismo)
3. Reemplaza el primer elemento del *stack* por alguna cadena
   - Puede ser el mismo símbolo de arriba del *stack* (no hace nada)
   - Hace *pop* (lo que corresponde con $\epsilon$)
   - Cambia el primer elemento por otro (reemplazo sin *push* o *pop* o se puede ver como haciendo ambos)
   - Hace *push* de una cadena al *stack* posiblemente cambiando el primer elemento

# Pushdwon Automata



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

- $Q$: es un conjunto finito de estados
- $\Sigma$: es un alfabeto finito de entradas
- $\Gamma$: 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 $\gamma \in \Gamma$ 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

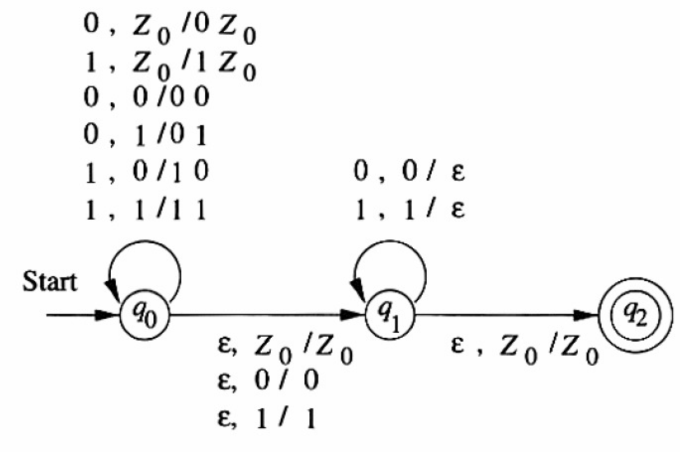
Sea $L_{ww^r} = \{ww^R : w \in \{0,1\}^*\}$ con una gramática:
$P \rightarrow 0P0, P \rightarrow 1P1, P \rightarrow \epsilon$.
Podemos definir un PDA para $L_{ww^r}$ con 3 estados operando de la siguiente manera:

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 $q_2$ y acepta.

# Pushdwon Automata



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



# Pushdwon Automata



- 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 $(p, \alpha)$. Osea nos dice la entrada ($a$) y cómo estaba ($X$) y cómo queda ($\alpha$) la parte superior del *stack*.
- Lo único que no nos dice es cuál es el símbolo inicial del *stack*. Por convención se usa $Z_0$.

# Ejemplo

Así queda el PDA:
$P = (\{q_0, q_1, q_2\}, \{0, 1\}, \{0, 1, Z_0\}, \delta, q_0, Z_0, \{q_2\})$.
Donde $\delta$ está definido por la siguiente tabla:

| | $0, Z_0$ | $1, Z_0$ | $0, 0$ | $0, 1$ | $1, 0$ |
|---|---|---|---|---|---|
| $\rightarrow q_0$ | $q_0, 0Z_0$ | $q_0, 1Z_0$ | $q_0, 00$ | $q_0, 01$ | $q_0, 10$ |
| $q_1$ | | | $q_1, \epsilon$ | | |
| $*q_2$ | | | | | |

cont...

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

El primer elemento se puede escribir como:
$\delta(q_0, 0, Z_0) = \{(q_0, 0Z_0)\}$

# Lo que calcula un PDA



- A diferencia de un FA en donde el estado es todo lo que se necesita saber, en un PDA también importa el contenido del *stack*
- Como el *stack* puede ser arbitrariamente grande, muchas veces es la parte más importante del PDA
- El estado de un PDA se puede representar por una tripleta representando el estado, la cadena y el *stack*

# Descripciones instantáneas o IDs



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

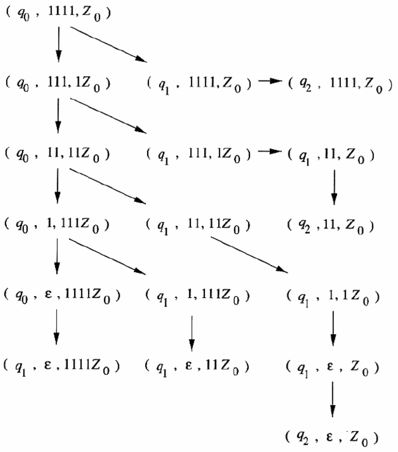
# Cerradura de $\vdash$



- Definimos como la cerradura reflexiva-transitiva de $\vdash$ a $\overset{*}{\vdash}$, para representar cero o más movimientos del PDA.
- Esto es parecido al $\hat{\delta}$ de los FA.
- Por definición $I \overset{*}{\vdash} I$ para toda ID $I$
- Si $I \vdash K$ y si $K \overset{*}{\vdash} J$ entonces $I \overset{*}{\vdash} J$

# Ejemplo

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



# 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.

# Pushdown Automata



**Teorema:** Si $P = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ es un PDA y $(q, x, \alpha) \overset{*}{\vdash} (p, y, \beta)$, entonces $\forall w \in \Sigma^*$ y $\gamma \in \Gamma^*$ también es cierto que

$$(q, xw, \alpha\gamma) \overset{*}{\vdash} (p, yw, \beta\gamma)$$

que cubre la primera propiedad si $\gamma = \epsilon$ y la segunda si $w = \epsilon$ y si tenemos $(q, xw, \alpha) \overset{*}{\vdash} (p, yw, \beta)$ podemos cambiarlo por $(q, x, \alpha) \overset{*}{\vdash} (p, y, \beta)$ que cubre la tercera propiedad

# Ejemplo



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

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

# Ejemplo (cont.)

Autómatas de Pila
Descripciones instantáneas o IDs
El Lenguaje de PDA
Equivalencia entre PDAs y CFGs

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

- 01

$$\begin{aligned}(q, 01, Z_0) \vdash (q, 1, XZ_0) \quad &\vdash (q, \epsilon, XZ_0) \vdash (p, \epsilon, Z_0) \\ &\vdash (p, 1, Z_0) \vdash (p, \epsilon, \epsilon)\end{aligned}$$

- 0011 (T)

Autómatas de Pila
Descripciones instantáneas o IDs
El Lenguaje de PDA
Equivalencia entre PDAs y CFGs

# El Lenguaje de PDA

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

1. Aceptar por "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 "*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



- 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 $P_N (q_0)$.

Autómatas de Pila

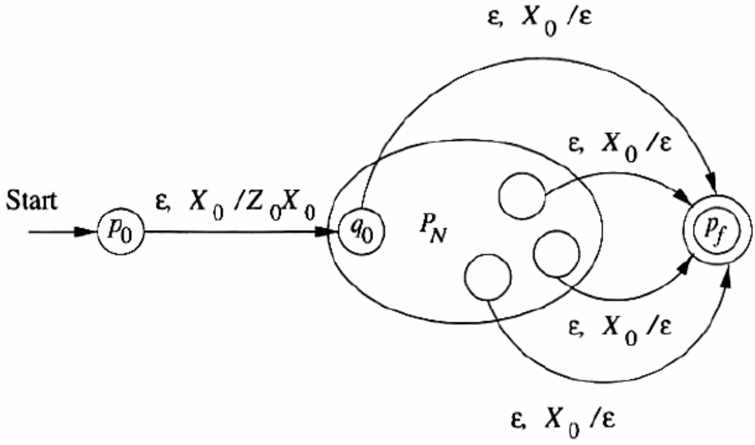
Descripciones instantáneas o IDs

El Lenguaje de PDA

Equivalencia entre PDAs y CFGs

# 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:



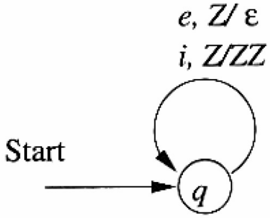
# De *stack* vacío a estado final



Lo anterior lo podemos expresar como sigue:

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

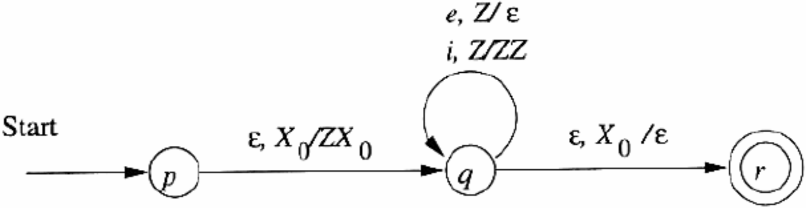
**Ejemplo:** podemos pasar del siguiente PDA que acepta por *stack* vacío a un PDA que acepta por estado final:



# Ejemplo (cont.)



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



Autómatas de Pila

Descripciones instantáneas o IDs

El Lenguaje de PDA

Equivalencia entre PDAs y CFGs

# Ejemplo (cont.)

$P_F = (\{p, q, r\}, \{i, e\}, \{Z, X_0\}, \delta_F, p, X_0, r)$, donde $\delta_F$ 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



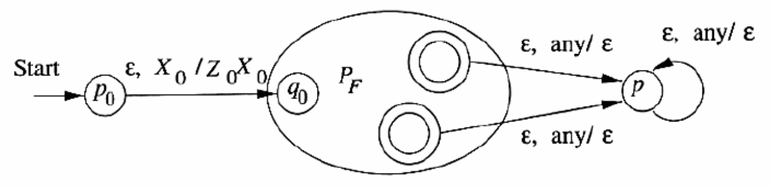
- La idea es la siguiente: Cada vez que se llega a un estado final, se hace una transición vacía a un nuevo estado en donde se vacía el *stack* sin consumir símbolos de entrada.
- Para evitar simular una situación en donde se vacía el *stack* sin aceptar la cadena, de nuevo se introduce al principio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

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

Autómatas de Pila
Descripciones instantáneas o IDs
El Lenguaje de PDA
Equivalencia entre PDAs y CFGs

# 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:



# Ejemplos

Autómatas de Pila
Descripciones instantáneas o IDs
El Lenguaje de PDA
Equivalencia entre PDAs y CFGs

- Diseñe un PDA que acepte el siguiente lenguaje: $\{0^n 1^n | n \geq 1\}$
- Para el siguiente PDA genere las trazas de IDs para la cadena *bab*:

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

- Diseñe un PDA que acepte el siguiente lenguaje: $\{a^i b^j c^k | i = j \text{ o } j = k\}$

Autómatas de Pila

Descripciones instantáneas o IDs

El Lenguaje de PDA

Equivalencia entre PDAs y CFGs

# Soluciones

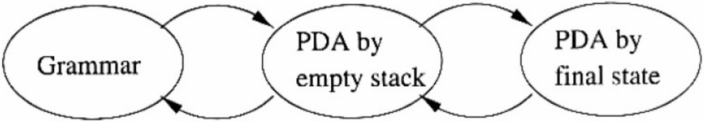
- Aceptando por stack vacío:
  $\delta(q, 0, Z_0) = \{(q, X)\}$ $\quad \delta(q, 0, X) = \{(q, XX)\}$
  $\delta(q, 1, X) = \{(p, \epsilon)\}$ $\quad \delta(p, 1, X) = \{(p, \epsilon)\}$
- $(q_0, bab, Z_0) \vdash (q_2, ab, BZ_0) \vdash (q_3, b, Z_0) \vdash (q_1, b, AZ_0) \vdash (q_1, \epsilon, Z_0) \vdash (q_0, \epsilon, Z_0) \vdash (f, \epsilon, \epsilon)$

# Soluciones

- 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 $\delta(q_2, a, X) = \{(q_2, XX)\}$
  $\delta(q_2, b, X) = \delta(q_4, b, X) = \{(q_4, \epsilon)\}$
  $\delta(q_4, \epsilon, Z_0) = \{(q_1, Z_0)\}$
  $\delta(q_3, a, Z_0) = \{(q_3, Z_0)\}$
  $\delta(q_3, b, Z_0) = \{(q_5, XZ_0)\}$
  $\delta(q_5, b, X) = \{(q_5, XX)\}$
  $\delta(q_5, c, X) = \delta(q_6, c, X) = \{(q_6, \epsilon)\}$
  $\delta(q_6, \epsilon, Z_0) = \{(q_3, \epsilon)\}$

# 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 $\overset{*}{\Rightarrow}_{lm}$
- 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 $\alpha$ 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

Autómatas de Pila
Descripciones instantáneas o IDs
El Lenguaje de PDA
Equivalencia entre PDAs y CFGs

- $A\alpha$ 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 $A \rightarrow \beta$.
- Esto corresponde a un PDA que consume primero a $x$ con $A\alpha$ en el *stack* y luego con $\epsilon$ saca (*pops*) $A$ y mete (*pushes*) $\beta$.
- 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 $\beta$, 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.

# De un CFG a un PDA



Formalmente, sea $G = (V, T, Q, S)$ un CFG. Se define un PDA $P_G$ que acepta $L(G)$ por *stack* vacío como $P_G = (\{q\}, T, V \cup T, \delta, q, S)$ donde:

1. Para cada variable $A \in V$:
   $\delta(q, \epsilon, A) = \{(q, \beta) : A \rightarrow \beta \in Q\}$
2. Para cada símbolo terminal $a \in T$:
   $\delta(q, a, a) = \{(q, \epsilon)\}$

Autómatas de Pila
Descripciones instantáneas o IDs
El Lenguaje de PDA
Equivalencia entre PDAs y CFGs

# Ejemplo

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

Autómatas de Pila
Descripciones instantáneas o IDs
El Lenguaje de PDA
Equivalencia entre PDAs y CFGs

# Ejemplo 2

- Convirtamos la siguiente gramática en un PDA:
  $S \rightarrow 0S1|A$
  $A \rightarrow 1A0|S|\epsilon$
- Los símbolos terminales del PDA son: $\{0, 1\}$
- $P = (\{q\}, \{0, 1\}, \{S, A, 0, 1\}, \delta, q, S)$
- La función de transición del PDA es:
  $\delta(q, \epsilon, S) = \{(q, 0S1), (q, A)\}$
  $\delta(q, \epsilon, A) = \{(q, 1A0), (q, S), (q, \epsilon)\}$
  $\delta(q, 0, 0) = \{(q, \epsilon)\}, \delta(q, 1, 1) = \{(q, \epsilon)\}$

Convertir: $S \rightarrow aAA$, $A \rightarrow aS|bS|a$

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



- Se puede demonstrar que si el PDA $P$ se construye con las dos reglas anteriores, entonces generan el mismo lenguaje
- Sea $w \in L(G)$, entonces, $S = \gamma_1 \Rightarrow_{lm} \gamma_2 \Rightarrow_{lm} \cdots \gamma_n = w$
- Se prueba por inducción en $i$ que: $(q, w, S) \overset{*}{\vdash} (q, y_i, \alpha_i)$ donde $\gamma_i = x_i \alpha_i$ y $w = x_i y_i$
- Por otro lado, también se prueba por inducción que si $A$ es una variable:
  $(q, x, A) \vdash (q, x, Y_1 Y_2 \dots Y_k) \vdash \dots \vdash (q, \epsilon, \epsilon)$
  entonces $A \rightarrow Y_1 Y_2 \dots Y_k$ está en $G$.

# De PDA's a Gramáticas



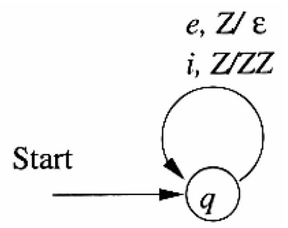
- Se puede también demostrar como pasar de PDA's a CFG's. Osea como consumir una cadena $x = x_1 x_2 \ldots x_n$ y vaciar el *stack*.
- La idea es definir una gramática con variables de la forma $[p_{i-1} Y_i p_i]$ que representan ir de $p_{i-1}$ a $p_i$ haciendo un *pop* de $Y_i$.
- Formalmente, sea $P = (Q, \Sigma, \Gamma, \delta, q_0, Z_0)$ un PDA. Definimos una gramática $G = (V, \Sigma, R, S)$, donde: $V = \{[pXq] : \{p, q\} \subseteq Q, X \in \Gamma\} \cup \{S\}$ $R = \{S \rightarrow [q_0 Z_0 p] : p \in Q \cup \{[qXr_k] \rightarrow a[rY_1 r_1] \ldots [r_{k-1} Y_k r_k] : a \in \Sigma \cup \{\epsilon\}, \{r_1, \ldots, r_k\} \subseteq Q, (r, Y_1 Y_2 \ldots Y_k) \in \delta(q, a, X)\}$

# 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 $\epsilon$), 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 $R = \{S \rightarrow [qZq], [qZq] \rightarrow i[qZq][qZq], [qZq] \rightarrow e\}$.
- Suponemos 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 que tenemos es entonces: $S \rightarrow [qZq]$.
- Debido que tenemos la transición: $\delta_N(q, i, Z) = \{(q, ZZ)\}$ generamos la producción: $[qZq] \rightarrow i[qZq][qZq]$, y con $\delta_N(q, e, Z) = \{(q, \epsilon)\}$, generamos la producción: $[qZq] \rightarrow e$.

# 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 $\delta$ 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:
  $[qZ_0q] \rightarrow 1[qXq], [qZ_0q]$
  $[qZ_0q] \rightarrow 1[qXp], [pZ_0q]$
  $[qZ_0p] \rightarrow 1[qXq], [qZ_0p]$
  $[qZ_0p] \rightarrow 1[qXp], [pZ_0p]$
- De la 2:
  $[qXq] \rightarrow 1[qXq], [qXq]$
  $[qXq] \rightarrow 1[qXp], [pXq]$
  $[qXp] \rightarrow 1[qXq], [qXp]$
  $[qXp] \rightarrow 1[qXp], [pXp]$

Autómatas de Pila
Descripciones instantáneas o IDs
El Lenguaje de PDA
Equivalencia entre PDAs y CFGs

# Ejemplo (cont.)

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

# De PDA's a Gramáticas

Autómatas de Pila
Descripciones instantáneas o IDs
El Lenguaje de PDA
Equivalencia entre PDAs y CFGs

- 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: $[qXp] \overset{*}{\Rightarrow} w$

# Ejemplo



Cambiar el siguiente PDA a una CFG:

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

## 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.

Autómatas de Pila

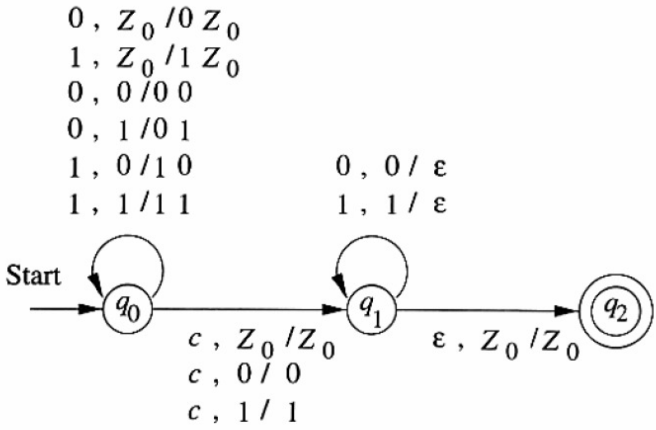
Descripciones instantáneas o IDs

El Lenguaje de PDA

Equivalencia entre PDAs y CFGs

# Ejemplo

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



# 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.

# Ejemplos



- Convertir la gramática:

  $S \rightarrow 0S1 \mid A$
  $A \rightarrow 1A0 \mid S \mid \epsilon$

  A un PDA que acepta el mismo lenguaje por *stack* vacío
- Convertir el PDA $P = (\{p, q\}, \{0, 1\}, \{X, Z_0\}, \delta, q, Z_0)$ a un CFG, donde $\delta$ es:

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

Autómatas de Pila
Descripciones instantáneas o IDs
El Lenguaje de PDA
Equivalencia entre PDAs y CFGs

# Ejemplos

- Convertir el siguiente CFG a un PDA.
  $\{a^n b^m c^{2(n+m)} \mid n \geq 0, m \geq 0\}$
  Una posibilidad es primero construir una gramática y después pasarla a un PDA.
- Diseñar un DPDA que acepte: $\{0^n 1^m | n \leq m\}$