Capítulo 11.

Autómatas a pila

11.1. Concepto de AP

Definición, Representación, Lenguaje reconocido, AP por vaciado de pila, AP por estado final, AP determinista.

11.2. Equivalencia entre AP por vaciado de pila y AP por estado final

11.3. AP y lenguajes independientes de contexto

Obtener un AP para aceptar el lenguaje generado por una GIC. Obtener una GIC para generar el lenguaje aceptado por un AP.

11.1. Concepto de Autómata a pila

Introducción

De igual manera que los lenguajes regulares se pueden representar mediante autómatas finitos deterministas, los lenguajes independientes del contexto tienen su correspondencia en otro tipo de dispositivo: el **Autómata a Pila (AP).**

Un autómata a pila es un dispositivo que tiene acceso a:

- Una secuencia de símbolos de entrada, que en general se representa por una cinta que se desplaza frente a un mecanismo de captación de dichos símbolos.
- El símbolo superior de una memoria en pila (LIFO)

Un autómata a pila se encuentra en cada momento en un estado determinado y el estado siguiente depende de los tres elementos siguientes:

- estado actual
- símbolo de entrada
- símbolo superior de la pila

Generalmente, el autómata a pila es no determinista en el sentido de que se permite que haya varias acciones posibles en cada momento.

Un AP puede realizar dos tipos de operaciones elementales:

1. *Dependientes de la entrada*.

   Se lee la cinta y se avanza la cabeza lectora,

   En función:

   - del estado actual (q_i)
   - del símbolo leído en la cinta (a)
   - del símbolo en la cima de la pila (Z)

   Se pasa a un nuevo estado, se elimina el elemento Z de la cima de la pila y se introduce en su lugar una cadena de símbolos.

2. *Independientes de la entrada.*

   Las mismas operaciones que en el caso anterior, sólo que no se lee la cinta, ni se avanza la cabeza lectora. Se maneja la pila sin la información de entrada.

Definición formal de un AP

Un autómata a pila es una séptupla:

$$AP= (\Sigma, \Gamma, Q, A_0, q_0, f, F)$$

donde :

1. Σ es el alfabeto de entrada
2. Γ es el alfabeto de la pila
3. Q es un conjunto finito de estados
4. $A_0 \in \Gamma$ es el símbolo inicial de la pila
5. $q_0 \in Q$ el estado inicial del autómata
6. $F \subseteq Q$ es el subconjunto de estados finales
7. f es una aplicación denominada *función de transición* de ternas (estado, símbolo de entrada o λ, símbolo de pila) en el conjunto de las partes $Q\times\Gamma^*$

$$f : Q\times\{\Sigma\cup\{\lambda\}\}\times \Gamma \rightarrow 2^{Q\times\Gamma^*} \text{ (subconjunto finito)}$$

Un AP comienza su funcionamiento en la configuración inicial:

- en el estado inicial (q_0)
- con sólo un símbolo en la pila (A_0)
- con la cabeza lectora en el primer símbolo de la entrada

A partir de esta configuración realiza transiciones según la definición de la función *f*.

Interpretación de la función de transición

Representaremos con:

(a, b,...) los elementos de Σ
(A, B, C..) los de Γ
(x, y, z,...) los de Σ^*
(X, Y, Z,...) los de Γ^*

La interpretación de f es:

a) $f(q, a, A) = \{(q_1, Z_1), (q_2, Z_2),... (q_n, Z_n)\}$

cuando el autómata se encuentra en el estado *q*, lee el símbolo de entrada *a* y tiene el símbolo *A* en la cima de la pila; el autómata pasará a algún estado q_i (recordar que es **no** determinista), eliminará el símbolo *A* de la pila e introducirá en ella la palabra Z_i , quedando la cabeza de Z_i en la cima de la pila.

b) $f(q, \lambda, A) = \{(q_1, Z_1), (q_2, Z_2),... (q_n, Z_n)\}$

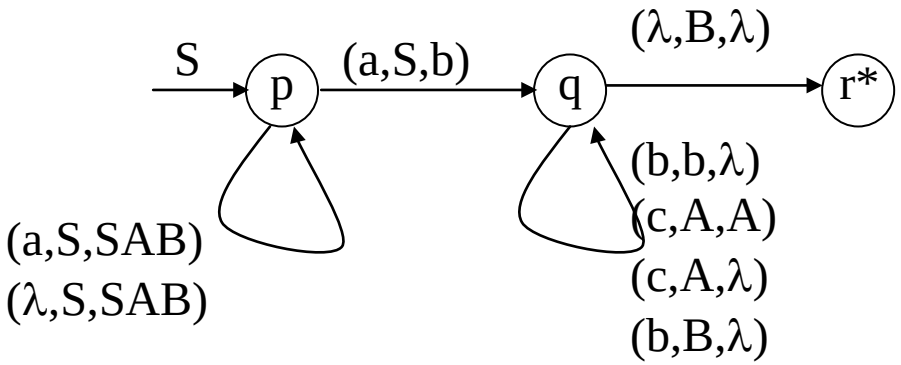
cuando el autómata se encuentra en el estado *q*, y tiene el símbolo *A* en la cima de la pila; el autómata pasará a algún estado q_i (recordar que es **no** determinista), eliminará el símbolo *A* de la pila e introducirá en ella la palabra Z_i , quedando la cabeza de Z_i en la cima de la pila.

Se entiende que el resultado de la función f para las configuraciones (estado, símbolo de entrada y símbolo de pila) no explícitamente especificadas es el conjunto vacío. Estas representan configuraciones “muertas” del autómata.

Representación gráfica

AP=({a,b,c},{S,A,B,b},{p,q,r},S,p,f,{r})

f(p,a,S)={(p,SAB), (q,b)}
f(p,λ,S)={(p,SAB)}
f(q,b,b)={(q,λ)}
f(q,b,B)={(q,λ)}
f(q,c,A)={(q,A),(q,λ)}
f(q,λ,B)={(r,λ)}



- Cada estado corresponde a un nodo en el grafo y está etiquetado con el nombre del estado (si es un estado final se marca además con *)
- Cada transición (q, Z)∈f(p, a, A) corresponde a un arco del nodo *p* al nodo *q* y tiene la etiqueta (a,A,Z).
- Las etiquetas de los arcos pueden tener la forma: (a,A,λ), (a,A,B), (a,A,BC...), (λ,A,λ), (λ,A,B), (λ,A,BC...) siendo (a∈Σ, A,B,C∈Γ).
- No hay transiciones de forma:
  (..., λ, ...), (ab..., ..., ...) o (..., AB..., ...)

Descripción instantánea

Una descripción instantánea o configuración es una terna (q, x, Z) donde $q \in Q$, $x \in \Sigma^*$, $Z \in \Gamma^*$ que define: el estado del autómata, la entrada que resta por leer y el contenido de la pila en un momento dado.

Movimientos

Un movimiento es el paso de una descripción instantánea a otra y se produce como resultado de la aplicación de la función *f*.
Se representa por el símbolo $\vdash$, que se lee como *"precede a"*.

Hay dos tipos de movimientos:

$$(q, az, AZ) \vdash (p, z, YZ) \quad \text{si } (p, Y) \in f(q, a, A)$$

$$(q, z, AZ) \vdash (p, z, YZ) \quad \text{si } (p, Y) \in f(q, \lambda, A)$$

$(q,p \in Q,\ a \in \Sigma,\ A \in \Gamma,\ z \in \Sigma^*,\ Z,Y \in \Gamma^*)$

Utilizaremos el símbolo $\vdash^*$ para expresar el cierre transitivo y reflexivo de $\vdash$. Es decir se entiende por $I_i \vdash^* I_j$ el resultado de una serie movimientos tales que

$$I_i \vdash^* I_j \Rightarrow I_i \vdash I_k \ldots\ldots \vdash I_j$$

Lenguaje aceptado por un autómata a pila

Se describe el proceso de aceptación o rechazo de una palabra de Σ^* mediante una sucesión de movimientos.

Un AP= (Σ, Γ, Q, A_0, q_0, f, F) puede reconocer palabras del alfabeto de entrada de dos formas distintas:

- por estado final:

$L_F(AP) = \{x \mid (q_0, x, A_0) \vdash^* (p, \lambda, X), \text{ con } p \in F, X \in \Gamma^*\}$

- por vaciado de pila :

$L_V(AP) = \{ x \mid (q_0, x, A_0) \vdash^* (p, \lambda, \lambda) \text{ con } p \in Q\}$

$L_F(AP)$ y $L_V(AP)$ representan a los lenguajes reconocidos por el autómata AP por estado final y por vaciado de pila respectivamente.

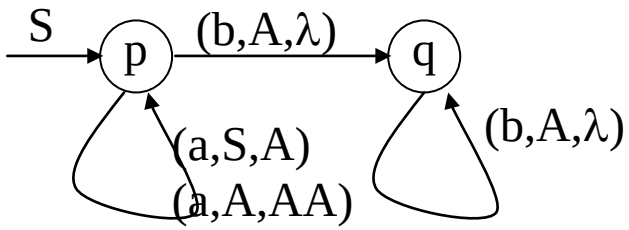
Cuando la aceptación se realiza por vaciado de pila, el conjunto de estados finales F es irrelevante.

Equivalencia de AP

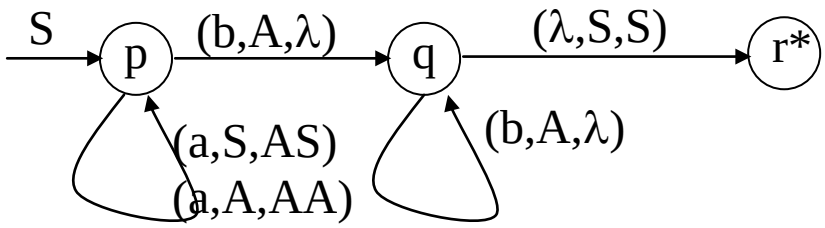
Dos autómata a pila (por vaciado de pila o por estado final), AP_1 y AP_2, son equivalentes, si aceptan el mismo lenguaje, es decir, si $L(AP_1)=L(AP_2)$.

Ejemplo:

Autómata a pila para el lenguaje $L=\{a^n b^n \mid n > 0\}$
APV=({a,b},{S,A},{p,q},S,p,f,∅)
f(p,a,S)={(p,A)}
f(p,a,A)={(p,AA)}
f(p,b,A)={(q, λ)}
f(q,b,A)={(q, λ)}



APF=({a,b},{S,A},{p,q,r},S,p,f,{r})
f(p,a,S)={(p,AS)}
f(p,a,A)={(p,AA)}
f(p,b,A)={(q,λ)}
f(q,b,A)={(q,λ)}
f(q,λ,S)={(r,S)}



Movimientos para (p,aabb,S) (p,abb,S) (p,aab,S)

Otro ejemplo:
Autómata a pila para el lenguaje
$L=\{ww^R \mid w \in \{a,b\}^*, w^R$ – imagen inversa de w}.

Autómatas a pila deterministas

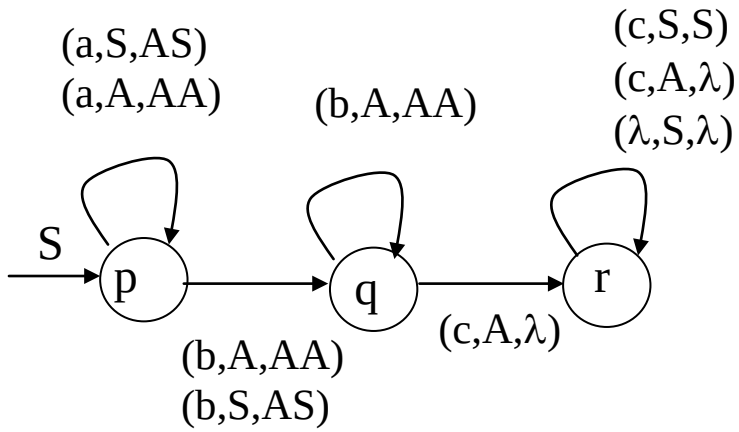
Decimos que un autómata a pila es determinista si se verifica lo siguiente:

1. $\forall\ q \in Q,\ \forall\ A \in \Gamma$:
   $f(q, \lambda, A) \neq \varnothing \Rightarrow \forall\ a \in \Sigma: f(q, a, A) = \varnothing$
2. $\forall q \in Q,\ \forall A \in \Gamma, \forall a \in \Sigma \cup \{\lambda\}$:
   $f(q, a, A)$ contiene como máximo un elemento.

A lo sumo, desde una configuración cualquiera existe como mucho un posible movimiento.

A diferencia de los autómatas finitos, se entiende que un AP es **no** determinista, a menos que se diga lo contrario.

Ejemplo:
AP (por vaciado de pila) para $L=\{a^n b^m c^p \mid n \geq 0, m \geq 1, p \geq n+m\}$



¿Determinista o no?

Construir un APP determinista para $L=\{a^n b^m c^p \mid n \geq 0, m \geq 1, p = n+m\}$.

11.2. Equivalencia de APF y APV

Teorema:

El conjunto de lenguajes aceptados por estado final por los autómatas a pila L_{APF} es igual que el conjunto de lenguajes aceptados por vaciado por pila de los autómatas a pila L_{APV}.

Método de demostración:

1. $L_{APF} \subseteq L_{APV}$

Sea $AP= (\Sigma, \Gamma, Q, A_0, q_0, f, F)$ un autómata a pila y $L_F(AP)$ el lenguaje aceptado (por estado final) de este autómata. Construimos $AP'= (\Sigma, \Gamma \cup \{B\}, Q \cup \{s,r\}, B, s, f', \varnothing)$, con $B \notin \Gamma$ y $s,r \notin Q$, donde f' esta definido por:

$f'(s,\lambda,B)=\{(q_0,A_0B)\}$
$f'(q,a,A)=f(q,a,A)$ para todo $q \in Q$, $q \notin F$, $a \in \Sigma \cup \{\lambda\}$ y $A \in \Gamma$
$f'(q,a,A)=f(q,a,A)$ para todo $q \in F$, $a \in \Sigma$ y $A \in \Gamma$
$f'(q,\lambda,A)=f(q,\lambda,A) \cup \{(r, \lambda)\}$ para todo $q \in F$ y $A \in \Gamma$
$f'(q,\lambda,B)= \{(r, \lambda)\}$ para todo $q \in F$
$f'(r,\lambda,A)= \{(r, \lambda)\}$ para todo $A \in \Gamma \cup \{B\}$

Se puede mostrar que $L_F(AP)=L_V(AP')$. Por tanto se verifica que $L_{APF} \subseteq L_{APV}$.

2. $L_{APV} \subseteq L_{APF}$

Sea AP= (Σ, Γ, Q, A_0, q_0, f, F) un autómata a pila y L_V(AP) el lenguaje aceptado (por vaciado de pila) de este autómata. Construimos AP'= (Σ, $\Gamma \cup \{B\}$, $Q \cup \{s,r\}$, B, s, f', {r}), con $B \notin \Gamma$ y $s,r \notin Q$, donde f' esta definido por:

f'(s,λ,B)={(q_0,A_0B)}
f'(q,a,A)=f(q,a,A) para todo $q \in Q$, $a \in \Sigma \cup \{\lambda\}$ y $A \in \Gamma$
f'(q,λ,B)= {(r, λ)} para todo $q \in Q$

Se puede mostrar que L_V(AP)=L_F(AP'). Por tanto se verifica que $L_{APV} \subseteq L_{APF}$.

De $L_{APF} \subseteq L_{APV}$ y $L_{APV} \subseteq L_{APF}$ se sigue que $L_{APV} = L_{APF}$, lo que demuestra el teorema.

11.3. AP y lenguajes independientes del contexto

Construcción de un AP a partir de una GIC

Teorema:
Sea AP_Σ el conjunto de todos los autómatas a pila sobre un alfabeto Σ y $L_{AP\_\Sigma}=\{L \mid L=L(AP) \text{ y } AP \in AP_\Sigma\}$ la familia de lenguajes sobre Σ aceptados por los autómatas a pila. (Da igual si se acepta el lenguaje por estado final o por vaciado de pila.)
Sea $L_{G\_2\_\Sigma}$ la familia de todos los lenguajes sobre Σ generados por gramáticas del tipo 2 (lenguajes independientes del contexto).

$$\Rightarrow L_{G\_2\_\Sigma} = L_{AP\_\Sigma} .$$

Método de demostración:

1. $L_{G\_2\_\Sigma} \subseteq L_{APV\_\Sigma}$
   Para cada GIC, G, existe un autómata a pila por vaciado de pila, AP, tal que $L(G)=L_V(AP)$.
   $\Rightarrow$ Encontrar un algoritmo para construir AP a partir de una gramática genérica G.
2. $L_{G\_2\_\Sigma} \supseteq L_{APV\_\Sigma}$
   Para cada autómata a pila, AP, por vaciado a de pila existe una GIC, G, tal que $L(G)=L_V(AP)$.
   $\Rightarrow$ Encontrar un algoritmo para construir G a partir de un AP genérico.

$L_{G\_2\_\Sigma} \subseteq L_{APV\_\Sigma}$ y $L_{G\_2\_\Sigma} \supseteq L_{APV\_\Sigma} \Rightarrow L_{G\_2\_\Sigma}=L_{APV\_\Sigma}$

Se verifica: $L_{ic\_\Sigma}= L_{G\_2\_\Sigma}=L_{APV\_\Sigma}=L_{APF\_\Sigma}=L_{AP\_\Sigma}$

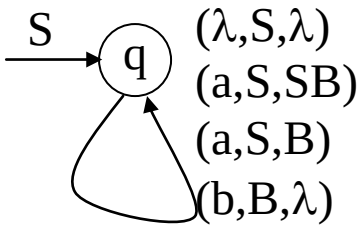
GIC $\Rightarrow$ AP reconocedor por vaciado de pila

Ejemplo:

Lenguaje L={ $a^n b^n$ | n≥0}:
S::=aSb|λ $\Rightarrow$(bien formada) S::= aSb|ab|λ

1. Transformar gramática en FNG:
G=({a,b},{S,B},S,P)
P={S::= aSB|aB|λ, B::= b}

2. Construir autómata:



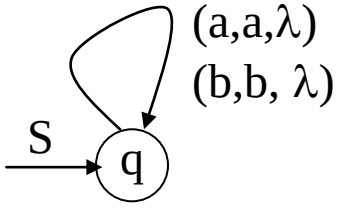
AP= ({a,b}, {S,B}, {q}, S, q, f, ∅), donde f se define por:
f(q,λ,S)={(q, λ)} f(q,b,B)={(q,λ)}
f(q,a,S)={(q,SB),(q,B)}

(q,aabb,S) ├ (q,abb,SB)├ (q,bb,SBB)├ (q,bb,BB)├ (q,b,B)├ (q,λ,λ)
otra posibilidad: (q,aabb,S) ├ (q,abb,B) no

(q,abb,S) ├ (q,bb,SB)├ (q,bb,B)├ (q,b,λ) no
otra posibilidad:
(q,abb,S) ├ (q,abb,λ) no
(q,abb,S) ├ (q,bb,B)├ (q,b,λ) no

Otra posibilidad:

Gramática en FNG
G=({a,b},{S,B},S,P)
P={S::= aSB|aB|λ, B::= b}



AP= ({a,b}, {S,B,a,b}, {q}, S, q, f, ∅), donde f se define:
f(q,λ,S)={(q,aSB),(q,aB),(q,λ)}
f(q,λ,B)={(q,b)}
f(q,a,a)={(q,λ)}
f(q,b,b)={(q,λ)}

(q,aabb,S)├ (q,aabb,aSB)├ (q,abb,SB)├ (q,abb,aBB)├ (q,bb,BB)├ (q,bb,bB)├ (q,b,B)├ (q,b,b)├ (q,λ, λ)

(q,abb,S)├ (q,abb,aSB)├ (q,bb,SB)├ (q,bb,aBB) no
otra posibilidad:
(q,abb,S)├ (q,abb,λ) no
(q,abb,S)├ (q,abb,aB)├ (q,bb,B)├ (q,bb,b)├ (q,b,λ) no

Algoritmo:

Sea la gramática $G = (\Sigma_T, \Sigma_N, S, P)$, que suponemos en **forma normal de Greibach.** A partir de ella construimos un autómata a pila $AP= (\Sigma, \Gamma, Q, A_0, q_0, f, F)$ como sigue:

Algoritmo 1:

$\Sigma = \Sigma_T$, $\Gamma= \Sigma_N$, $Q=\{q\}$, $A_0 = S$, $q_0=q$, $F= \varnothing$

y f se define por:

1. $(q, \lambda) \in f(q, \lambda, S)$ si $(S::= \lambda) \in P$
2. $(q, Z) \in f(q, a, A)$ si $(A::= aZ) \in P$

Algoritmo 2:

$\Sigma = \Sigma_T$, $\Gamma= \Sigma_N \cup \Sigma_T$, $Q=\{q\}$, $A_0 = S$, $q_0=q$, $F= \varnothing$

y f se define por:

1. $(q, \lambda) \in f(q, \lambda, S)$ si $(S::= \lambda) \in P$
2. $(q, aZ) \in f(q, \lambda, A)$ si $(A::= aZ) \in P$
3. $f(q, a, a)=\{(q, \lambda)\}$ para todo $a \in \Sigma_T$

Ejemplo:
Lenguaje de las paréntesis (bien formada):
S::=SS|aSb|λ $\Rightarrow$ S::= SS|aSb|ab|λ

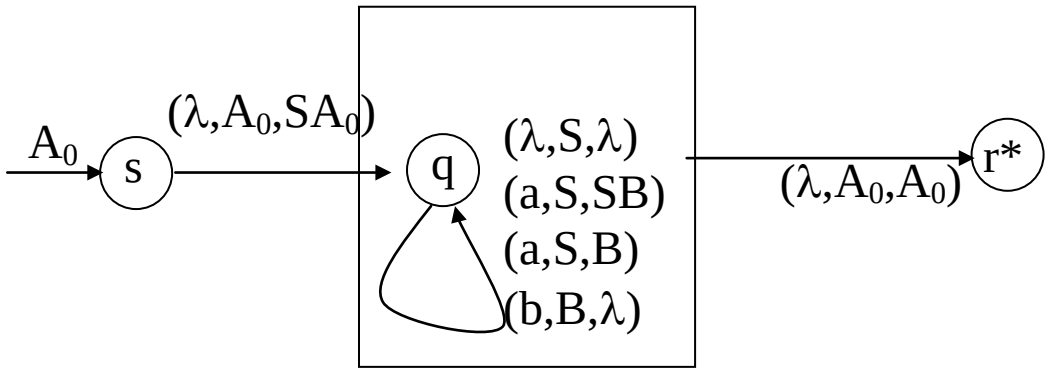
1. Transformar gramática en FNG (proceso sin S::= λ):
G=({a,b},{S,B,C},S,P)
P= {S::= aSBC|aBC|aSB|aB|λ
C::= aSBCC|aBCC|aSBC|aBC|aSB|aB, B::= b}

2. Construir autómata:
AP= ({a,b}, {S,B,C}, {q}, S, q, f, ∅), donde f se define:
f(q,λ,S)={(q, λ)}
f(q,a,S)={(q,SBC),(q,BC),(q,SB),(q,B)}
f(q,a,C)={(q,SBCC),(q,BCC),(q,SBC),(q,BC),(q,SB),(q,B)}
f(q,b,B)={(q,λ)}

(q,abaababb,S)├(q,baababb,BC)├(q,aababb,C)├(q,ababb,SB)├
(q,babb,BCB)├(q,abb,CB)├(q,bb,BB)├(q,b,B)├(q,λ,λ)

GIC ⇒ AP reconocedor por estado final

Idea:
Construir el AP por vaciado de pila (de un estado) y añadir dos estados nuevos:



Algoritmo:
Sea la gramática $G = (\Sigma_T, \Sigma_N, S, P)$, que suponemos en **forma normal de Greibach.** A partir de ella construimos un autómata a pila AP como sigue:

1. Construir el autómata $AP' = (\Sigma', \Gamma', \{q\}, S, q, f', F')$ que acepta por vaciado de pila con el algoritmo dado anteriormente. Sea q el único estado de este autómata.

2. $AP=(\Sigma', \Gamma' \cup A_0, \{q, r, s\}, A_0, s, f, \{r\})$ con $A_0 \notin \Gamma'$ y f definido por:

$f(s, \lambda, A_0)=\{(q, SA_0)\}$

$f(q, a, A)=f'(q,a,A)$ para todo $a \in \Sigma' \cup \{\lambda\}$ y $A \in \Gamma'$

$f(q, \lambda, A_0)=\{(r, A_0)\}$

Construcción de una GIC a partir de un AP genérico

Sea el autómata a pila AP=(Σ, Γ, Q, A_0, q_0, f, $\varnothing$) (que acepta por vaciado de pila). Para todo q_i,$q_j \in Q$, $A \in \Gamma$:

El símbolo [$q_i A q_j$] *representa el conjunto de cadenas* x *que permiten al autómata eliminar el símbolo* A *de la cima de la pila, partiendo del estado* q_i *y terminando en el estado* q_j, *consumiendo todos los símbolos de* x**.** *Formalmente*:

$$\{ x \in \Sigma^* / (q_i,x,A) \vdash^* (q_j,\lambda,\lambda) \}$$

A partir de este autómata, y de la definición anterior, construimos la siguiente GIC: G=(Σ_T, Σ_N, S, P), donde:

$$\Sigma_T=\Sigma,\ \Sigma_N=\{S\} \cup \{[q_i A q_j] \mid q_i, q_j \in Q,\ A \in \Gamma\}$$

y el conjunto de producciones P:

1. para todo $q_i \in Q$:
   $S ::= [q_0 A_0 q_i] \in P$
2. por cada $(q_j, Y_1 Y_2 \ldots Y_k) \in f(q_i, a, A)$, donde $k > 0$, y $a \in \Sigma \cup \{\lambda\}$:
   $\{[q_i A r_k] ::= a[q_j Y_1 r_1][r_1 Y_2\ r_2] \ldots [r_{k-1} Y_k\ r_k]$,
   para todas las listas de estados $(r_1,r_2,\ldots,r_k) \in Q^k \} \subset P$
3. por cada $(q_j, \lambda) \in f(q_i, a, A)$, donde $a \in \Sigma \cup \{\lambda\}$:
   $\{[q_i A q_j] ::= a$, para todo $q_j \in Q \} \subset P$

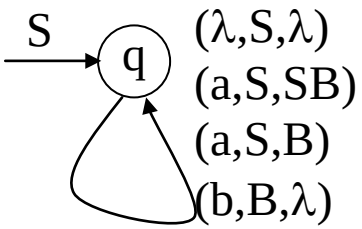
Se puede demostrar que $L(G)=L_V(AP)$, es decir:

$S \rightarrow^* x$, si y sólo si, $(q_0,x,A_0) \vdash^* (q_j,\lambda,\lambda)$, para algún q_j

Eso completa la demostración para $L_{G\_2\_\Sigma} \supseteq L_{APV\_\Sigma}$.
Dado que $L_{G\_2\_\Sigma} \subseteq L_{APV\_\Sigma}$ (construcción de AP a partir de G) se verifica $L_{G\_2\_\Sigma} = L_{APV\_\Sigma}$.

Ejemplo:

Autómata (por vaciado de pila):



AP= ({a,b}, {S,B}, {q}, S, q, f, ∅), donde f se define por:

f(q,λ,S)={(q, λ)} f(q,b,B)={(q,λ)}

f(q,a,S)={(q,SB),(q,B)}

Gonstruir G a partir de AP:

G=({a,b}, Σ_N, A, P)

Σ_N={A, [qSq], [qBq]}

P={ A::=[qSq] , [qSq]::= λ, [qBq]::= b

[qSx]::=a[qBx] con x∈Q ⇒ [qSq]::=a[qBq]

[qSx]::=a[qSy][yBx] con x,y ∈Q ⇒ [qSq]::=a[qSq][qBq] }

Usamos otros símbolos y simplificamos:

P={ A::=S, S::=λ|aB|aSB, B::=b} ⇒ {A::=λ|aB|aAB, B::=b}

Autómata:

(q,aaabbb,S) ├ (q,aabbb,SB)├ (q,abbb,SBB) ├ (q,bbb,BBB)├
(q,bb,BB)├ (q,b,B)├ (q,λ,λ)

Gramática:

A→aAB→aaABB→aaaBBB→aaabBB→ aaabbB→ aaabbb

Otro ejemplo: