INSTITUTO POLITÉCNICO NACIONAL
ESCUELA SUPERIOR DE CÓMPUTO

MATERIA: Teoría Computacional.

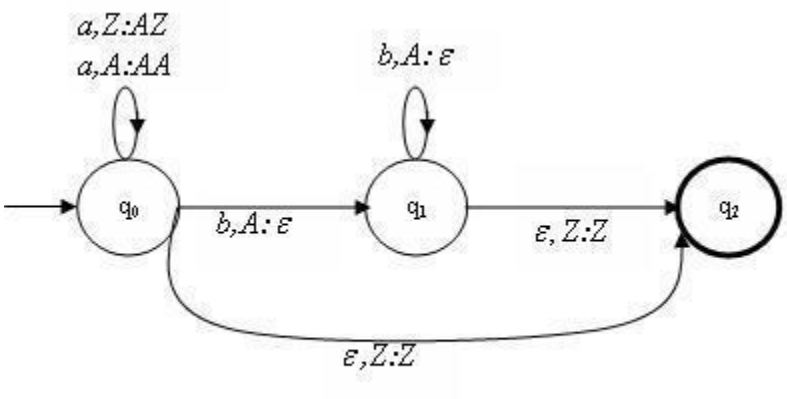
PRÁCTICA: Práctica 6. ADPND.

ALUMNO: Meza Vargas Brandon David.

PROFESOR: Jorge Luis Rosas Trigueros.

FECHA PRÁCTICA: 04-dic-2020

FECHA DE ENTREGA: 11-dic-2020



MARCO TEÓRICO

Autómata De Pila No Determinista

Un autómata de pila no determinista (ADPND) es una 7-tupla
M=(Q,Σ,Γ,Δ,s,F,z)

- Q es un conjunto finito de estados
- Σ es un alfabeto de entrada
- **Γ es un alfabeto llamado alfabeto de la pila**
- Δ es una regla de transición Δ ⊆ Qx(ΣU{ε}) xΓxQxΓ*
- sϵQ es el estado inicial
- **zϵΓ es el símbolo inicial de la pila**
- F⊆Q es el conjunto de estados finales o de aceptación

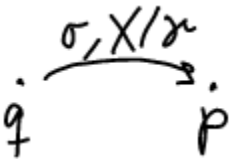
La pila funciona de manera que el ultimo carácter que se almacena en ella es el primero en salir (“LIFO” por las siglas en ingles), como si empiláramos platos uno encima de otro, y naturalmente el primero que quitaremos es el último que hemos colocado.

Al igual que los AF, los AP tienen estados finales, que permiten distinguir cuando una palabra de entrada es aceptada. De hecho, para que una palabra de entrada sea aceptada en un AP se deben cumplir todas las condiciones siguientes:

- La palabra de entrada se debe haber agotado (consumido totalmente).
- El AP se debe encontrar en un estado final.
- La pila debe estar vacía.

En la representación gráfica de un ADPND, los elementos de Q, s, F se representan de la misma manera que en los AFD. En las transiciones, ilustraremos con una flecha los estados inicial y final, y la etiqueta tendrá la forma σ,X/γ, donde σϵ(ΣU{ε}) es el símbolo consumido de la cadena de entrada, XϵΓ es el símbolo que se encuentra visible en la pila (este símbolo es retirado de la pila) y γϵΓ* es lo que se introduce a la pila.

Δ(q,σ,X)= (p,γ)



La terna (q,w,u) donde q es el estado actual, w es la cadena de entrada restante y u el contenido de la pila (con el símbolo de la cima en el extremo de la izquierda) se llama descripción instantánea del autómata.

(p, aw,bx) ⊢ (q, w, yx)

Representa Δ(p,a,b)=(q,y)

DESARROLLO

La práctica consistió en la programación de dos autómatas de pila no determinista, el primer ejercicio constó del ADPND

L1= {a^nb^n}

Primeramente, tenemos los elementos del autómata, los vemos en la figura 1.

```
Q=['q0','ql','q2','q3']
sigma=['a','b','e']
Ap=['A','B','']
s='q0'
z='A'
F=['q3']
pila=[]
```

Figura 1. Elementos del autómata.

Después hice una función llamada insertar, la cual se encarga de insertar la cadena en la pila, la vemos en la figura 2.

```
def insertar(gamma,pila):
    for s in gamma[::-1]:
        pila.insert(0,s)
    return pila;
```

Figura 2. Función insertar.

Posteriormente, tenemos dos ciclos for, el primer ciclo se encarga de recorrer las cadenas dentro de una lista, las cuales serán evaluadas y determinar si van o no en nuestro ADPND.

En el primero inicializamos la pila, así como el estado actual y definimos una posición para determinar la cantidad de a's y de b's. Ver figura 3

```
for y in Ejemplos_L:
    pila=[z]
    estado=s
    pos=y.find("b")
    bs=y[pos: ]
    ca=y[:pos]

    p=y+'e'
```

Figura 3. Inicio del primer for.

En el segundo if, encontramos ifs que determinan que carácter de la cadena se va a evaluar, de esta forma determinar que se introducirá a la pila y que se sacará, checando en qué estado se encuentra nuestro autómata y a qué estado se dirigirá.

Esto lo podemos ver en la figura 4.

```
if estado==s:
    if x==sigma[0]:
        pila.pop(0)
        insertar('BA',pila)
        estado=Q[1]
    if x==sigma[2] and pila[0]==z:
        pila.pop(0)
        estado=Q[3]
    if x==sigma[1]:
        break;

elif estado==Q[1]:
    if x==sigma[0] and pila[0]==Ap[1]:
        pila.pop(0)
        insertar('BB',pila)

    if x==sigma[1] and pila[0]==Ap[1]:
        pila.pop(0)
        estado=Q[2]

elif estado==Q[2]:
    if x==sigma[0]:
        break;
    if x==sigma[1] and pila[0]==Ap[1]:
        pila.pop(0)

    if  x==sigma[2] and pila[0]==Ap[0]:
        pila.pop(0)
        pila.insert(0,'A')
        estado=Q[3]
```

Figura 4. Segundo ciclo for.

Al final del primer if verificamos que la cadena este en un estado de aceptación para saber si es o no aceptada la cadena, ver figura 5.

```
if estado in F:
    if len(bs)!=len(ca):
        print(y, "No aceptada")
    else:
        print(y, " es aceptada")
        print(pila)
else:
    print(y,"No aceptada")
```

Figura 5. Final del primer for.

La salida nos muestra algunas cadenas que son aceptadas y otras que no lo son, además del estado final de la pila. Ver figura 6.

```
  es aceptada
[]
ab  es aceptada
['A']
aabb  es aceptada
['A']
aaabbb  es aceptada
['A']
aaaabbbb  es aceptada
['A']
a No aceptada
aba No aceptada
baba No aceptada
aabbbbb No aceptada
```

Figura 6. Salido del código.

L2= {cadenas que tengan el mismo número de a's que de b's }

Primeramente, colocamos los elementos del autómata. (ver figura 2)

```
Q=['q0','ql']
sigma=['a','b','e']
Ap=['A','B','Z']
s='q0'
z='Z'
F=['ql']
pila=[]
```

Figura 7. Elementos del ADPND

Después hice una función llamada insertar, la cual se encarga de insertar la cadena en la pila, a vemos en la figura 8.

```
def insertar(gamma,pila):
    for s in gamma[::-1]:
        pila.insert(0,s)
    return pila;
```

Figura 8. Función insertar.

Después hacemos un ciclo for el cuál recorre todas las palabras dentro de una lista para verificar si las acepta o no el autómata de pila, dentro de este encontramos otro ciclo que va recorriendo cada carácter de las palabras, ver figura 9.

```
Ejemplos_L=['','ab','ba','aabb','abb','a','b','abab','baba','ababa','abba']
estado=s
c=0

for y in Ejemplos_L:
    pila=[z]
    estado=s
    p=y+'e'
    for x in p:
```

Figura 9. Ciclos for.

Dentro de este segundo for encontramos varios ifs para determinar lo que se ingresará a la pila y lo que se quitará, hacemos uso de la función de la figura 2, así como el pop para sacar elementos de la pila.

El primer if sirve para verificar en que estado estamos, los ifs dentro de este se encargan de checar que carácter de la cadena va y que símbolos hay en la pila para determinar que se saca y que se mete a ella. Esto lo vemos en la figura 10.

```
if estado==s:
    if x==sigma[0] and pila[0]==z and c==0:
        pila.pop(0)
        insertar('AZ',pila)
        c=1
    if x==sigma[0] and pila[0]==Ap[0] and c==0:
        pila.pop(0)
        insertar('AA',pila)
        c=1
    if x==sigma[0] and pila[0]==Ap[1] and c==0:
        pila.pop(0)
        c=1
    if x==sigma[1] and pila[0]==z and c==0:
        pila.pop(0)
        insertar('BZ',pila)
        c=1
    if x==sigma[1] and pila[0]==Ap[1] and c==0:
        pila.pop(0)
        insertar('BB',pila)
        c=1
    if x==sigma[1] and pila[0]==Ap[0]and c==0:
        pila.pop(0)
        c=1
    if x==sigma[2] and pila[0]==z and c==0:
        pila.pop(0)
        pila.insert(0,z)
        estado=Q[1]
        c=1
```

Figura 10. Ifs que conforman el segundo ciclo for.

Finalmente, dentro del primer if verificamos si el estado al que se llego esta en el de aceptación y si es así, la cadena es aceptada. Ver figura 11.

```
if estado in F:
        print(y, " es aceptada")
        print(pila)
else:
    print(y,"No aceptada")
```

Figura 11. Verificamos si la cadena es aceptada o no.

La salida nos muestra algunas cadenas que son aceptadas y otras que no lo son, además del estado final de la pila. Ver figura 12.

```
  es aceptada
['Z']
ab  es aceptada
['Z']
ba  es aceptada
['Z']
aabb  es aceptada
['Z']
abb No aceptada
a No aceptada
b No aceptada
abab  es aceptada
['Z']
baba  es aceptada
['Z']
ababa No aceptada
abba  es aceptada
['Z']
```

Figura 12. Salida de nuestro código.

CONCLUSIONES Y RECOMENDACIONES

A partir de los ejercicios realizados en la práctica, pude comprender de una mejor manera lo revisado en la clase de autómata de pila no determinista, ya que al momento de estar programándolo repasé como es que funcionan estos autómatas y eso me permitió entenderlo mejor.

Tengo que decir que me costo un poco de trabajo establecer las condiciones y lo necesario para ir formando el autómata de pila no determinista, pero al final lo logre de una buena forma, pero siendo sincero, creo que existe una forma más óptima de lograr esto.

La práctica fue realizada de manera adecuada, pues el profesor resolvió dudas presentadas y dio el tiempo necesario para su realización.

BIBLIOGRAFÍA

- Apuntes de la clase de Teoría Computacional por el profesor Jorge Luis Rosas Trigueros.
- Román, P. (2016). “Autómata de Pila”. Obtenido de: https://es.slideshare.net/PedroRoman10/automata-de-pila-y-maquina-de-turing-no-deterministas