Autómata de Pila

Sebastián Acevedo Juárez
Teoría de la computación

June 2022

1 Introducción

En este reporte se muestra la resolución del problema del autómata de pila que menciona lo siguiente: Programar un autómata de pila que sirva para reconocer el lenguaje libre de contexto $0^n1^n|n >= 1$. Se pueden encontrar tanto las capturas de la ejecución, la implementación en código y una breve explicación del tema a tratar.

2 Marco Teórico

Los autómatas de pila, en forma similar a como se usan los autómatas finitos, también se pueden utilizar para aceptar cadenas de un lenguaje definido sobre un alfabeto A. Los autómatas de pila pueden aceptar lenguajes que no pueden aceptar los autómatas finitos. Un autómata de pila cuenta con una cinta de entrada y un mecanismo de control que puede encontrarse en uno de entre un número finito de estados. Uno de estos estados se designa como estado inicial, y además algunos estados se llaman de aceptación o finales. A diferencia de los autómatas finitos, los autómatas de pila cuentan con una memoria auxiliar llamada pila. Los símbolos (llamados símbolos de pila) pueden ser insertados o extraídos de la pila, de acuerdo con el manejo last-in-first-out (LIFO).
Las transiciones entre los estados que ejecutan los autómatas de pila dependen de los símbolos de entrada y de los símbolos de la pila. El autómata acepta una cadena x si la secuencia de transiciones,comenzando en estado inicial y con pila vacía, conduce a un estado final, después de leer toda la cadena x. Un Autómata Finito Determinista consta de:

1. Q: Conjunto finito de estados
2. Σ: Alfabeto o conjunto finito de símbolos de la cinta de entrada
3. L: Alfabeto o conjunto finito de símbolos de la Pila.(lenguaje)
4. δ: función de transición de estados
5. $q0$: Estado inicial $q0$

6. $Z0$: Símbolo distinguido(En el programa usamos el símolo \$)

7. F: Conjunto de estados finales o estados de aceptación.

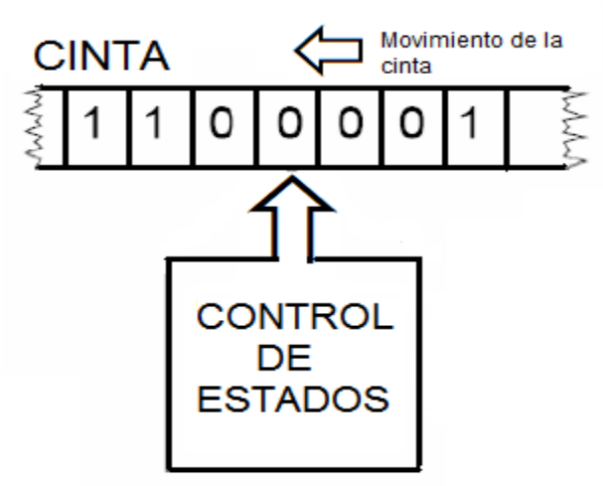


Figure 1: Representación gráfica de un PDA

Estas gramáticas, conocidas también como gramáticas de tipo 2 o gramáticas independientes del contexto, son las que generan los lenguajes libres o independientes del contexto. Los lenguajes libres del contexto son aquellos que pueden ser reconocidos por un autómata de pila determinístico o no determinístico. Como toda gramática se definen mediante una cuadrupla G = (N, T, P, S), siendo

1. N es un conjunto finito de símbolos no terminales

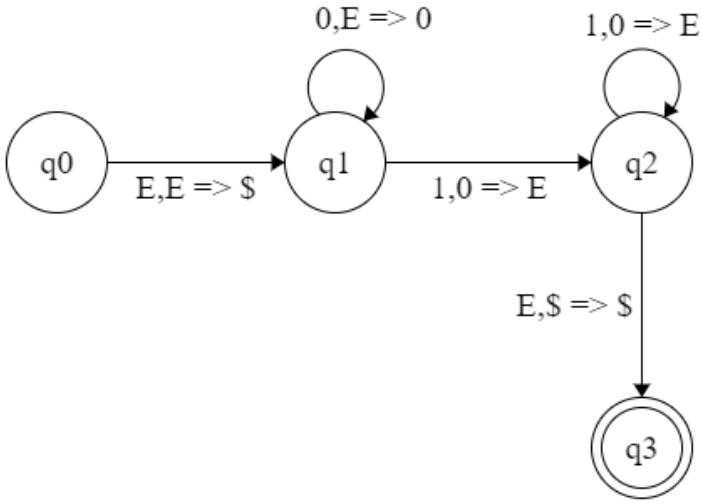
2. T es un conjunto finito de símbolos terminales

3. P es un conjunto finito de producciones

4. S es el símbolo distinguido o axioma

3 Desarrollo

En la implementación, primero planteamos un autómata que nos permita realixar la tarea de un pushdown automata.



Representación gráfica del NFA propuesto

En el autómata planteado, utilizamos el signo $ para delimitar el fin de la pila. La letra E se puede ver como la representación de epsilon. Este autómata funciona cuando en el string de entrada, consta de 0's y 1's en la misma cantidad. Dado esto, podemos decir que: $\Sigma = [0, 1]$ $L = [0^n 1^n | n >= 1]$

```
from random import randint

from numpy import append

def main():
    print("|----------- PUSHDOWN AUTOMATA ----------|")

    print("1. Autom tico\n2. Manual")
    opc = int(input())

    if(opc == 1):
        if 0<randint(0,10)<9:
            aux = randint(0,5)
            string = " "+("0"* aux)+("1"* aux)+" "
        else:
            string = " "+("0"* randint(0,5))+("1"* randint(0,5)+" "
    )

    elif(opc == 2):

        print("Ingrese la cadena a evaluar")
        string = " "+input()+" "
```

```
22
23      else:
24          print("Opci n no v lida")
25          string = ""
26
27      if(pda(string) == 1):
28          print("=============== CADENA VALIDADA ==================="
        )
29      else:
30          print("!!!!!!!!!!!!!!! CADENA INVALIDADA !!!!!!!!!!!!!!!!")
31
32  def pda(string):
33      file = open("pda.txt","w")
34      file.write("Cadena ==> ",string)
35      print(string)
36      stack = []
37      estado = 0
38      for char in string:
39          top = len(stack) - 1
40          file.write("Char = ",char)
41          print("char =" ,char)
42          file.write("Estado = ",estado)
43          print("estado =",estado)
44          file.write(stack)
45          print(stack)
46          file.write("/----------------------/")
47          print("/----------------------/")
48          if estado ==0 :
49              stack.append("$")
50              estado = 1
51              continue
52          if estado == 1:
53              if char == "0":
54                  stack.append(char)
55                  estado = 1
56              if char == "1" and stack[top] == "0":
57                  stack.pop(len(stack)-1)
58                  estado = 2
59              continue
60
61          if estado == 2:
62              if char == "1" and stack[top] == "0":
63                  stack.pop(len(stack)-1)
64                  estado = 2
65              if char == " " and stack[top] == "$":
66                  stack.pop(len(stack)-1)
67                  estado = 3
68                  stack.append("$")
69                  return 1
70              continue
71
72
73
74  main()
```

Algoritmo 1: Implementación de DFA

```
|----------- PUSHDOWN AUTOMATA ----------|
1. Automático
2. Manual
1
 01
char =
estado = 0
[]
/----------------------/
char = 0
estado = 1
['$']
/----------------------/
char = 1
estado = 1
['$', '0']
/----------------------/
char =
estado = 2
['$']
/----------------------/
============== CADENA VALIDADA ==================
PS E:\4to semestre\Teoría de la computación\Tablero>
```

```
|----------- PUSHDOWN AUTOMATA ----------|
1. Automático
2. Manual
2
Ingrese la cadena a evaluar
0011
 0011
char =
estado = 0
[]
/----------------------/
char = 0
estado = 1
['$']
/----------------------/
char = 0
estado = 1
['$', '0']
/----------------------/
char = 1
estado = 1
['$', '0', '0']
/----------------------/
char = 1
estado = 2
['$', '0']
/----------------------/
char =
estado = 2
['$']
/----------------------/
============== CADENA VALIDADA ==================
```

4 Conclusión

Gracias a este programa, pude terminar de comprender el tema de Pushdown automata. En general me pareció un ejercicio bastante sencillo, tomando en cuenta que previamente ya había formulado una implementación para un autómata, aunque sin la peculiaridad de la pila. Con los programas previos fue mucho más fácil guiarme tanto a la hora de plantear el problema, como a la hora de escribir el código.

References

[1] iencias de la Computación. (2008). Autómatas de Pila. Recuperado 5 de junio de 2022, de https://users.exa.unicen.edu.ar/catedras/ccomp1/Apunte4.pdf

[2] iencias de la Computación. (2008). Grmáticas Libre de Contexto. Recuperado 5 de junio de 2022, de https://users.exa.unicen.edu.ar/catedras/ccomp1/Apunte5.pdf