Universidad La Salle

Compiladores

Informe de la Práctica 04

Autómatas y Expresiones Regulares

Karlo Emigdio Pacha Curimayhua

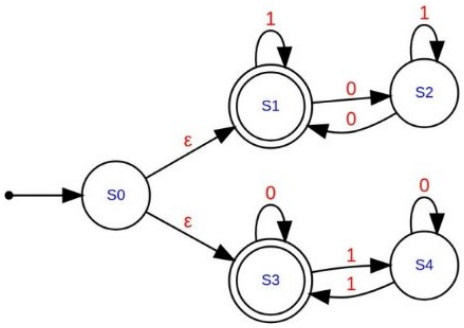
Sexto Semestre - Ingeniería de Software

2023

Ejercicio 1

Enunciado

Convierta el siguiente AFND a un AFD. Detalle todo el desarrollo, puede incluir fotos si lo hizo en papel. (7 puntos)



Solución

Tabla de transiciones AFND

States	0	1	ϵ
>S0	$\emptyset$	$\emptyset$	{S1,S3}
*S1	S2	S1	S1
S2	S1	S2	$\emptyset$
*S3	S3	S4	S3
S4	S4	S3	$\emptyset$

Tabla de transiciones AFD

States	0	1
>*{S0,S1,S3}	{S2,S3}	{S1,S4}
*{S2,S3}	{S1,S3}	{S2,S4}
*{S1,S4}	{S2,S4}	{S1,S3}
*{S1,S3}	{S2,S3}	{S1,S4}
{S2,S4}	{S1,S4}	{S2,S3}

Tabla de transiciones AFD (Renombrando Estados)

Para tener mejor control y lectura del AFD, es recomendable renombrar los estados.

States	
>*{S0,S1,S3}	A
*{S2,S3}	B
*{S1,S4}	C
*{S1,S3}	D
{S2,S4}	E

AFD inicial

>*A	B	C
*B	D	E
*C	E	D
*D	B	C
E	C	B

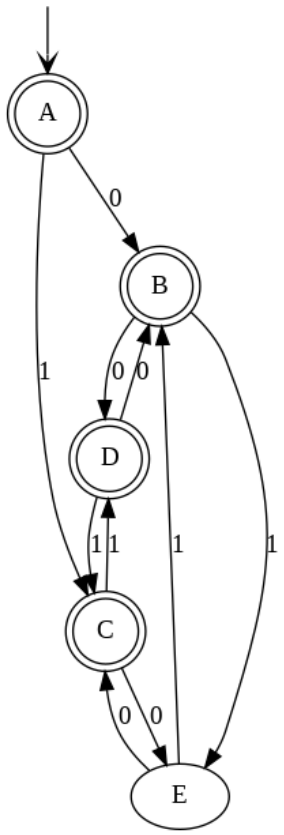


Figure 1: AFD inicial

AFD final (reducido)

Si bien el autómata parece estar terminado, se le puede aplicar una reducción. El estado D tiene transiciones a B y C que no son necesarias ya que ambos estados son estados finales o de aceptación, basta con redireccionar esas transiciones hacia A.

States	0	1
>*A	B	C
*B	A	D
*C	D	A
D	C	B

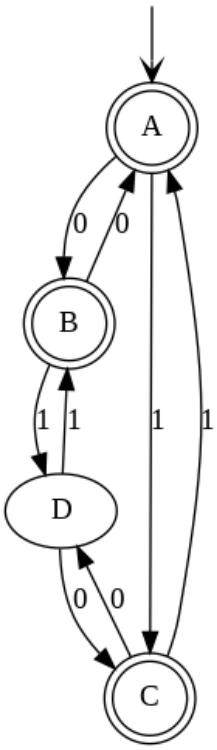


Figure 2: AFD final

Código para graficar los Autómatas

```
from graphviz import Digraph
from IPython.display import Image

def CreateAutomata(name):
  g = Digraph()
  g.node('invisible', style='invisible')
  finalStates = []
  transitions = []
  initialStates = []

  print('1-Input Initial States: ')
```

```python
while True:
  inputState = input('initial state: ')
  if inputState == '': break
  initialStates.append(inputState)

print('2-Input Final States: ')
while True:
  inputState = input('final state: ')
  if inputState == '': break
  finalStates.append(inputState)

print('3-Input Transitions: ')
while True:
  sourceState = input('source state: ')
  destinationState = input('destination state: ')
  inputValue = input('input value: ')

  if sourceState =='' and destinationState == '' and inputValue == '': break
  transitions.append([sourceState, destinationState, inputValue])

for initialState in initialStates: g.edge('invisible', initialState, dir='front', arrowhead='vee')

for finalState in finalStates: g.node(finalState, shape='circle', peripheries='2')

for transition in transitions: g.edge(transition[0], transition[1], label=transition[2])

g.render(name, format='png', view=True)
```

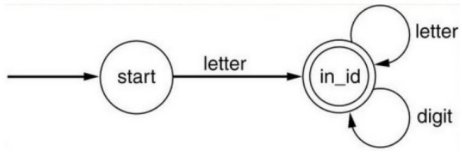
Ejercicio 2

Enunciado

Normalmente para reconocer una expresión regular se siguen los siguientes pasos:

- Convertir la expresión regular a un Automata Finito No Determinista (AFND).
- El AFND debe ser transformado a un Automata Finito Determinista (AFD).
- Finalmente este AFD, es representado mediante una tabla de transiciones.
- Se implementa un programa para reconocer las ocurrencias de una expresi on regular utilizando la tabla de transiciones.

En esta caso, le brindamos el AFD para reconocer identificadores, se le pide obtener la tabla de transiciones e implementar un programa en Python que tome esta tabla (la puedes definir estáticamente) y reconozca los identificadores de un archivo de texto (similar a la función findall de python). (11 puntos)



Solución 1

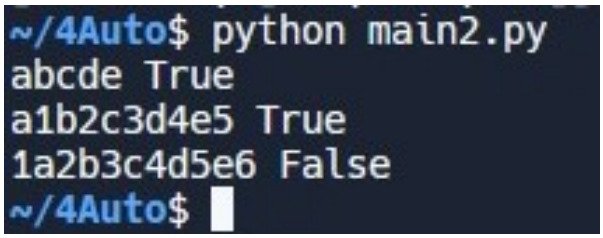
Esta solución no usa Regex en las transiciones, pero cumple satisfactoriamente con lo esperado.

Código para Strings)

```
def Automata(A, initialState, finalState, input):
  state = initialState
  i = 0
  while i < len(input):
    try:
      state = A[state,input[i]]
    except KeyError:
      return False
    i += 1
  if state == finalState: return True
  return False

A = {
  ('start', 'a'): 'in_id',
  ('start', 'b'): 'in_id',
  ('start', 'c'): 'in_id',
  ('start', 'd'): 'in_id',
  ('start', 'e'): 'in_id',
  ('start', 'f'): 'in_id',
  ('start', 'g'): 'in_id',
  ('start', 'h'): 'in_id',
  ('start', 'i'): 'in_id',
  ('start', 'j'): 'in_id',
  ('start', 'k'): 'in_id',
  ('start', 'l'): 'in_id',
  ('start', 'm'): 'in_id',
  ('start', 'n'): 'in_id',
  ('start', '  '): 'in_id',
  ('start', 'o'): 'in_id',
  ('start', 'p'): 'in_id',
  ('start', 'q'): 'in_id',
  ('start', 'r'): 'in_id',
  ('start', 's'): 'in_id',
  ('start', 't'): 'in_id',
  ('start', 'u'): 'in_id',
  ('start', 'v'): 'in_id',
  ('start', 'w'): 'in_id',
  ('start', 'x'): 'in_id',
  ('start', 'y'): 'in_id',
  ('start', 'z'): 'in_id',
  ('start', 'A'): 'in_id',
  ('start', 'B'): 'in_id',
```

```
('start', 'C'): 'in_id',
('start', 'D'): 'in_id',
('start', 'E'): 'in_id',
('start', 'F'): 'in_id',
('start', 'G'): 'in_id',
('start', 'H'): 'in_id',
('start', 'I'): 'in_id',
('start', 'J'): 'in_id',
('start', 'K'): 'in_id',
('start', 'L'): 'in_id',
('start', 'M'): 'in_id',
('start', 'N'): 'in_id',
('start', '  '): 'in_id',
('start', 'O'): 'in_id',
('start', 'P'): 'in_id',
('start', 'Q'): 'in_id',
('start', 'R'): 'in_id',
('start', 'S'): 'in_id',
('start', 'T'): 'in_id',
('start', 'U'): 'in_id',
('start', 'V'): 'in_id',
('start', 'W'): 'in_id',
('start', 'X'): 'in_id',
('start', 'Y'): 'in_id',
('start', 'Z'): 'in_id',
#------------------------
('in_id', '0'): 'in_id',
('in_id', '1'): 'in_id',
('in_id', '2'): 'in_id',
('in_id', '3'): 'in_id',
('in_id', '4'): 'in_id',
('in_id', '5'): 'in_id',
('in_id', '6'): 'in_id',
('in_id', '7'): 'in_id',
('in_id', '8'): 'in_id',
('in_id', '9'): 'in_id',
#------------------------
('in_id', 'a'): 'in_id',
('in_id', 'b'): 'in_id',
('in_id', 'c'): 'in_id',
('in_id', 'd'): 'in_id',
('in_id', 'e'): 'in_id',
('in_id', 'f'): 'in_id',
('in_id', 'g'): 'in_id',
('in_id', 'h'): 'in_id',
('in_id', 'i'): 'in_id',
('in_id', 'j'): 'in_id',
('in_id', 'k'): 'in_id',
('in_id', 'l'): 'in_id',
('in_id', 'm'): 'in_id',
('in_id', 'n'): 'in_id',
('in_id', '  '): 'in_id',
('in_id', 'o'): 'in_id',
('in_id', 'p'): 'in_id',
('in_id', 'q'): 'in_id',
('in_id', 'r'): 'in_id',
('in_id', 's'): 'in_id',
('in_id', 't'): 'in_id',
```

```
  ('in_id', 'u'): 'in_id',
  ('in_id', 'v'): 'in_id',
  ('in_id', 'w'): 'in_id',
  ('in_id', 'x'): 'in_id',
  ('in_id', 'y'): 'in_id',
  ('in_id', 'z'): 'in_id',
  ('in_id', 'A'): 'in_id',
  ('in_id', 'B'): 'in_id',
  ('in_id', 'C'): 'in_id',
  ('in_id', 'D'): 'in_id',
  ('in_id', 'E'): 'in_id',
  ('in_id', 'F'): 'in_id',
  ('in_id', 'G'): 'in_id',
  ('in_id', 'H'): 'in_id',
  ('in_id', 'I'): 'in_id',
  ('in_id', 'J'): 'in_id',
  ('in_id', 'K'): 'in_id',
  ('in_id', 'L'): 'in_id',
  ('in_id', 'M'): 'in_id',
  ('in_id', 'N'): 'in_id',
  ('in_id', '  '): 'in_id',
  ('in_id', 'O'): 'in_id',
  ('in_id', 'P'): 'in_id',
  ('in_id', 'Q'): 'in_id',
  ('in_id', 'R'): 'in_id',
  ('in_id', 'S'): 'in_id',
  ('in_id', 'T'): 'in_id',
  ('in_id', 'U'): 'in_id',
  ('in_id', 'V'): 'in_id',
  ('in_id', 'W'): 'in_id',
  ('in_id', 'X'): 'in_id',
  ('in_id', 'Y'): 'in_id',
  ('in_id', 'Z'): 'in_id'
}

initialState = 'start'
finalState = 'in_id'

inputString = 'abcde'
result = Automata(A, initialState, finalState, inputString)
print(inputString, result)

inputString = 'a1b2c3d4e5'
result = Automata(A, initialState, finalState, inputString)
print(inputString, result)

inputString = '1a2b3c4d5e6'
result = Automata(A, initialState, finalState, inputString)
print(inputString, result)
```



Código para lectura de archivos(ds) y salida txt

```
def Automata(A, initialState, finalState, input):
  matchs = []
  state = initialState
  i = 0
  k = 0
  l = 1
  string = ''

  while i < len(input):
    if input[i] == '\n': l += 1
    try:
      state = A[state,input[i]]
      string += input[i]
    except KeyError:
      if string != '':
        matchs.append([string, [k, i - 1,l]])
        string = ''
      k = i + 1
      state = initialState
    i += 1
  if state == finalState and string != '':
    matchs.append([string, [k, i - 1,l]])
  return matchs

A = {
  ('start', 'a'): 'in_id',
  ('start', 'b'): 'in_id',
  ('start', 'c'): 'in_id',
  ('start', 'd'): 'in_id',
  ('start', 'e'): 'in_id',
  ('start', 'f'): 'in_id',
  ('start', 'g'): 'in_id',
  ('start', 'h'): 'in_id',
  ('start', 'i'): 'in_id',
  ('start', 'j'): 'in_id',
  ('start', 'k'): 'in_id',
  ('start', 'l'): 'in_id',
  ('start', 'm'): 'in_id',
  ('start', 'n'): 'in_id',
  ('start', '  '): 'in_id',
  ('start', 'o'): 'in_id',
  ('start', 'p'): 'in_id',
  ('start', 'q'): 'in_id',
  ('start', 'r'): 'in_id',
  ('start', 's'): 'in_id',
  ('start', 't'): 'in_id',
```

```
('start', 'u'): 'in_id',
('start', 'v'): 'in_id',
('start', 'w'): 'in_id',
('start', 'x'): 'in_id',
('start', 'y'): 'in_id',
('start', 'z'): 'in_id',
('start', 'A'): 'in_id',
('start', 'B'): 'in_id',
('start', 'C'): 'in_id',
('start', 'D'): 'in_id',
('start', 'E'): 'in_id',
('start', 'F'): 'in_id',
('start', 'G'): 'in_id',
('start', 'H'): 'in_id',
('start', 'I'): 'in_id',
('start', 'J'): 'in_id',
('start', 'K'): 'in_id',
('start', 'L'): 'in_id',
('start', 'M'): 'in_id',
('start', 'N'): 'in_id',
('start', '  '): 'in_id',
('start', 'O'): 'in_id',
('start', 'P'): 'in_id',
('start', 'Q'): 'in_id',
('start', 'R'): 'in_id',
('start', 'S'): 'in_id',
('start', 'T'): 'in_id',
('start', 'U'): 'in_id',
('start', 'V'): 'in_id',
('start', 'W'): 'in_id',
('start', 'X'): 'in_id',
('start', 'Y'): 'in_id',
('start', 'Z'): 'in_id',
#-----------------------
('in_id', '0'): 'in_id',
('in_id', '1'): 'in_id',
('in_id', '2'): 'in_id',
('in_id', '3'): 'in_id',
('in_id', '4'): 'in_id',
('in_id', '5'): 'in_id',
('in_id', '6'): 'in_id',
('in_id', '7'): 'in_id',
('in_id', '8'): 'in_id',
('in_id', '9'): 'in_id',
#-----------------------
('in_id', 'a'): 'in_id',
('in_id', 'b'): 'in_id',
('in_id', 'c'): 'in_id',
('in_id', 'd'): 'in_id',
('in_id', 'e'): 'in_id',
('in_id', 'f'): 'in_id',
('in_id', 'g'): 'in_id',
('in_id', 'h'): 'in_id',
('in_id', 'i'): 'in_id',
('in_id', 'j'): 'in_id',
('in_id', 'k'): 'in_id',
('in_id', 'l'): 'in_id',
('in_id', 'm'): 'in_id',
```

```
  ('in_id', 'n'): 'in_id',
  ('in_id', '  '): 'in_id',
  ('in_id', 'o'): 'in_id',
  ('in_id', 'p'): 'in_id',
  ('in_id', 'q'): 'in_id',
  ('in_id', 'r'): 'in_id',
  ('in_id', 's'): 'in_id',
  ('in_id', 't'): 'in_id',
  ('in_id', 'u'): 'in_id',
  ('in_id', 'v'): 'in_id',
  ('in_id', 'w'): 'in_id',
  ('in_id', 'x'): 'in_id',
  ('in_id', 'y'): 'in_id',
  ('in_id', 'z'): 'in_id',
  ('in_id', 'A'): 'in_id',
  ('in_id', 'B'): 'in_id',
  ('in_id', 'C'): 'in_id',
  ('in_id', 'D'): 'in_id',
  ('in_id', 'E'): 'in_id',
  ('in_id', 'F'): 'in_id',
  ('in_id', 'G'): 'in_id',
  ('in_id', 'H'): 'in_id',
  ('in_id', 'I'): 'in_id',
  ('in_id', 'J'): 'in_id',
  ('in_id', 'K'): 'in_id',
  ('in_id', 'L'): 'in_id',
  ('in_id', 'M'): 'in_id',
  ('in_id', 'N'): 'in_id',
  ('in_id', '  '): 'in_id',
  ('in_id', 'O'): 'in_id',
  ('in_id', 'P'): 'in_id',
  ('in_id', 'Q'): 'in_id',
  ('in_id', 'R'): 'in_id',
  ('in_id', 'S'): 'in_id',
  ('in_id', 'T'): 'in_id',
  ('in_id', 'U'): 'in_id',
  ('in_id', 'V'): 'in_id',
  ('in_id', 'W'): 'in_id',
  ('in_id', 'X'): 'in_id',
  ('in_id', 'Y'): 'in_id',
  ('in_id', 'Z'): 'in_id'
}

initialState = 'start'
finalState = 'in_id'

data = ''

with open('example.ds', 'r') as file:
  data = file.read()
tokens = Automata(A, initialState, finalState, data)

out = open("out1.txt", "w")
for token in tokens:
  print(token)
  out.write(str(token) + "\n")
out.close()
```

```
print('-----------------------------------------')

with open('example2.ds', 'r') as file:
  data = file.read()
tokens = Automata(A, initialState, finalState, data)

out = open("out2.txt", "w")
for token in tokens:
  print(token)
  out.write(str(token) + "\n")
out.close()
```

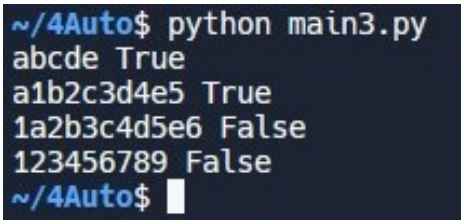
Solución 2

Esta solución usa Regex en las transiciones y cumple satisfactoriamente con lo esperado, aunque al seguir la lógica del código proporcionado, tiene mayor gasto de recursos y tiempo.

Código para Strings

```
import re

def Automata(A, initialState, finalState, input):
  state = initialState
  keys = list(A.keys())
  i = 0
  while i < len(input):
    j = GetKey(keys,state,input[i])
    if j == 0: return False
    state = A[state, keys[j-1][1]]
    i += 1

  if state == finalState: return True
  return False

def GetKey(keys,state,input):
  i = 0
  for key in keys:
    if key[0] == state and re.search(key[1],input): return i+1
    i+=1
  return 0

A = {
  ('start', r'[a-zA-Z]'): 'in_id',
  ('in_id', r'\d'): 'in_id',
  ('in_id', r'[a-zA-Z]'): 'in_id'
}


initialState = 'start'
finalState = 'in_id'

inputString = 'abcde'
result = Automata(A, initialState, finalState, inputString)
print(inputString, result)

inputString = 'a1b2c3d4e5'
```

```python
result = Automata(A, initialState, finalState, inputString)
print(inputString, result)

inputString = '1a2b3c4d5e6'
result = Automata(A, initialState, finalState, inputString)
print(inputString, result)

inputString = '123456789'
result = Automata(A, initialState, finalState, inputString)
print(inputString, result)
```



Código para lectura de archivos(ds) y salida txt

```python
import re

def Automata(A, initialState, finalState, input):
  matchs = []
  state = initialState
  keys = list(A.keys())
  i = 0
  j = 0
  k = 0
  l = 1
  string = ''

  while i < len(input):
    if input[i] == '\n': l += 1
    j = GetKey(keys, state, input[i])

    if j == 0:
      if string != '':
        matchs.append([string, [k, i - 1,l]])
        string = ''
      k = i + 1
      state = initialState
    else:
      state = A[state, keys[j - 1][1]]
      string += input[i]

    i += 1

  if state == finalState and string != '':
    matchs.append([string, [k, i - 1,l]])
  return matchs


def GetKey(keys, state, input):
```

```
  i = 0
  for key in keys:
    if key[0] == state and re.search(key[1], input): return i + 1
    i += 1
  return 0


A = {
  ('start', r'[a-zA-Z]'): 'in_id',
  ('in_id', r'\d'): 'in_id',
  ('in_id', r'[a-zA-Z]'): 'in_id'
}

initialState = 'start'
finalState = 'in_id'

data = ''

with open('example.ds', 'r') as file:
  data = file.read()
tokens = Automata(A, initialState, finalState, data)

out = open("out1.txt", "w")
for token in tokens:
  print(token)
  out.write(str(token) + "\n")
out.close()

print('-----------------------------------------')

with open('example2.ds', 'r') as file:
  data = file.read()
tokens = Automata(A, initialState, finalState, data)

out = open("out2.txt", "w")
for token in tokens:
  print(token)
  out.write(str(token) + "\n")
out.close()
```

Entrada y Salida de archivos para las soluciones completas

Entrada 1

```
func Square (a) {
  return a*a
}

func main() {
    Var a2 = 0
    var b3 = 10.5
    var cCaA4 = null
    var BB5 = "abc"
    var S6 = 'abc'
    var fA = false

    if a < b & c == null {
        for i = 0, b, 1 {
```

```
            a += 1
        }
    }

    Square(a)
}
```

Salida 1

```
['func', [0, 3, 1]]
['Square', [5, 10, 1]]
['a', [13, 13, 1]]
['return', [20, 25, 2]]
['a', [27, 27, 2]]
['a', [29, 29, 3]]
['func', [34, 37, 5]]
['main', [39, 42, 5]]
['Var', [52, 54, 6]]
['a2', [56, 57, 6]]
['var', [67, 69, 7]]
['b3', [71, 72, 7]]
['var', [85, 87, 8]]
['cCaA4', [89, 93, 8]]
['null', [97, 100, 9]]
['var', [106, 108, 9]]
['BB5', [110, 112, 9]]
['abc', [117, 119, 9]]
['var', [126, 128, 10]]
['S6', [130, 131, 10]]
['abc', [136, 138, 10]]
['var', [145, 147, 11]]
['fA', [149, 150, 11]]
['false', [154, 158, 12]]
['if', [165, 166, 13]]
['a', [168, 168, 13]]
['b', [172, 172, 13]]
['c', [176, 176, 13]]
['null', [181, 184, 13]]
['for', [196, 198, 14]]
['i', [200, 200, 14]]
['b', [207, 207, 14]]
['a', [226, 226, 15]]
['Square', [258, 263, 19]]
['a', [265, 265, 19]]
```

Entrada 2

```
func main() {
    print(1)
    var a = read(2)
    var 1MiIdentificador = 10
    var mi_segun1do_identificador = 20
    var a1 = true
    var 1en1tero = 123
    var 1flotante = 1.23
    var b3ooleano = true
    var 1nu2lo = null

```

```
    return 0
}
```

Salida 2

```
['func', [0, 3, 1]]
['main', [5, 8, 1]]
['print', [18, 22, 2]]
['var', [31, 33, 3]]
['a', [35, 35, 3]]
['read', [39, 42, 3]]
['var', [51, 53, 4]]
['MiIdentificador', [56, 70, 4]]
['var', [81, 83, 5]]
['mi', [85, 86, 5]]
['segun1do', [88, 95, 5]]
['identificador', [97, 109, 5]]
['var', [120, 122, 6]]
['a1', [124, 125, 6]]
['true', [129, 132, 7]]
['var', [138, 140, 7]]
['en1tero', [143, 149, 7]]
['var', [161, 163, 8]]
['flotante', [166, 173, 8]]
['var', [186, 188, 9]]
['b3ooleano', [190, 198, 9]]
['true', [202, 205, 10]]
['var', [211, 213, 10]]
['nu2lo', [216, 220, 10]]
['null', [224, 227, 11]]
['return', [234, 239, 12]]
```

Repositorio GitHub