Practica1

February 9, 2024

1 Práctica 1

1.1 Expresiones regulares y lexers

1.1.1 1. Realiza un sistema para identificar patrones dentro de cadenas textuales a partir de expresiones regulares. Para esto realiza:

a. Realiza una clase que simule a un autómata finito (puede ser no determinístico)

```
class Automata(object):
    def __init__(self, states, initial, final, transitions):
    """Clase para objeto Autómata finito"""
    ...
    def accepts(self, cadena):
    """Método que acepta o rechaza una cadena según si esta pertenece o no al lenguaje"""
    ...
    def match_token(self, texto):
    """Regresa la posición de un tóken cuando este es aceptado por el autómata"""
    ...
```

b. Define la función de construcción del autómata a partir de los patrones de expresiones regu

```
def compile(regex):
     """Función que regresa un autómata finito contruyéndolo a partir de una expresión regular
     ...
    return Automata
```

c. Prueba el autómata con el siguiente patrón y sobre las siguientes cadenas:

[10]:
```
from automata import compile, tokenize

#Construye el autómata de la regex
pattern = compile('niña|os?')

#Cadenas que acepta
print(pattern.accepts('niño'))
print(pattern.accepts('niña'))
print(pattern.accepts('niñas'))
print(pattern.accepts('niños'))
print(pattern.accepts('niñs'))
```

```
True
True
True
True
False
```

d. Utiliza el autómata para localizar los patrones que coinciden en el siguiente texto (la fun

[12]:
```
#Tokenización del texto
text = tokenize('Las niñas extranjeras jugaban con el niño y la niña en el␣
 ↪patio.')

#Encuentra las correspondecias
matches = pattern.match_token(text)

#Imprime índices y tokenes
#que cumplen el patrón dentro del texto
for i in matches:
    print(i, text[i])
```

```
1 niñas
6 niño
9 niña
```

1.1.2 2. Utilizando lex, construye un lexer que pueda identificar los siguientes tipos de tókens

1. Número enteros (INT) como 0, 1, 2, etc.
2. Número de punto flotante (FLOAT) como 0.1, 1.4e-3, 3e+10, etc.
3. Palabras clave (KEYWORD), enfocándose en if, then y else.
4. Operadores (RELOP) +, *, **, <, >, =, <=, >=.
5. Booleanos (BOOL) True y False.
6. Delimitadores (DELIM) { } ;

Por ejemplo, el siguiente código:

```
if x<=0.5e+10 then {
    if x>= 1e-10 then {
        if x < 3 then {
            True
        }
    }
    else {
        False
    }
}
```

Obtendrá como resultado algo como:

```
(if: KEYWORD)
(x: ID)
```

```
(<= RELOP)
(0.5e+10: FLOAT)
(then: KEYWORD)
({: DELIM)
(if: KEYWORD)
(x: ID)
(>= RELOP)
(1e-10: FLOAT)
(then: KEYWORD)
({: DELIM)
(if: KEYWORD)
(x: ID)
(< RELOP)
(3: INT)
(then: KEYWORD)
({: DELIM)
(True: BOOL)
(}: DELIM)
(}: DELIM)
(else: KEYWORD)
({: DELIM)
(False: BOOL)
(}: DELIM)
(}: DELIM)
```