Computación y Estructuras Discretas III

Andrés A. Aristizábal P.
aaaristizabal@icesi.edu.co

Departamento de Computación y Sistemas Inteligentes

UNIVERSIDAD ICESI

2024-2

Agenda del día

1. **Regular languages and Automata theory**
    - Implementing FST with Pyformlang
    - Implementing grammars with Pyformlang
    - Exercises

Agenda del día

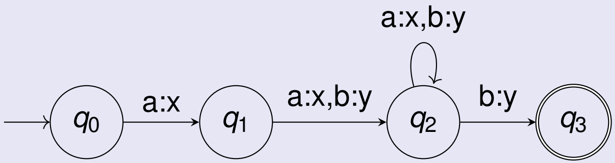
1. **Regular languages and Automata theory**
   - Implementing FST with Pyformlang
   - Implementing grammars with Pyformlang
   - Exercises

Implementing FST with Pyformlang

Given $\Sigma = \{a, b\}$ build build a FST that translates strings represented by $a(b \cup a)^{+}b$ into $x(y \cup x)^{+}y$

Implementing FST with Pyformlang

Given $\Sigma = \{a, b\}$ build build a FST that translates strings represented by $a(b \cup a)^{+}b$ into $x(y \cup x)^{+}y$



Implementing FST with Pyformlang

Let’s build it in pyformlang:

Implementing FST with Pyformlang

Let’s build it in pyformlang:

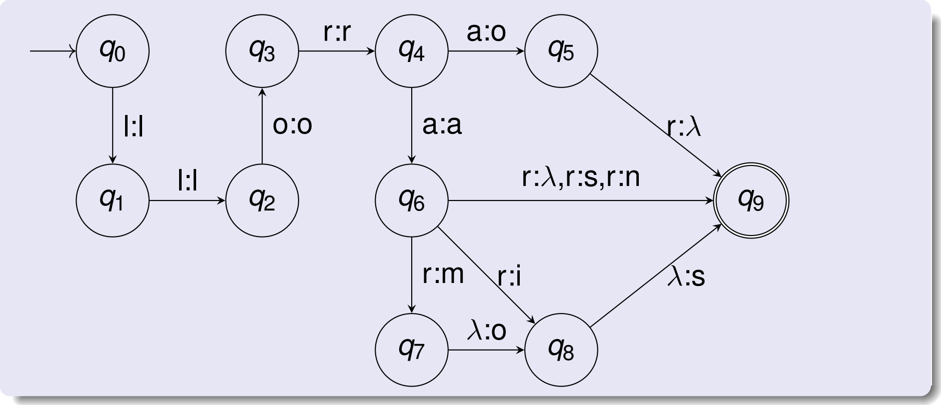
```
from pyformlang.fst import FST

transducer = FST()

transducer.add_transitions([(’q0’,’a’,’q1’,[’x’]),
                     (’q1’,’a’,’q2’,[’x’]),
                     (’q1’,’b’,’q2’,[’y’]),
                     (’q2’,’a’,’q2’,[’x’]),
                     (’q2’,’b’,’q2’,[’y’]),
                     (’q2’,’b’,’q3’,[’y’])])

transducer.add_start_state(’q0’)
transducer.add_final_state(’q3’)

print("".join(list(transducer.translate(’ababb’))[0]))
print("".join(list(transducer.translate(’aaaaab’))[0]))
print("".join(list(transducer.translate(’abb’))[0]))
```

Implementing FST with Pyformlang

Build a FST that given the verb *llorar* in infinitive gives you all the possible conjugations

Implementing FST with Pyformlang

Build a FST that given the verb *llorar* in infinitive gives you all the possible conjugations



Implementing FST with Pyformlang

Let’s build it in pyformlang:

Implementing FST with Pyformlang

Let’s build it in pyformlang:

```
from pyformlang.fst import FST
conjug = FST()
conjug.add_transitions([('q0','l','q1',['l']),
                  ('q1','l','q2',['l']),
                  ('q2','o','q3',['o']),
                  ('q3','r','q4',['r']),
                  ('q4','a','q5',['o']),
                  ('q4','a','q6',['a']),
                  ('q6','r','q7',['m']),
                  ('q6','r','q8',['i']),
                  ('q7','epsilon','q8',['o']),
                  ('q8','epsilon','q9',['s']),
                  ('q5','r','q9',['']),
                  ('q6','r','q9',['']),
                  ('q6','r','q9',['s']),
                  ('q6','r','q9',['n'])])
conjug.add_start_state('q0')
conjug.add_final_state('q9')
```

Implementing FST with Pyformlang

Let’s evaluate our FST:

Implementing FST with Pyformlang

Let’s evaluate our FST:

```
print(list(map(lambda x:
    "".join(x),list(conjug.translate(’llorar’)))))
```

Agenda del día

1. **Regular languages and Automata theory**
   - Implementing FST with Pyformlang
   - Implementing grammars with Pyformlang
   - Exercises

Implementing grammars with Pyformlang

Construct a grammar that generates the language $a^* b^*$

Implementing grammars with Pyformlang

Construct a grammar that generates the language $a^* b^*$

$$\left\{ \begin{array}{l} S \to aS \,|\, A \\ A \to bA \,|\, \lambda \end{array} \right.$$

Implementing grammars with Pyformlang

Let’s build it in Pyformlang:

Implementing grammars with Pyformlang

Let’s build it in Pyformlang:

```
from pyformlang.cfg import Production, Variable,
    Terminal, CFG, Epsilon
var_S = Variable(’S’)
var_A = Variable(’A’)
ter_a = Terminal(’a’)
ter_b = Terminal(’b’)
prod_0 = Production(var_S,[ter_a,var_S])
prod_1 = Production(var_S,[var_A])
prod_2 = Production(var_A,[ter_b,var_A])
prod_3 = Production(var_A,[Epsilon()])
cfg = CFG({var_S, var_A}, {ter_a, ter_b}, var_S,
    {prod_0, prod_1, prod_2, prod_3})
print(cfg.contains(’aaa’))
print(cfg.contains(’bb’))
```

Implementing grammars with Pyformlang

Another way would be:

Implementing grammars with Pyformlang

Another way would be:

```
from pyformlang.cfg import CFG
cfg1 = CFG.from_text("""
   S -> a S | A
   A -> b A | epsilon""")
print(cfg1.contains(’aaa’))
print(cfg1.contains(’bb’))
```

Agenda del día

1. **Regular languages and Automata theory**
   - Implementing FST with Pyformlang
   - Implementing grammars with Pyformlang
   - Exercises