Autómatas y Lenguajes

Práctica 1 - Autómatas Finitos

Uso de la herramienta graphviz

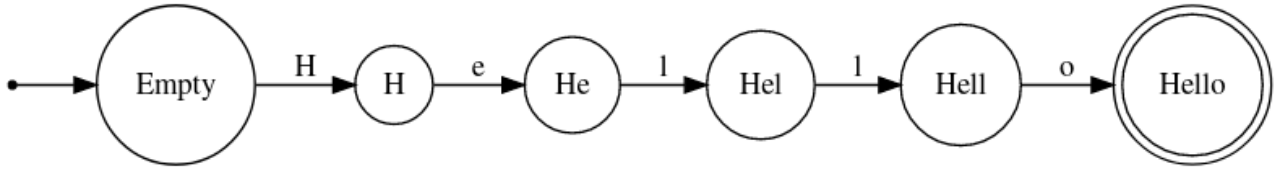
Graphviz (https://graphviz.org/)

- Herramienta Open-source de visualización de grafos.
- Usa el lenguaje **DOT.**

```
digraph {
  rankdir=LR;

  node [shape = point]; __start_point__
  Empty[shape=circle]
  H[shape=circle]
  He[shape=circle]
  Hel[shape=circle]
  Hell[shape=circle]
  Hello[shape=doublecircle]

  __start_point__ -> Empty
  Empty -> H[label="H"]
  H -> He[label="e"]
  He -> Hel[label="l"]
  Hel -> Hell[label="l"]
  Hell -> Hello[label="o"]
}
```



Uso para la práctica.

- Método **write_dot** en **utils.py**
- Ejemplo **imprimiendo por terminal**:

```
from automata.automaton import FiniteAutomaton
from automata.utils import AutomataFormat, deterministic_automata_isomorphism, write_dot


class TestTransform(ABC, unittest.TestCase):
    """Base class for string acceptance tests."""

    def _check_transform(self, automaton, expected):
        """Test that the transformed automaton is as the expected one."""
        transformed = automaton.to_deterministic()
        equiv_map = deterministic_automata_isomorphism(expected, transformed)

        print("Automaton")
        print(write_dot(automaton))

        print("Expected Automaton")
        print(write_dot(expected))

        print("Transformed Automaton")
        print(write_dot(transformed))
```

Uso para la práctica.

- Ejemplo imprimiendo por terminal:

```
Automaton
digraph {
  rankdir=LR;

  node [shape = point]; __start_point__
  q0[shape=circle]
  qf[shape=doublecircle]

  __start_point__ -> q0
  q0 -> qf[label="0"]
  qf -> qf[label="1"]
}
```

```
Expected Automaton
digraph {
  rankdir=LR;

  node [shape = point]; __start_point__
  q0[shape=circle]
  qf[shape=doublecircle]
  empty[shape=circle]

  __start_point__ -> q0
  q0 -> qf[label="0"]
  q0 -> empty[label="1"]
  qf -> empty[label="0"]
  qf -> qf[label="1"]
  empty -> empty[label="0"]
  empty -> empty[label="1"]
}
```

```
Transformed Automaton
digraph {
  rankdir=LR;

  node [shape = point]; __start_point__
  "q0"[shape=circle]
  "qf"[shape=doublecircle]
  ""[shape=circle]

  __start_point__ -> "q0"
  "q0" -> "qf"[label="0"]
  "q0" -> ""[label="1"]
  "qf" -> ""[label="0"]
  "qf" -> "qf"[label="1"]
  "" -> ""[label="0"]
  "" -> ""[label="1"]
}
```

Copiar a archivo:
automaton.dot

Uso para la práctica

- Método **write_dot** en **utils.py**
- Ejemplo **imprimiendo por terminal**:

```
class TestTransform(ABC, unittest.TestCase):
    """Base class for string acceptance tests."""

    def _check_transform(self, automaton, expected):
        """Test that the transformed automaton is as the expected one."""
        transformed = automaton.to_deterministic()
        equiv_map = deterministic_automata_isomorphism(expected, transformed)

        with open('automata.dot', 'w') as f_dot:
            f_dot.write(write_dot(automaton))

        with open('automata_transformed.dot', 'w') as f_dot:
            f_dot.write(write_dot(transformed))

        self.assertTrue(equiv_map is not None)
```

Dot en terminal

- sudo apt install graphviz
- dot -Tpng automata.dot -o automata.png;

```
                    ~/AUTLEN/P1/automata$ python tests/test_to_deterministic.py
.
----------------------------------------------------------------------
Ran 1 test in 0.001s

OK
                    ~/AUTLEN/P1/automata$ ls
automata  automaton.dot  automaton_transformed.dot  tests
                    ~/AUTLEN/P1/automata$ dot -Tpng automaton.dot -o automaton.png
                    ~/AUTLEN/P1/automata$ dot -Tpng automaton_transformed.dot -o automaton_transformed.png
                    ~/AUTLEN/P1/automata$ ls
automata  automaton.dot  automaton.png  automaton_transformed.dot  automaton_transformed.png  tests
                    ~/AUTLEN/P1/automata$
```

Graphviz online

```
digraph {
  rankdir=LR;

  node [shape = point]; __start_point__
  q0[shape=circle]
  qf[shape=doublecircle]

  __start_point__ -> q0
  q0 -> qf[label="0"]
  qf -> qf[label="1"]
}
```

Engine: dot Format: svg