TEORÍA DE AUTÓMATAS Y LENGUAJES FORMALES

PRÁCTICA 2

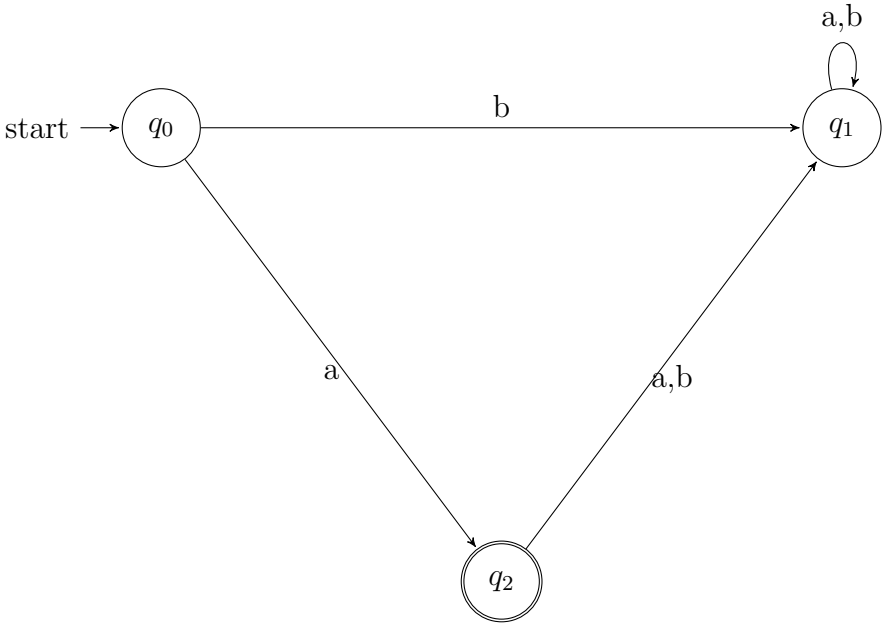
EJERCICIO 1. Consider the language over the alphabet $\{a, b\}$ only containing the string a.

a. Build a DFA that recognizes this language and rejects all those strings that do not belong to the language.

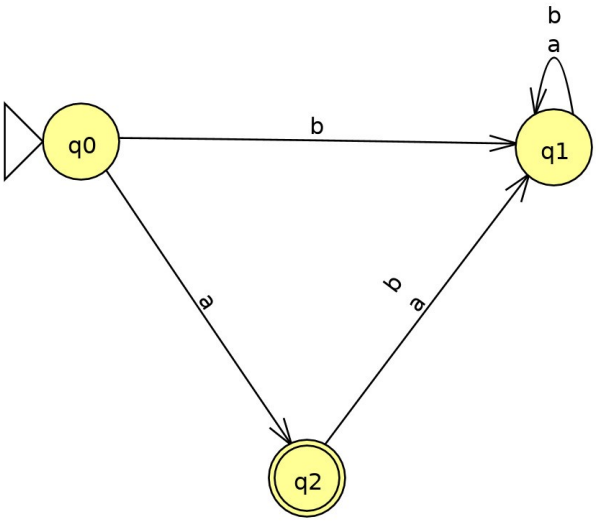
b. Test the automaton that you have created by introducing 6 chains.

Consideremos el siguiente DFA, $M = (K, \Sigma, \delta, s, F)$

$= (\{q0, q1, q2\}, \{a, b\}, \{(q0, a, q2), (q0, b, q1), (q1, a, q1), (q1, b, q1), (q2, a, q1), (q2, b, q1)\}, q0, \{q2\})$



Efectivamente, si desde el estado inicial, q0, detectamos una a, llegaremos al estado final q2, pero si la cadena es de mayor longitud, de éste pasaremos al estado q1, desde el que no habrá forma de volver a q2. Estas cadenas y aquellas empezadas en b, nunca serán aceptadas por el autómata.



Input	Result
aaaa	Reject
a	Accept
baab	Reject
baba	Reject
abab	Reject
abbba	Reject

EJERCICIO 2. Finite automaton in Octave:

a. Open the Octave finiteautomata.m script and test it with the given example (see script help) in the GitHub repository.

b. Create a JSON file that describes the automaton created in Activity 1 and test it!

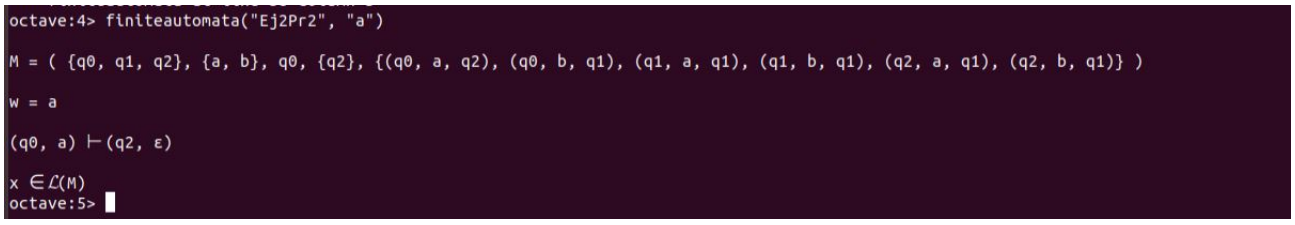
En primer lugar, probemos el script, con uno de los lenguajes ejemplo incluidos en finiteautomata.json. En concreto, veamos que la cadena "$abba$" pertenece al lenguaje definido por la expresión abb^*aa^*.

```
octave:12> finiteautomata("a*bb*aa*", "abba")

M = ( {q0, q1, q2, q3}, {a, b}, q0, {q2}, {(q0, a, q0), (q0, b, q1), (q1, a, q2), (q1, b, q1), (q2, a, q2), (q2, b, q3), (q3, a, q3), (q3, b, q3)} )

w = abba

(q0, abba) ⊢ (q0, bba) ⊢ (q1, ba) ⊢ (q1, a) ⊢ (q2, ε)

x ∈ ℒ(M)
octave:13>
```

A continuación, añadamos al código de finiteautomata.json la descripción del autómata del Ejercicio 1. Es decir, el fragmento siguiente:

```
"name" : "Ej2Pr2",
"representation" : {
"K" : ["q0", "q1", "q2"],
"A" : ["a", "b"],
"s" : "q0",
"F" : ["q2"],
"t" : [["q0", "a", "q2"],
["q0", "b", "q1"],
["q1", "a", "q1"],
["q1", "b", "q1"],
["q2", "a", "q1"],
["q2", "b", "q1"]]
}
```

Ahora ya podemos probarlo. Como sabemos, el lenguaje solo acepta la cadena "a" y, efectivamente, esto corresponde con el resultado obtenido al ejecutarlo en Octave. Todas las cadenas introducidas, exceptuando esa, son rechazadas:



```
octave:5> finiteautomata("Ej2Pr2", "baaab")

M = ( {q0, q1, q2}, {a, b}, q0, {q2}, {(q0, a, q2), (q0, b, q1), (q1, a, q1), (q1, b, q1), (q2, a, q1), (q2, b, q1)} )

w = baaab

(q0, baaab) ⊢ (q1, aaab) ⊢ (q1, aab) ⊢ (q1, ab) ⊢ (q1, b) ⊢ (q1, ε)

x ∉ ℒ(M)
octave:6>
```

```
octave:6> finiteautomata("Ej2Pr2", "b")

M = ( {q0, q1, q2}, {a, b}, q0, {q2}, {(q0, a, q2), (q0, b, q1), (q1, a, q1), (q1, b, q1), (q2, a, q1), (q2, b, q1)} )

w = b

(q0, b) ⊢ (q1, ε)

x ∉ ℒ(M)
octave:7>
```

```
octave:7> finiteautomata("Ej2Pr2", "")

M = ( {q0, q1, q2}, {a, b}, q0, {q2}, {(q0, a, q2), (q0, b, q1), (q1, a, q1), (q1, b, q1), (q2, a, q1), (q2, b, q1)} )

w =

(q0, ε)

x ∉ ℒ(M)
octave:8>
```

```
octave:9> finiteautomata("Ej2Pr2", "abababab")

M = ( {q0, q1, q2}, {a, b}, q0, {q2}, {(q0, a, q2), (q0, b, q1), (q1, a, q1), (q1, b, q1), (q2, a, q1), (q2, b, q1)} )

w = abababab

(q0, abababab) ⊢ (q2, bababab) ⊢ (q1, ababab) ⊢ (q1, babab) ⊢ (q1, abab) ⊢ (q1, bab) ⊢ (q1, ab) ⊢ (q1, b) ⊢ (q1, ε)

x ∉ ℒ(M)
octave:10>
```