NFA – Nondeterministic finite automaton: poate tranzita si fi in mai multe stari in acelasi timp

DFA – Deterministic finite automaton: poate fi intr-o singura stare

Fiecare DFA este si NFA, dar nu si viceversa.

∑ - simboluri de intrare/input

S – multimea starilor

S_0 – starea initiala

δ: S x ∑ -> S – functia de tranzitie

F – multimea starilor finale

Regex rules:

* - 0 sau mai multe aparitii

+ - 1 sau mai multe aparitii

? – 0 sau 1 aparitii

. – orice caracter

\ - escape

() – grup

[] – unul din caracterele din paranteze

^ - marcheaza inceputul stringului/liniei

$ - marcheaza sfarsitul stringului/liniei

{} – numar de repetitii

https://regex101.com/

http://madebyevan.com/fsm/

Exemplu1: ∑ = {0,1} cuvinte care se termina cu 1:

Regex: ^[01]*1$

Exemplu2: ∑ = {a,b} cuvinte care contin oricati de b intre 2 de a:

Regex: ^(ab+a)+$

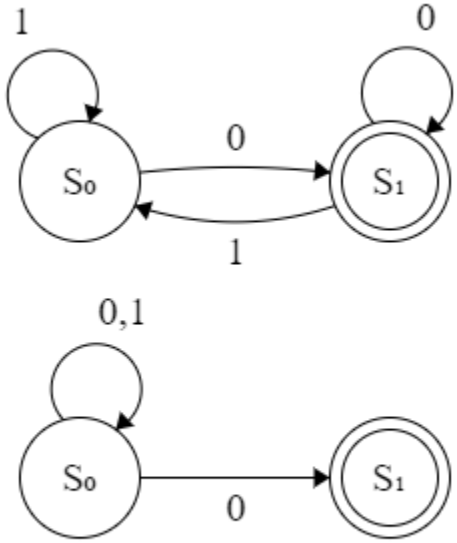
Exemplu3: ∑ = {a,b} cuvinte cu lungime divizibila cu 2

Regex: ^([ab][ab])+$

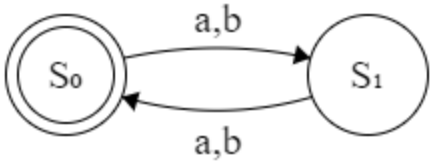
Exemplu4: ∑ = {a,b} cuvinte care contina ’a’

Regex: ^b*a[ab]*$

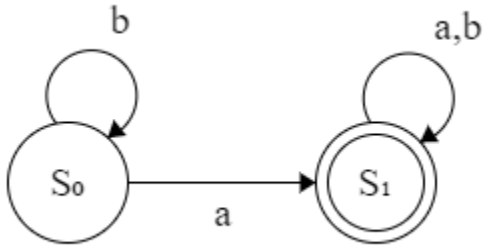
Exemplu1. DFA si NFA ∑ = {0,1} – cuvinte care se termina cu 0.



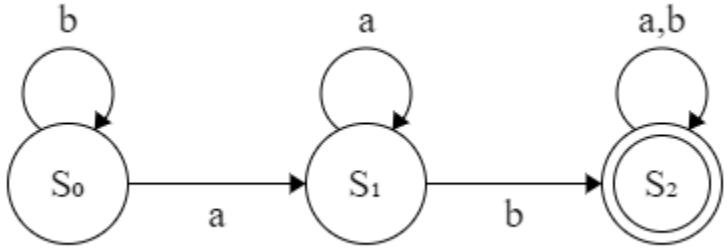
Exemplu2. DFA ∑ = {a,b} cuvinte cu lungime divizibila cu 2.



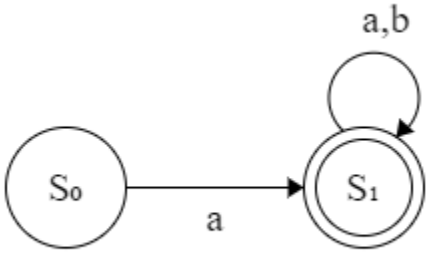
Exemplu3. DFA ∑ = {a,b} cuvinte care contin litera ’a’.



Exemplu4. DFA ∑ = {a,b} cuvinte care contin ’ab’.

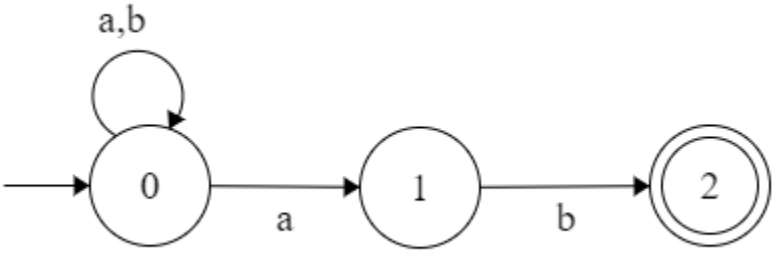


Exemplu5. DFA ∑ = {a,b} cuvinte care incep cu ’a’.



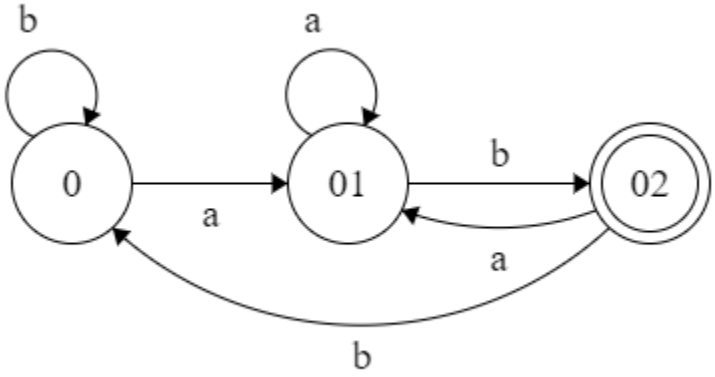
Exemplu1. Conversie NFA in DFA

NFA Form:



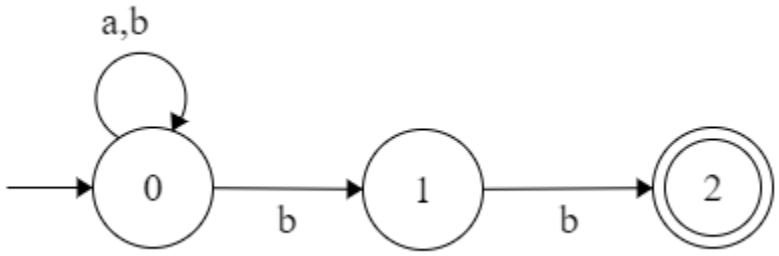
Stare	a	b
0	0,1	0
0,1	0,1	0,2
0,2	0,1	0

DFA Form:



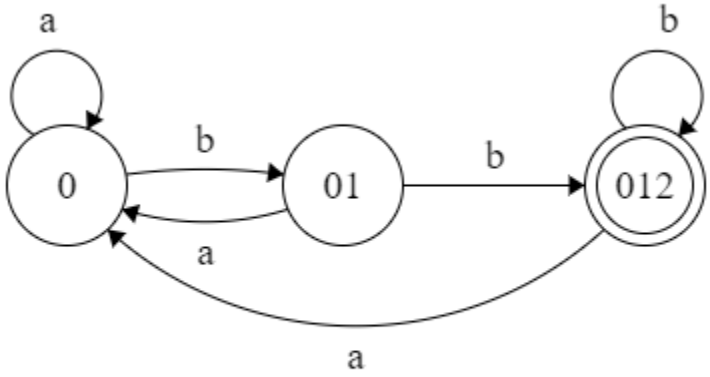
Exemplu2. Conversie NFA in DFA

NFA Form:



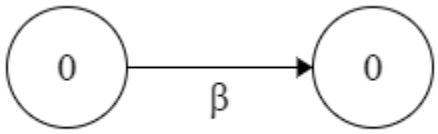
Starea	a	b
0	0	0,1
0,1	0	0,1,2
0,1,2	0	0,1,2

DFA Form:

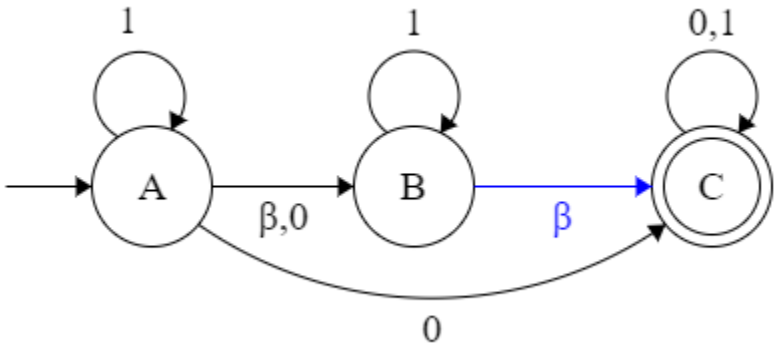


Exemplu3. Conversie NFA cu tranzitii **ε(β)** in DFA

Nu se consuma simbol



NFA Form:



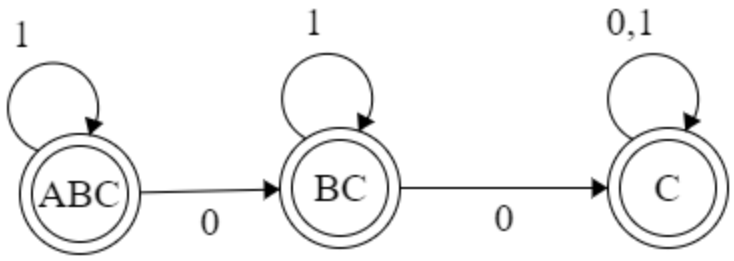
Stare	0	1	ε(β)
A	B,C	A,B,C	B,C
B	C	B,C	C
C	C	C	-

Closure(A): {A,B,C}

Closure(B): {B,C}

Closure(C): {C}

	0	1
A,B,C	B,C	A,B,C
B,C	C	B,C
C	C	C




```
import re

print("cuvinte care se termina cu 1")
test = re.match("^[01]*1$", "01001101")
print(test)
test = re.match("^[01]*1$", "0100110")
print(test)

print("cuvinte care contin oricati de b intre 2 de a")
test = re.match("^(ab+a)+$", "abbbbaabbbbbbba")
print(test)
test = re.match("^ab+a$", "baba")
print(test)

print("cuvinte cu lungime divizibila cu 2")
test = re.match("^([ab][ab])+$", "abbbba")
print(test)
test = re.match("([ab][ab])$", "ababa")
print(test)

print("cuvinte care contina 'a'")
test = re.match("^b*a+[ab]*$", "abbbba")
print(test)
test = re.match("^b*a+[ab]*$", "bbbbbbbbb")
print(test)

print("cuvinte care sa contina intre 4 si 8 caractere")
test = re.match("^([ab]){4,8}$", "abbbba")
print(test)
test = re.match("([ab]){4,8}$", "aba")
print(test)
```

Tema:

1. Scrie o expresie regulată pentru a valida un număr de telefon. Regulile sunt:

Numărul de telefon poate să conțină caracterele: 0-9, "-", "(", ")" și poate începe cu "+".

Dacă numărul de telefon începe cu "+", trebuie să urmeze cel puțin un caracter numeric.

Numărul de telefon trebuie să aibă între 8 și 15 caractere (inclusiv).

2. NFA in DFA