Formalne metode u softverskom inženjerstvu

Laboratorijska vježba – Konačni automati stanja

Zadatak 1. Realizovati `accepts` funkciju za klasu `Dfa`.

In [3]:
```
# NAPOMENE:
# __init__ je specijalna metoda koja se poziva prilikom kreiranja objekta, konstruk
# self referenca je isto kao this u Javi i nije skriveno polje, mora biti eksplicit

class Dfa:
    # Konstruktor
    def __init__(self, T, q0, F):
        self.T = T
        self.q0 = q0
        self.F = F

    # Metoda koja izvršava simulaciju rada DFA nad ulaznim nizom i vraća True ako j
    def accepts(self, w) -> bool:
        current_state = self.q0
        for s in w:
            current_state = self.T[(current_state, s)]
        return current_state in self.F
```

In [4]:
```
# Primjer korištenja klase Dfa
dfa1 = Dfa(
    T = {
      (0, 'a') : 0,
      (0, 'b') : 1,
      (1, 'a') : 1,
      (1, 'b') : 0,
    },
    q0 = 0,
    F = { 1 })

assert dfa1.accepts("") == False
assert dfa1.accepts("a") == False
assert dfa1.accepts("b") == True
assert dfa1.accepts("aa") == False
assert dfa1.accepts("ab") == True
assert dfa1.accepts("ba") == True
assert dfa1.accepts("bb") == False
```

Zadatak 2. Izučiti postojeće implementacije DKA, epsilon NKA i algoritma za minimizaciju DKA.

Zadatak 3. Realizovati i testirati funkciju kojom se izračunava DKA reprezentacija unije regularnih jezika, na osnovu postojećih DKA reprezentacija regularnih jezika.

In [ ]:
```python
def dfa_union(dfa1, dfa2):
    pass # TODO Implementirati metodu

# TODO Testirati dfa_union
```

Zadatak za vježbu: Realizovati i testirati metodu kojom se konvertuje epsilon NKA u DKA.

In [ ]:
```python
def enfa_to_dfa(alphabet, transition_function, start_state, accept_states):
    pass # TODO Implementirati
```