T.C.

SAKARYA ÜNİVERSİTESİ

BİLGİSAYAR VE BİLİŞİM BİLİMLERİ FAKÜLTESİ

BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

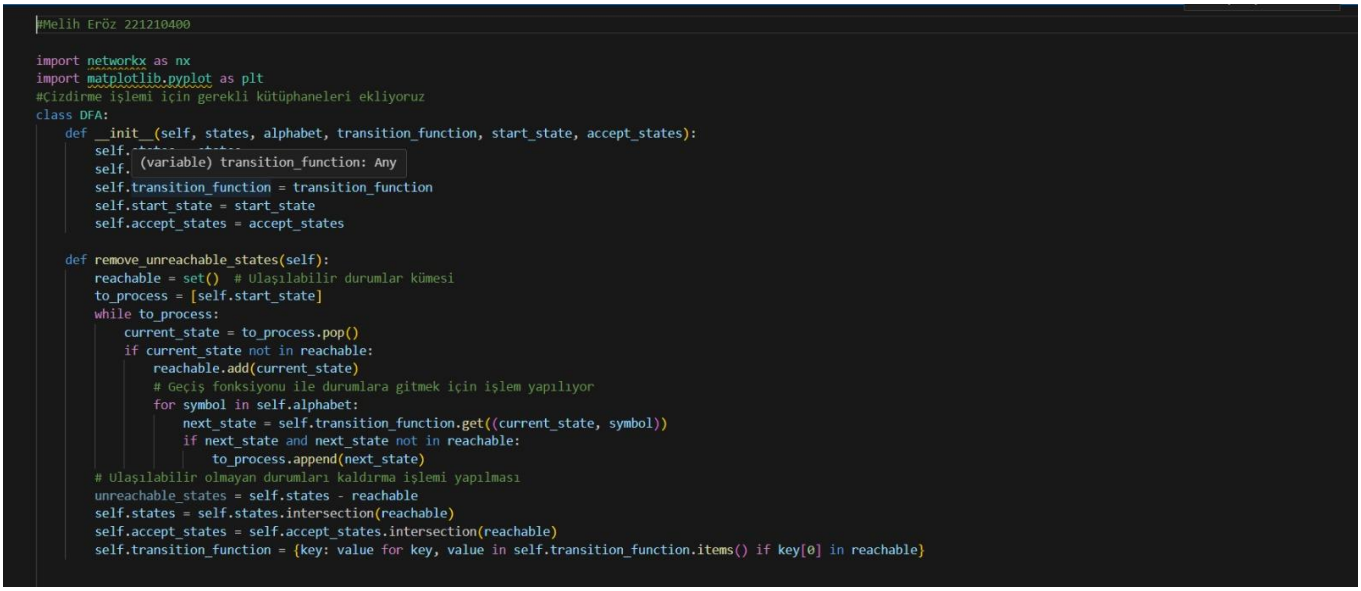
BİÇİMSEL DİLLER Ve Soyut Makineler Ödev RAPORU

Melih Eröz 1.Öğretim –B221210400- 1.A Grubu

Soru 1: Bir DFA makinesinde durum sayısını indirgemek için bir algoritma geliştiriniz ve istediğiniz bir programlama dilinde kodlayınız. (Ulaşılamayan durumların kaldırılması ve denk durumların birleştirilmesi)

Kod Özeti

Sınıf Tanımlaması ve Ulaşılamaz Durumları Kaldırma



```python
#Melih Eröz 221210400

import networkx as nx
import matplotlib.pyplot as plt
#Çizdirme işlemi için gerekli kütüphaneleri ekliyoruz
class DFA:
    def __init__(self, states, alphabet, transition_function, start_state, accept_states):
        self.states = states
        self.alphabet = alphabet
        self.transition_function = transition_function
        self.start_state = start_state
        self.accept_states = accept_states

    def remove_unreachable_states(self):
        reachable = set()  # Ulaşılabilir durumlar kümesi
        to_process = [self.start_state]
        while to_process:
            current_state = to_process.pop()
            if current_state not in reachable:
                reachable.add(current_state)
                # Geçiş fonksiyonu ile durumlara gitmek için işlem yapılıyor
                for symbol in self.alphabet:
                    next_state = self.transition_function.get((current_state, symbol))
                    if next_state and next_state not in reachable:
                        to_process.append(next_state)
        # Ulaşılabilir olmayan durumları kaldırma işlemi yapılması
        unreachable_states = self.states - reachable
        self.states = self.states.intersection(reachable)
        self.accept_states = self.accept_states.intersection(reachable)
        self.transition_function = {key: value for key, value in self.transition_function.items() if key[0] in reachable}
```

(variable) transition_function: Any

- **`states`**: DFA'nın durum kümesi. Örneğin: `{'q0', 'q1', 'q2', ...}`.
- **`alphabet`**: DFA'nın kabul ettiği giriş alfabesi. Örneğin: `{'a', 'b'}`.
- **`transition_function`**: DFA'nın geçiş fonksiyonu. Bir durum ve sembol çiftini alarak bir sonraki duruma gider. Örneğin: `{('q0', 'a'): 'q1'}`.
- **`start_state`**: DFA'nın başlangıç durumu. Örneğin: `'q0'`.
- **`accept_states`**: DFA'nın kabul durumu/durumları. Örneğin: `{'q6'}`.
- **Amaç**: DFA'da ulaşılabilir olmayan durumları kaldırmak.
- **Nasıl Çalışır?**

1. Başlangıç durumundan (`start_state`) ulaşılabilecek tüm durumları `reachable` kümesine ekler.
2. Her bir durum için, `transition_function` kullanılarak geçiş yapılan durumları kontrol eder ve bunları da küme ekler.
3. Ulaşılamaz durumları `states` ve `accept_states` kümelerinden çıkarır.

Durumları Geçiş Fonksiyonlarına Göre Gruplama ve DFA İndirgeme

```python
def group_states_by_transition(self):
    groups = {}  # Durum kümeleri
    state_mapping = {}  # Durumların hangi kümeye ait olduğunu gösterir

    # Durumları geçişlerine göre gruplama işlemi yapılması
    for state in self.states:
        transition_signature = tuple(self.transition_function.get((state, symbol), None) for symbol in self.alphabet)

        # Aynı geçiş fonksiyonuna sahip olan durumları gruplandırma işlemi
        if transition_signature not in groups:
            groups[transition_signature] = set()
        groups[transition_signature].add(state)
        state_mapping[state] = transition_signature

    # Grupları döndürme işlemi
    return groups, state_mapping

def minimize(self):
    self.remove_unreachable_states()
    groups, state_mapping = self.group_states_by_transition()

    # Grupları güncelleme işlemi
    self.groups = groups
    self.state_mapping = state_mapping
```

- **Amaç**: Durumları geçiş fonksiyonlarına göre gruplandırmak.
- **Nasıl Çalışır?**
    1. Her bir durum için, tüm semboller (`alphabet`) üzerinden hangi durumlara geçtiğini kontrol eder (`transition_signature`).
    2. Aynı geçiş desenine sahip olan durumları bir gruba koyar.
    3. `groups` ve `state_mapping` (her bir durumun hangi gruba ait olduğunu gösterir) döndürülür.
- DFA'yı indirgemek için iki adım gerçekleştirilir:
    1. **Ulaşılamaz durumlar kaldırılır.**
    2. **Durumlar, geçiş fonksiyonlarına göre gruplandırılır.**
- Bu süreç sonunda, `groups` ve `state_mapping` güncellenir.

DFA Çizimi

```python
def plot_dfa(self):
    G = nx.DiGraph()
    # Durumları ve geçişleri grafiğe eklediğim yer
    for (state, symbol), next_state in self.transition_function.items():
        G.add_edge(state, next_state, label=symbol)

    pos = nx.spring_layout(G)
    plt.figure(figsize=(8, 6))
    nx.draw(G, pos, with_labels=True, node_size=3000, node_color='lightblue', font_size=16, font_weight='bold', arrowsize=20)
    labels = nx.get_edge_attributes(G, 'label')
    nx.draw_networkx_edge_labels(G, pos, edge_labels=labels)

    # Başlangıç durumu işaretleniyor
    plt.text(pos[self.start_state][0], pos[self.start_state][1] + 0.1, 'Start', horizontalalignment='center', fontsize=12, color='green')
    # Kabul durumu işaretleniyor
    for state in self.accept_states:
        plt.text(pos[state][0], pos[state][1] - 0.1, 'Accept', horizontalalignment='center', fontsize=12, color='red')

    plt.title("Başlangıç DFA Durumu")
    plt.show()
```

- **Amaç**: DFA'nın görselleştirilmesi.
- **Nasıl Çalışır?**

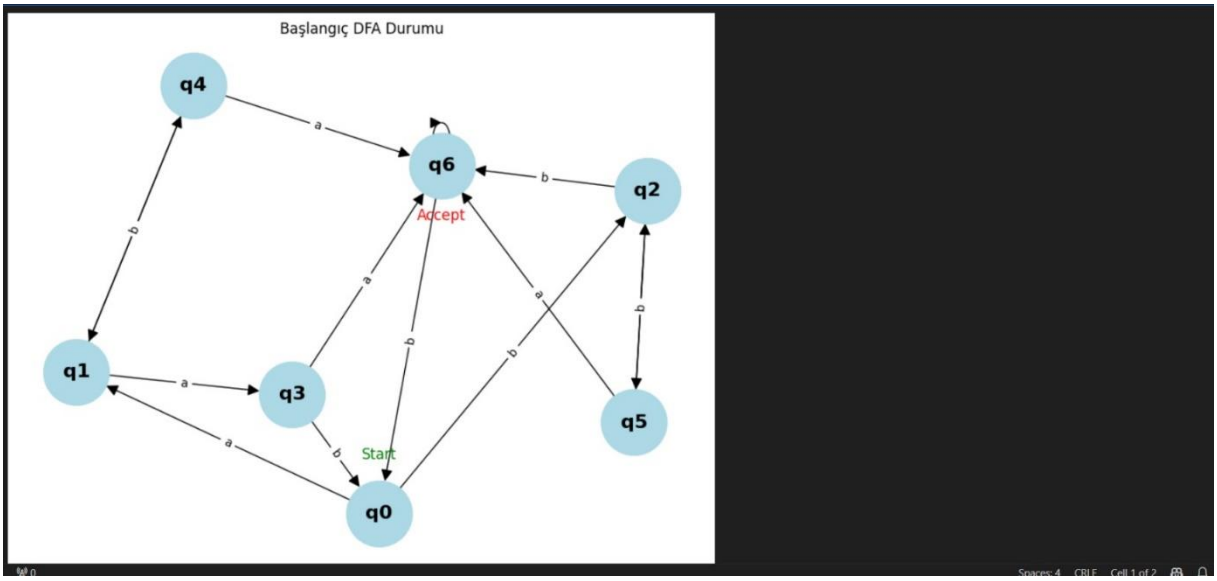
1. `networkx` kullanılarak durumlar ve geçişler bir grafik üzerinde çizilir.
2. Başlangıç durumu yeşil bir etiketle işaretlenir.
3. Kabul durumları kırmızı bir etiketle işaretlenir.

Durumlar, Alfabe ve Geçiş Fonksiyonu Tanımlaması

```
# Anlaşılması için düzgünce hazırlanmış DFA Tanımlaması
states = {'q0', 'q1', 'q2', 'q3', 'q4', 'q5', 'q6'}
alphabet = {'a', 'b'}
transition_function = {
    ('q0', 'a'): 'q1', ('q0', 'b'): 'q2',
    ('q1', 'a'): 'q3', ('q1', 'b'): 'q4',
    ('q2', 'a'): 'q5', ('q2', 'b'): 'q6',
    ('q3', 'a'): 'q6', ('q3', 'b'): 'q0',
    ('q4', 'a'): 'q6', ('q4', 'b'): 'q1',
    ('q5', 'a'): 'q6', ('q5', 'b'): 'q2',
    ('q6', 'a'): 'q6', ('q6', 'b'): 'q0',
}
start_state = 'q0'
accept_states = {'q6'}  # Kabul durumu q6

# DFA nesnesi oluşturuluyor
dfa = DFA(states, alphabet, transition_function, start_state, accept_states)

# Başlangıç DFA'sının çizimi
dfa.plot_dfa()

# Durum indirgeme işlemi
dfa.minimize()
```

- **`states`**: DFA'nın sahip olduğu durumlar kümesi belirtilmiştir.
    - Örneğin: `'q0', 'q1', ...` durumlar DFA'nın düğümleridir.
- **`alphabet`**: DFA'nın giriş alfabesi `{'a', 'b'}` olarak tanımlanmıştır.
- **`transition_function`**: DFA'nın geçiş fonksiyonu.
    - Örneğin: `'q0'` durumunda `'a'` sembolü alınırsa `'q1'` durumuna geçilir.

Ekran Çıktıları



```
Ulaşılamayan Durumlar: {'q0'}
İndirgenmiş DFA Grupları:
Grup: {'q1', 'q2', 'q3'}, Geçiş Fonksiyonu: ('q1', 'q2', 'q3')
Grup: {'q4', 'q5'}, Geçiş Fonksiyonu: ('q4', 'q5')
Grup: {'q6'}, Geçiş Fonksiyonu: ('q6', 'a')

Geçiş Fonksiyonu:
{'q1', 'q2', 'q3'} -> a -> {'q6'}
{'q1', 'q2', 'q3'} -> b -> {'q1', 'q2', 'q3'}
{'q4', 'q5'} -> a -> {'q4', 'q5'}
{'q4', 'q5'} -> b -> {'q6'}
{'q6'} -> a -> {'q6'}
{'q6'} -> b -> {'q1', 'q2', 'q3'}
```