ASSIGNMENT 1 : AUTOMATA THEORY

TASK

Write a python script to convert a Non-Deterministic finite automaton(NFA) to a deterministic finite automaton(DFA).

FUNCTIONS USED

def generatePowerSet(given_set, no_of_elements)

This function generates the powerset of a given_set (of type list) of size no_of_elements(of type integer). This function is used to generate all the possible states of the DFA and returns the powerset(of type list) of the set of states of the NFA.

def final_set_generator(set1, set2)

This function generates the set of all final(accepted) states of the DFA. It takes in two parameters: the set of all the possible states of the DFA i.e. the power set of set of all the states of the NFA (of type list) and the set of final states of the NFA(of type list) and returns the set of final states of the DFA (of type list) in accordance to the formula:

$$F_D = \{q \in Q_D : F_N \cap q \neq \phi\}.$$

def op_for(initial_state, t_func, input)

This function generates the next state in the DFA for the given initial state(of type list) and input (of type character) in accordance to the t_func of the NFA (of type list). It returns the union of the output of each element in initial_state for given input as the next state(of type list) in accordance to the formula:

$$\delta_D(R) = \bigcup \{r' | r' = \delta_N(r), \quad r', r \epsilon Q_N\}, \ R \epsilon Q_D$$

def generate_t_func_dfa(t_func_nfa, alphabet, all_states_dfa)

This function generates the t_func of the DFA, given the input alphabet (of type list of characters), the t_func of the nfa (of type list) and the set of all the possible states of the DFA (of type list). It generates the t_func in the following manner: It first considers each state in all_states_dfa. For each state and input character in the alphabet, it finds the next state using the function op_for (as defined above) and appends the transition (which is a list consisting of the state, input character and the next state) to the t_func of the DFA. This function returns a list.

EXPLANATION OF THE MAIN FUNCTION

1. Load the NFA as a dictionary from the input.json file

```
nfa = dict()
with open('input.json', 'r') as f:  # load the NFA from the file
    nfa = json.load(f)
```

2. Initialize a DFA as a dictionary object and assign the parameters

```
dfa = dict()
dfa["states"] = pow(2, nfa["states"])
dfa["letters"] = nfa["letters"]
all_states_dfa = generatePowerSet(list(range(0, nfa["states"])), nfa['states'])
dfa["t_func"] = generate_t_func_dfa(nfa["t_func"], nfa["letters"], all_states_dfa)
dfa["start"] = [nfa["start"]]
dfa["final"] = final_set_generator(all_states_dfa, nfa["final"])
```

Initialize DFA as an empty dictionary. And then assign its parameters:

1. Number of states in a DFA is 2^number of states of an NFA
2. Both NFA and DFA follow the same alphabet and hence have the same set of letters.
3. The states of the DFA is given by the power set of all the states of the NFA.
4. The t_func of the DFA is generated using the generate_t_func_dfa function as described above.
5. Both DFA and NFA have the same start state.
6. The set of all final/accept states of the DFA is generated using the final_set_generator function as described above.

3. Dump the DFA into the output.json file

```
with open('output.json', 'w') as outfile:
    json.dump(dfa, outfile, indent=2)
```

SAMPLE INPUT AND OUTPUT

input.json

```
{
    "states": 2,
    "letters": ["a", "b"],
    "t_func": [
        [0, "a", [0, 1]],
        [0, "b", [1]],
        [1, "b", [0, 1]]
    ],
    "start": 0,
    "final": [1]
}
```

output.json

```
{
    "states": 4,
    "letters": ["a", "b"],
    "t_func": [
        [[], "a", []],
        [[], "b", []],
        [[0], "a", [0, 1]],
        [[0], "b", [1]],
        [[1], "a", []],
        [[1], "b", [0, 1]],
        [[0, 1], "a", [0, 1]],
        [[0, 1], "b", [0, 1]]
    ],
    "start": [0],
    "final": [[1], [0, 1]]
}
```

ASSUMPTIONS MADE

1. There are no NULL transitions
2. All the transitions, including those that have no next state, is included in the t_func of the DFA