МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение
высшего образования
«МИРЭА – Российский технологический университет»
РТУ МИРЭА

Институт Информационных Технологий
Кафедра Вычислительной техники

ОТЧЕТ О ВЫПОЛНЕНИИ ПРАКТИЧЕСКОЙ РАБОТЫ №5

«Перевод из недетерминированного конечного автомата в детерминированный конечный автомат»

по дисциплине
«Теория формальных языков»

Выполнил студент группы ИКБО-42-23 *Туляшева А.Т.*

Принял старший преподаватель *Боронников А.С.*

Практическая работа выполнена «__»_____2024 г.

«Зачтено» «__»_____2024 г.

Москва 2024

Перевод из НКА в ДКА

Условие задачи: на выбранном языке программирования реализовать перевод из недетерминированного конечного автомат в детерминированный конечный автомат.

Код на языке Python:

```
from collections import defaultdict
def get_next_states(states, symbol, moves):
    result = set() #хранятся все состояния
    for state in states: result.update(moves.get((state, symbol), []))
    return result #множество состояний
def nfa_to_dfa(states, alph, moves, initial_states, final_states): #НКА в ДКА
    dfa_transitions = {} #переходы ДКА
    dfa_states = [] #список всех состояний ДКА
    start_states = [frozenset(initial_states)]  #начальное состояние (1)
    dfa_initial_state = start_states[0] #(1)
    while start_states:
        current = start_states.pop() #текущее состояние
        dfa_states.append(current)
        for symbol in alph:
            next_states = get_next_states(current, symbol, moves)
            next_states = frozenset(next_states)
            if next_states and next_states not in dfa_states and next_states
not in start_states:
                start_states.append(next_states)
            if next_states: #добавляем переход в список переходов
                dfa_transitions[(current, symbol)] = next_states
    dfa_final_states = [s for s in dfa_states if s.intersection(final_states)]
    return dfa_states, dfa_transitions, dfa_initial_state, dfa_final_states
states = input("Enter set of states:\n").split() #набор состояний
alph = input("Enter the input alphabet:\n").split() #алфавит
moves = defaultdict(list) #словарь переходов
print("Enter state-transitions function (current state, input character, next
state):")
while True: #ввод до пустой строки
    line = input()
    if not line: break #если ничего не вводится
    for transition in line.split(): #обрабатываем переходы
        current_state,           input_char,           next_state           =
transition.strip('()').split(',')
        #из текущего состояния по символу можно перейти в следующее состояние
        moves[(current_state, input_char)].append(next_state)
initial_states  =  set(input("Enter  a  set  of  initial  states:\n").split())
#начальное состояние
final_states = set(input("Enter a set of final states:\n").split()) #конечное
#преобразование НКА в ДКА
dfa_states, dfa_transitions, dfa_initial_state, dfa_final_states = nfa_to_dfa(
    states, alph, moves, initial_states, final_states)

#вывод
print("DFA:")
print("Set of states:", ", ".join(''.join(s) for s in dfa_states))
print("Input alphabet:", ", ".join(alph))
print("State-transitions function:")
for (state, symbol), next_state in dfa_transitions.items():
    print(f"D({''.join(state)}, {symbol}) = {''.join(next_state)}")
print("Initial states:", ''.join(dfa_initial_state))
print("Final states:", ', '.join(''.join(s) for s in dfa_final_states))
```

Тестирование показало, что программа работает успешно (рис. 1).

```
Enter set of states:
1 2 3
Enter the input alphabet:
a b
Enter state-transitions function (current state, input character, next state):
(1,a,1) (1,a,2) (1,b,3) (2,a,2) (2,b,1) (2,b,3) (3,a,3) (3,b,3)

Enter a set of initial states:
1
Enter a set of final states:
3
DFA:
Set of states: 1, 3, 21, 13, 213
Input alphabet: a, b
State-transitions function:
D(1, a) = 21
D(1, b) = 3
D(3, a) = 3
D(3, b) = 3
D(21, a) = 21
D(21, b) = 13
D(13, a) = 213
D(13, b) = 3
D(213, a) = 213
D(213, b) = 13
Initial states: 1
Final states: 3, 13, 213
```

Рисунок 1 – Тестирование