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 №4

«Преобразования недетерминированного конечного автомата (НКА) в детерминированный (ДКА) **на Python»**

по дисциплине
«Теория формальных языков**»**

Выполнил студент группы ИКБО-15-22 *Оганнисян Г.А.*

Принял старший преподаватель *Боронников А.С.*

Практическая работа выполнена «__»_____2023 г.

«Зачтено» «__»_____2023 г.

Москва 2021

```
D:\Work\Practic_MIREA\TFYA>python pr_4.py
Введите множество состояний (разделяйте пробелами):
```

Рисунок 1 – Программа просит ввести данные

```
Введите множество состояний (разделяйте пробелами): 1 2 3
Введите алфавит ввода (разделяйте пробелами): a b
Введите функцию переходов (текущее состояние, входной символ, следующее состояние): (1,a,1) (1,a,2) (1,b,3) (2,a,2)
 (2,b,1) (2,b,3) (3,a,3) (3,b,3)
Введите множество начальных состояний (разделяйте пробелами): 1
Введите множество конечных состояний (разделяйте пробелами): 3
```

Рисунок 2 - Ввод всех нужных для расчета данных

```
DFA:
Множество состояний: D(1), D(), D(1), D(2), D(1, 2), D(3), D(1, 3), D(2, 3), D(1, 2, 3)
Алфавит ввода: a, b
Функция переходов:
Начальные состояния: D(1)
Конечные состояния: D(3), D(1, 3), D(2, 3), D(1, 2, 3)
```

Рисунок 3 – Вывод программы

Эта программа решает задачу преобразования конечного автомата с недетерминированными переходами (NFA) в детерминированный автомат (DFA). Сначала она собирает информацию о состояниях, алфавите, переходах, начальных и конечных состояниях NFA из пользовательского ввода. Затем она использует эту информацию для построения эквивалентного DFA.

Программа использует алгоритм под названием "подмножества состояний". Она начинает с начальных состояний NFA и итеративно строит новые состояния DFA, рассматривая все возможные переходы по символам

алфавита. Это позволяет ей пошагово создавать множество состояний и определять, являются ли они конечными.

В конце программа выводит информацию о построенном DFA, включая множество состояний, алфавит, функцию переходов, начальные и конечные состояния.

Листинг кода

```
def powerset(s):
    if len(s) == 0:
        return [[]]
    subsets = powerset(s[:-1])
    return subsets + [item + [s[-1]] for item in subsets]

def convert_nfa_to_dfa(states, alphabet, transitions, initial_states, final_states):
    dfa_states = []
    dfa_transitions = {}
    dfa_initial_state = ''
    dfa_final_states = []

    # Create the initial state for the DFA
    dfa_initial_state = 'D(' + ', '.join(initial_states) + ')'
    dfa_states.append(dfa_initial_state)

    powerset_states = powerset(states)

    for subset in powerset_states:
        subset_name = 'D(' + ', '.join(subset) + ')'
        dfa_states.append(subset_name)

        for symbol in alphabet:
            next_state = set()
            for state in subset:
                for transition in transitions:
                    if transition[0] == state and transition[1] == symbol:
                        next_state.add(transition[2])
            if next_state:
                next_state_name = 'D(' + ', '.join(sorted(next_state)) + ')'
                dfa_transitions[(subset_name, symbol)] = next_state_name

                if next_state_name not in dfa_states:
                    dfa_states.append(next_state_name)

    for subset in powerset_states:
        for final_state in final_states:
            if final_state in subset:
```

```
                dfa_final_states.append('D(' + ', '.join(subset) + ')')
                break

    return dfa_states, alphabet, dfa_transitions, dfa_initial_state, dfa_final_states

# Ввод данных через консоль
states = input("Введите множество состояний (разделяйте пробелами): ").split()
alphabet = input("Введите алфавит ввода (разделяйте пробелами): ").split()
transitions_input = input("Введите функцию переходов (текущее состояние, входной символ, следующее
состояние): ").split()
transitions = [tuple(transitions_input[i:i+3]) for i in range(0, len(transitions_input), 3)]
initial_states = input("Введите множество начальных состояний (разделяйте пробелами): ").split()
final_states = input("Введите множество конечных состояний (разделяйте пробелами): ").split()

# Вызов функции для преобразования
dfa_states, dfa_alphabet, dfa_transitions, dfa_initial_state, dfa_final_states = convert_nfa_to_dfa(
    states, alphabet, transitions, initial_states, final_states)

# Вывод результата
print("\nDFA:")
print("Множество состояний:", ', '.join(dfa_states))
print("Алфавит ввода:", ', '.join(dfa_alphabet))
print("Функция переходов:")
for key, value in dfa_transitions.items():
    print(key[0], "=", key[1], "=", value)
print("Начальные состояния:", dfa_initial_state)
print("Конечные состояния:", ', '.join(dfa_final_states))
```