

Аметов К.Л. ПИ-231(2)

Задание: Минимизировать ДКА Мили, заданный таблицей выходов и переходов.

Построить эквивалентный автомат 2 рода. Провести проверку в JFLAP

№ 1

A \ Q	1	2	3	4	5	6	7	8	9
a	4,x	3,y	7,x	5,x	8,x	7,x	2,y	5,x	2,y
b	7,y	4,y	9,x	9,y	7,y	9,y	8,x	9,y	5,x

Построим множества π_1 эквивалентных состояний: это состояния, не различимые никаким однобуквенным воздействием: $\pi_1 = \{A, B, C, D\}$, $Ax = \{1, 4, 5, 6, 8\}$, $Bx = \{9, 7\}$, $Cx = \{3\}$, $Dy = \{2\}$.

Индексами для удобства обозначены реакции на воздействия a, b соответственно.

Класс Ax был расщеплен, процедура была завершена за два шага.

В самом низу картинки представлен эквивалентный гомоморфный нормализованный автомат мили с прежними обозначениями.

A \ Q	1	2	3	4	5	6	7	8	9
a	4x	3y	7x	5x	8x	7x	2y	5x	2y
b	7y	4y	9x	9y	7y	9y	8x	9y	5x

A \ Q	1	4	5	6	8	9	7	3	2
a	A	A	A	B	A	D	D	B	C
b	B	B	B	B	B	A	A	B	A

A \ Q	1	4	5	8	9	7	3	2	6
a	B0	B0	B0	B0	B3	B3	B1	B2	B1
b	B1	B1	B1	B1	B0	B0	B1	B0	B1

A \ Q	1	9	3	2	6
a	B0	B3	B1	B2	B1
b	B1	B0	B1	B0	B1

A \ Q	1	9	3	2	6
a	1/x	2/y	9x	3y	9x
b	9/y	1/x	9x	1y	9y

Убедимся на частном примере, что автоматные преобразования двух машин одинаковы.

Выберем в исходном автомате в качестве начального состояния 4 и подадим на его вход цепочку "abba". Так же проверим нормализованный автомат.

Исходная машина				
Момент времени	t	T+1	T+2	T+3
Анализируемый символ	a	b	b	a
Реакция	x	y	x	x
состояние к след моменту	4	9	5	8

Нормализованная машина				
Момент времени	t	T+1	T+2	T+3
Анализируемый символ	a	b	b	a
Реакция	x	y	x	x
состояние к след моменту	1	9	1	1

Перейдем к автомату 2 рода (Мура), построю его и покажу в табличном виде

	A	B	C	D	E	F	G	H
A/Q	1	9	3	2	6			
a	1/x	2/y	9x	3y	9x			
b	9/y	1/x	9x	1y	9y			
Состояния	Реак при дост	Реак						
1	1/x 1/y	A0/x A1/y						
9	9/y 9x	A2/y A3/x						
3	3/y	A4/y						
2	2/y	A5/y						
6	-	A6/-						
A/Q	A0	A1	A2	A3	A4	A5	A6	
a	A0	A0	A5	A5	A3	A4	A3	
b	A2	A2	A0	A0	A3	A1	A2	

Проверю автомат мура

Момент времени	t	T+1	T+2	T+3	T+4
Анализируемый символ	a	b	b	a	-
Реакция	x	x	y	x	x
состояние к след моменту	A0	A2	A0	A0	-
состояние в тек момент	A0	A0	A2	A0	A0

P.S

Так же была выполнена программная реализация автомата Мура и Мили, построение эквивалентного автомату Мили автомата мура и минимизация автомата на ЯП Python.

Ниже показанны скриншоты реализации и результаты прогонов строки "abba".

```

1 task > base_automata.py > BaseAutomata > _init_
2 """Modules"""
3
4 class BaseAutomata:
5     """Class for Automata"""
6     def __init__(self, states: list[str],
7                 initial_state: str,
8                 alphabet: list[str]) -> None:
9         if initial_state not in states:
10             raise ValueError("Initial state must be in states")
11         self.states = states
12         self.state = initial_state
13         self.alphabet = alphabet
14
15 1 task > moor_automata.py > ...
16 """Import base modules"""
17 from base_automata import BaseAutomata
18
19 class MoorAutomata(BaseAutomata):
20     """Realisation of Moor Automata"""
21     def __init__(self, states: list[str],
22                 initial_state: str,
23                 alphabet: list[str],
24                 table: dict[str, dict[str, str]],
25                 table_reactions: dict[str, str]) -> None:
26         super().__init__(states, initial_state, alphabet)
27         self.table = table
28         self.table_reactions = table_reactions
29
30     def step(self, inp: str) -> None:
31         """Machine step"""
32         self.state = self.table[self.state][inp]
33
34     def get_reaction(self):
35         """Return reaction"""
36         try:
37             return self.table_reactions[self.state]
38         except:
39             return ""
40
41 1 task > mealy_automata.py > ...
42 """Import base modules"""
43 from base_automata import BaseAutomata
44
45 class MealyAutomata(BaseAutomata):
46     """Realisation of Mealy Automata"""
47     def __init__(self, states: list[str],
48                 initial_state: str,
49                 alphabet: list[str],
50                 table: dict[str, dict[str, list[str]]]) -> None:
51         super().__init__(states, initial_state, alphabet)
52         self.table = table
53
54     def step(self, inp: str):
55         """Machine step"""
56         self.state, reaction = self.table[self.state][inp]
57         return reaction
  
```

Реализация построения эквивалентного автомату Мили, автомата Мура

```

1_task > transformation.py > mealy_to_moor
1  """Modules"""
2  from collections import defaultdict
3  from moor_automata import MoorAutomata
4  from mealy_automata import MealyAutomata
5
6  def mealy_to_moor(automata: MealyAutomata) -> MoorAutomata:
7      """Transforms a Mealy automaton into a Moore automaton."""
8      state_reactions = defaultdict(set)
9      for state in automata.states:
10         for symbol in automata.alphabet:
11             next_state, reaction = automata.table[state][symbol]
12             state_reactions[next_state].add(reaction)
13
14     new_states = {}
15     reaction_table = {}
16     old_to_new_state = {}
17     state_counter = 0
18
19     for state in automata.states:
20         if not state_reactions[state]:
21             state_reactions[state].add("") # Для состояния без реакций
22
23         for reaction in state_reactions[state]:
24             new_state = f"A{state_counter}"
25             new_states[(state, reaction)] = new_state
26             reaction_table[new_state] = reaction
27             old_to_new_state[state] = new_state
28             state_counter += 1
29
30     transition_table = defaultdict(dict)
31     for new_state, old_state in new_states.items():
32         # print(1, new_state, old_state)
33         for symbol in automata.alphabet:
34             next_old_state, reaction = automata.table[old_state][symbol]
35             transition_table[new_state][symbol] = new_states[(next_old_state, reaction)]
36
37     return MoorAutomata(
38         states=transition_table.keys(),
39         initial_state=old_to_new_state[automata.state],
40         alphabet=automata.alphabet,
41         table=transition_table,
42         table_reactions=reaction_table
43     )
44

```

Реализация минимизации

```
from mealy_automata import MealyAutomata
from collections import defaultdict
from moor_automata import MoorAutomata
from transformation import mealy_to_moor

def minimization(automata: MealyAutomata):
    reactions = {}
    ind = 1
    table = automata.table
    old_states_to_new = {}
    for s, i in table.items():
        reaction = tuple(i[a][1] for a in automata.alphabet)
        if reaction not in reactions:
            reactions[reaction] = f"A{ind}"
            ind+=1
    for s, i in table.items():
        reaction = tuple(i[a][1] for a in automata.alphabet)
        old_states_to_new[s] = reactions[reaction]

    table1= defaultdict(dict)
    for s, i in table.items():
        for a in automata.alphabet:
            table1[s][a] = old_states_to_new[table[s][a][0]]

    table2 = defaultdict(dict)
    ind2 = 66
    while True:
        old_states_to_new1 = {}
        reactions1 ={}
        ind = 1
        for s, i in table1.items():
            reaction1 = tuple(j for _,j in i.items())
            if reaction1 not in reactions1:
                reactions1[reaction1] = f"{chr(ind2)}{ind}"
                ind+=1
        for s, i in table1.items():
            reaction1 = tuple(j for _,j in i.items())
            old_states_to_new1[s] = reactions1[reaction1]
        for s, i in table.items():
            for a in alphabet:
                table1[s][a] = old_states_to_new1[table[s][a][0]]
        ind2+=1
        if table1 == table2:
            break
        table2 = table1
    new_to_old = {j:i for i,j in old_states_to_new1.items()}
    table2= defaultdict(dict)
    for s, i in table.items():
        for a in alphabet:
            table2[old_states_to_new1[s]][a] = (table1[s][a], table[s][a][1])
    table3 =defaultdict(dict)
    for s,i in table2.items():
        for a in alphabet:
            table3[new_to_old[s]][a] = (new_to_old[table2[s][a][0]], table2[s][a][1])
    return table3
```

Прогон строки “abba” на исходном, нормализованном и эквивалентном автомате второго рода.

```
69 # Таблица переходов и реакций для автомата Мили
70 table = {
71     "1": {"a": ["4", "x"], "b": ["7", "y"]},
72     "2": {"a": ["3", "y"], "b": ["4", "y"]},
73     "3": {"a": ["7", "x"], "b": ["9", "x"]},
74     "4": {"a": ["5", "x"], "b": ["9", "y"]},
75     "5": {"a": ["8", "x"], "b": ["7", "y"]},
76     "6": {"a": ["7", "x"], "b": ["9", "y"]},
77     "7": {"a": ["2", "y"], "b": ["8", "x"]},
78     "8": {"a": ["5", "x"], "b": ["9", "y"]},
79     "9": {"a": ["2", "y"], "b": ["5", "x"]}
80 }
81
82 #
83 # СПИСОК СОСТОЯНИЙ
84 states = list(table.keys())
85
86 # Начальное состояние
87 initial_state = "8"
88
89 # Алфавит входных символов
90 alphabet = ["a", "b"]
91
92 automata = MealyAutomata(states, initial_state, alphabet, table)
93 print("Результат работы над строкой 'abba' автомата Мили "+"".join([automata.step(i) for i in "abba"]))
94
95 automata.table = minimization(automata)
96 automata.states = automata.table.keys()
97 automata.state = "8"
98 print("Результат работы над строкой 'abba' нормализованного автомата Мили "+"".join([automata.step(i) for i in "abba"]))
99
100 res = ""
101 moor = mealy_to_moor(automata)
102 for i in "abba":
103     moor.step(i)
104     res += moor.get_reaction()
105 print(f"Результат работы над строкой 'abba' автомата Мура {res}")
```

```
(.venv) kemran@kemran:~/Automata_Theory_Solves$ cd /home/kemran/Automata_Theory_Solves/1
ter/../../debugpy/launcher 39693 -- /home/kemran/Automata_Theory_Solves/1
Результат работы над строкой 'abba' автомата Мили хухх
Результат работы над строкой 'abba' нормализованного автомата Мили хухх
Результат работы над строкой 'abba' автомата Мура хухх
(.venv) kemran@kemran:~/Automata_Theory_Solves$
```

Вывод: выполнена минимизация автомата Мили, построен эквивалентный нормализованный автомат Мура; автоматные преобразования для тестовой четырех-символьной цепочки совпадают.