МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ

(НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)

Кафедра теории вероятностей и компьютерного моделирования

**Лабораторная работа № 3**

**По спецкурсу «Теория сложности алгоритмов»**

**Разрешимость по Тьюрингу**

Выполнил: Покхарел П.К.

Группа: М8О-101М-22, Вариант 8

Преподаватель: Рассказова В.А.

Москва, 2023

**Задание.** Для заданного языка, в котором предполагается, что КС-грамматика, регулярное выражение или ДКА определены над алфавитом {0, 1},

1. построить описание МТ, решающей его;

2. реализовать данную МТ в виде программы.

INFINITEDFA = {⟨A⟩: A – ДКА и L(A) – бесконечный язык}

**Описание МТ:**

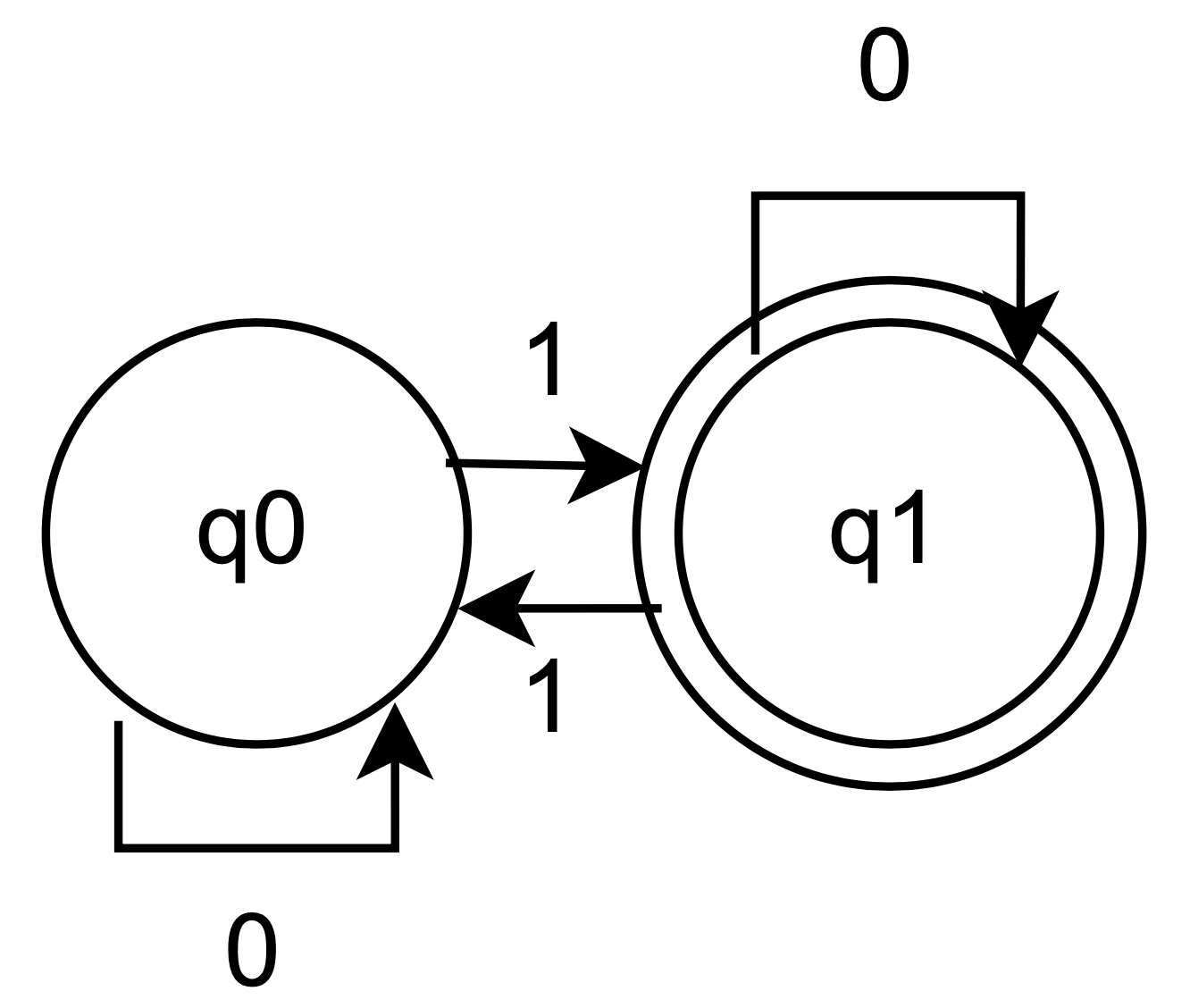
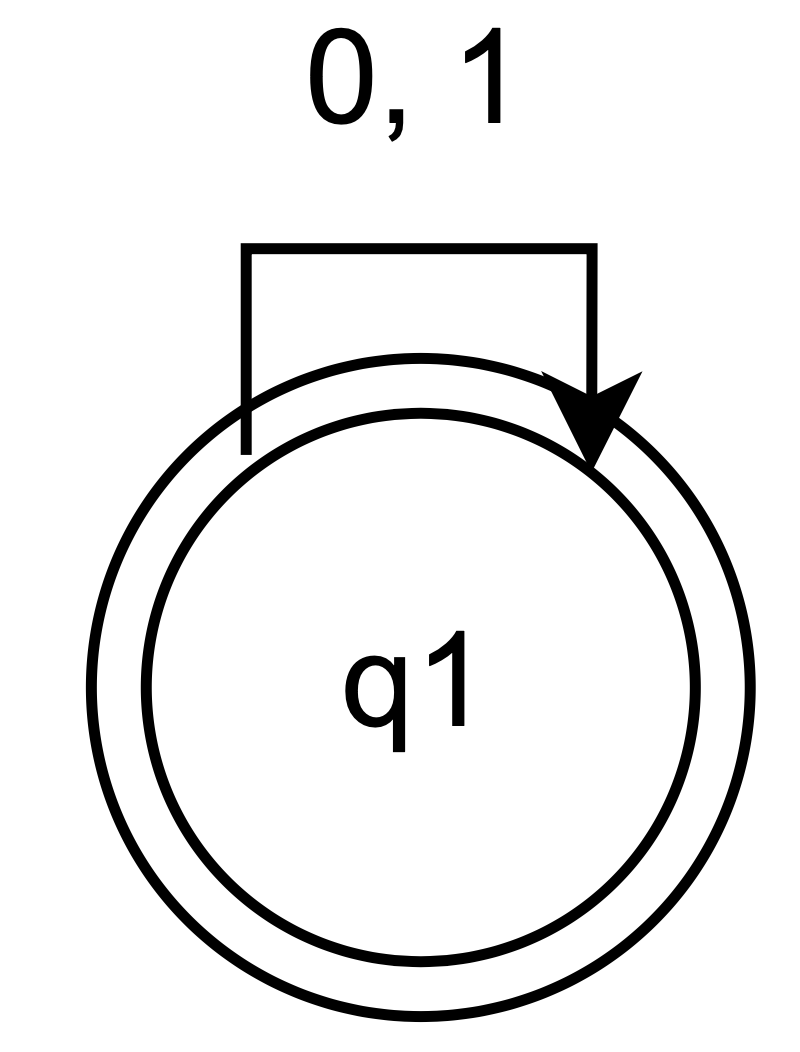
Алгоритм состоит в том, чтобы пройти все правила и проверить, что допустимое правило было посещено в ходе проверки как минимум 2 раза при считывании одного из символов 0 и 1.  
 В цикле будем проверять каждое правило, пока оно не будет проверено 2 раза, а также пока каждое состояние, в которое мы переходим по данному правилу, не будет проверено 2 раза  
 Если эти два условия выполнены, то следующие состояния и правила перехода из них не проверяем кроме случаев, когда эти состояния допустимые.

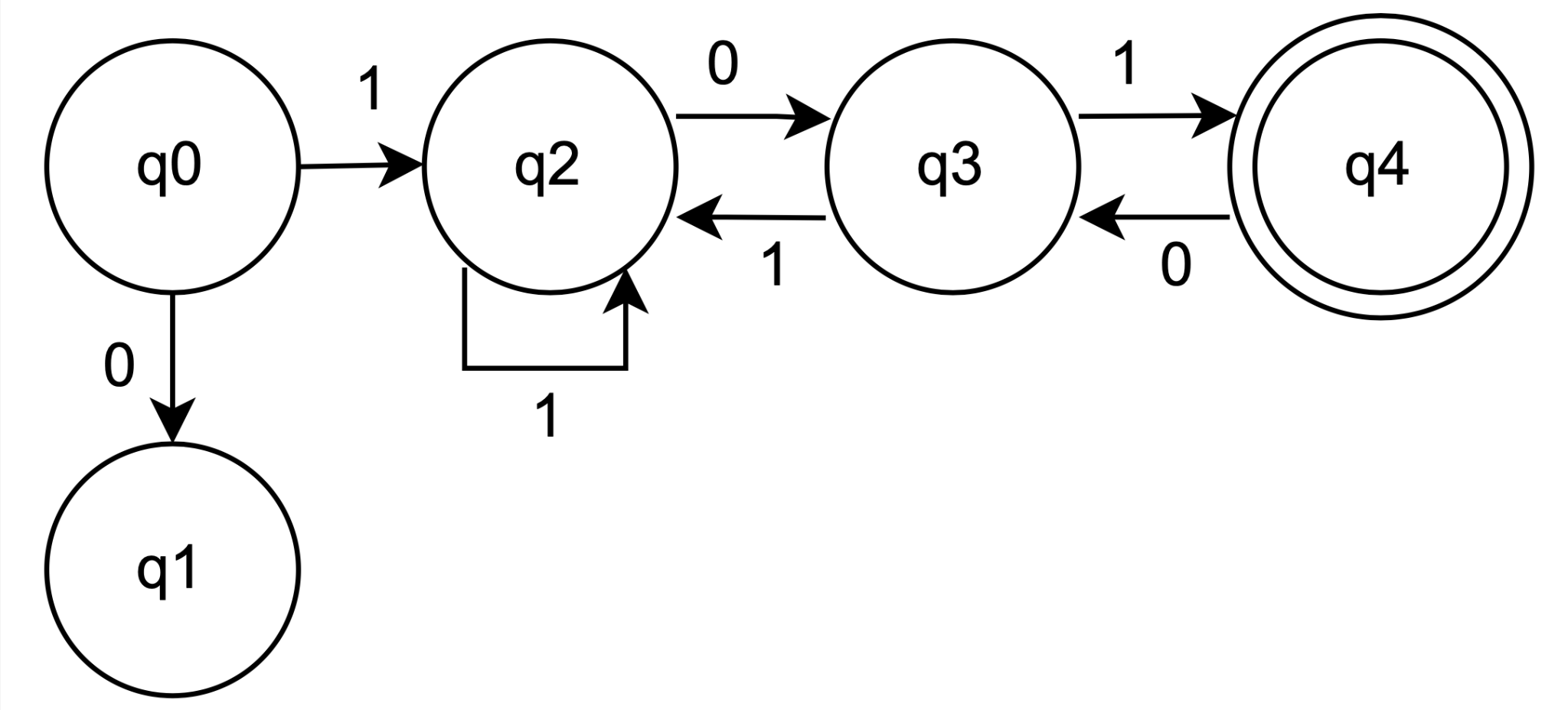
Вход: ДКА ⟨A⟩

Выход: 'ДКА (НЕ) ПОРОЖДАЕТ бесконечный язык'

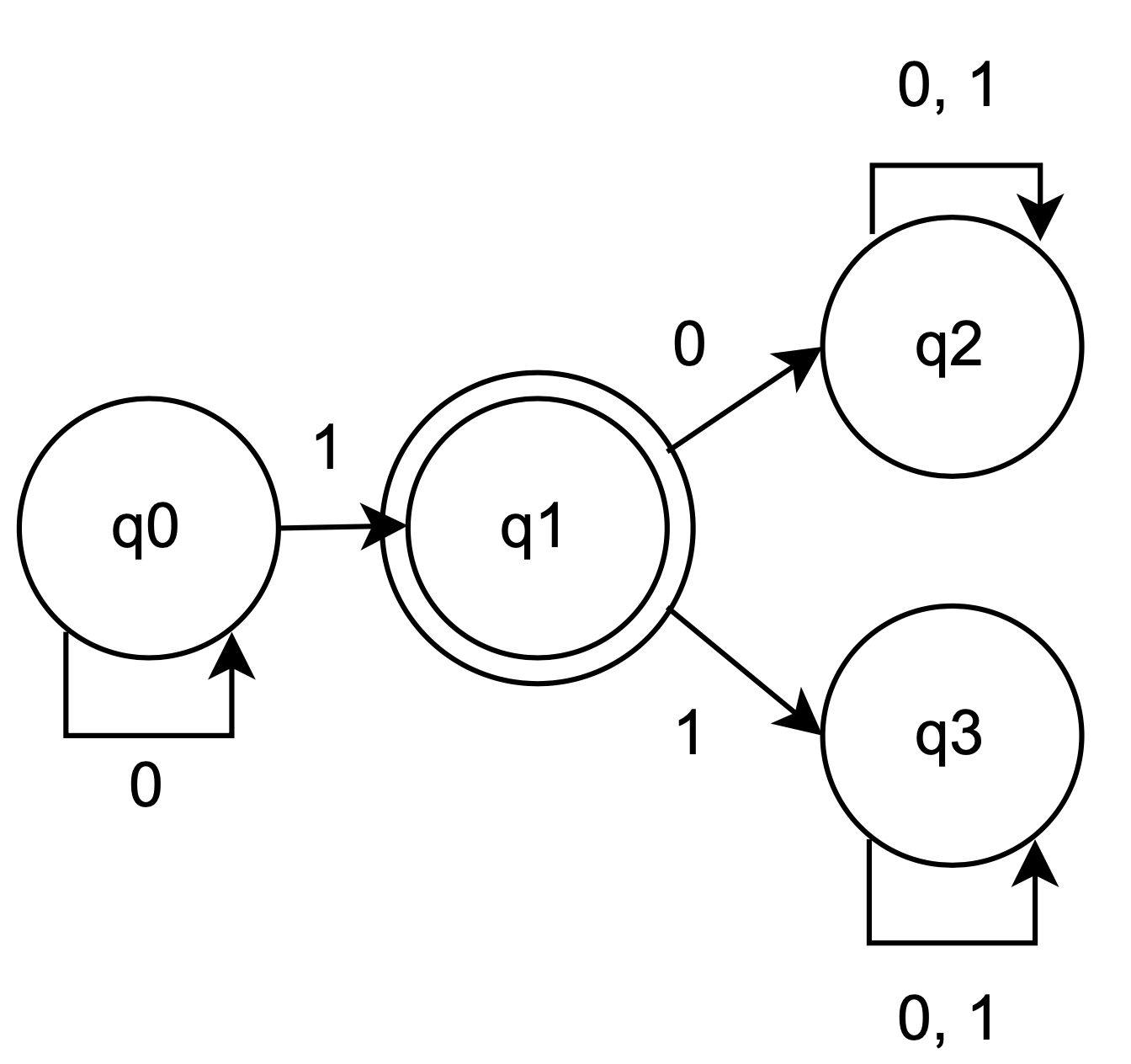
Примеры ДКА, которые распознаются нашей МТ:

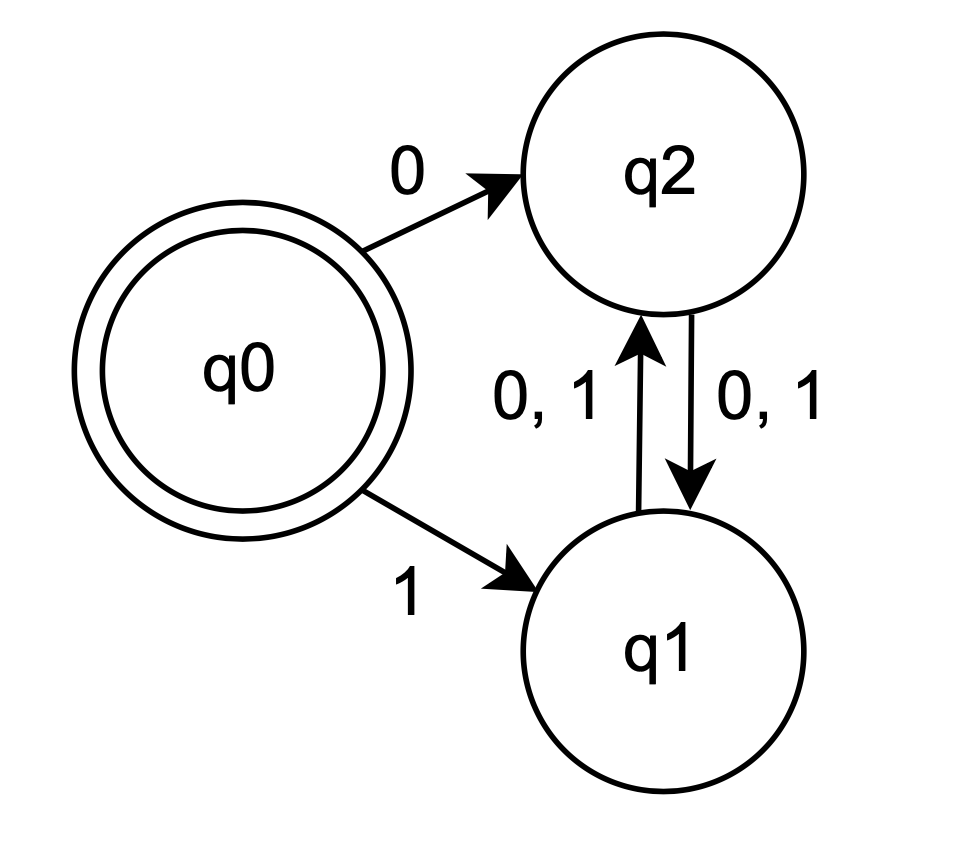
1. порождающие бесконечный язык

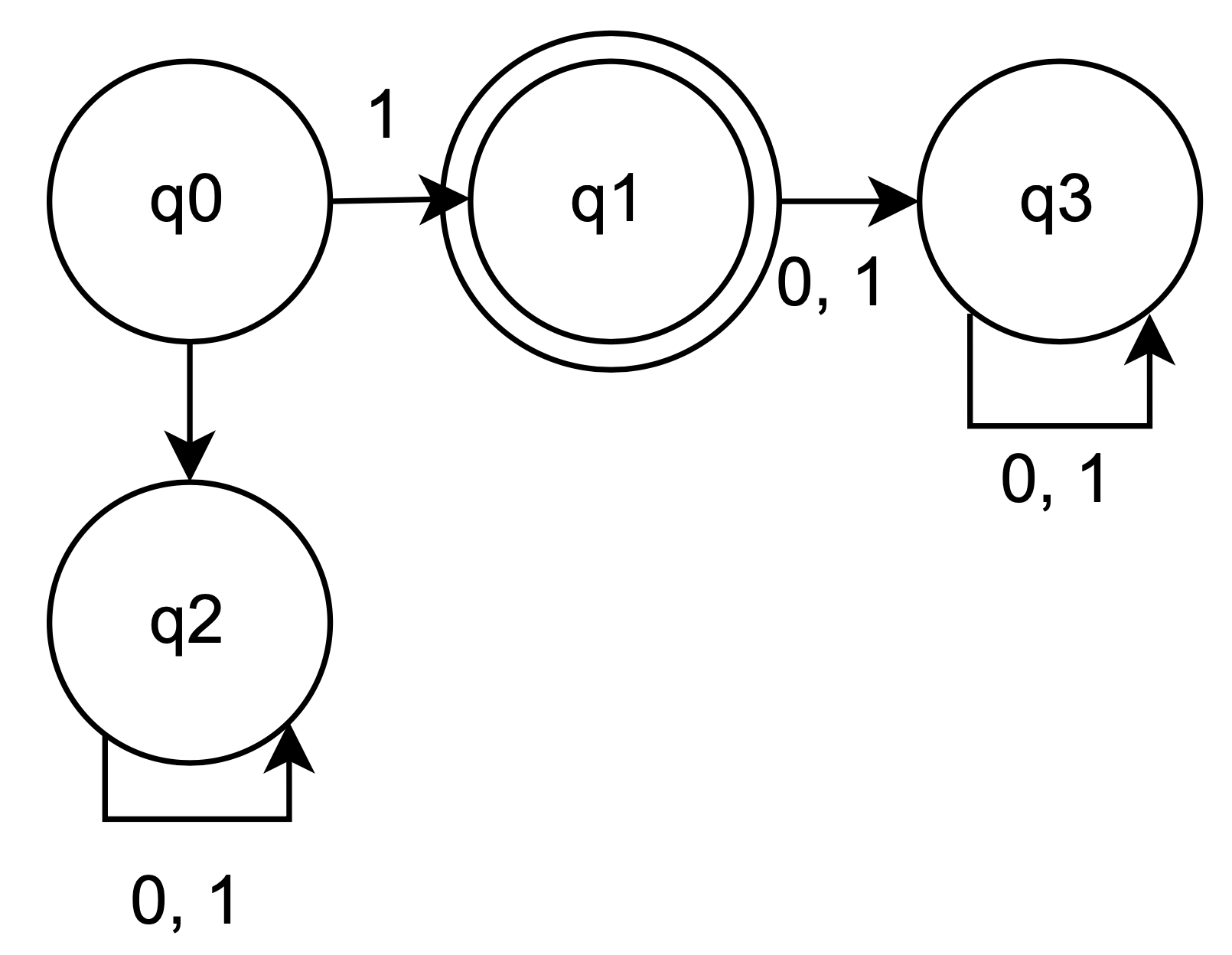


****

2) не порождающие бесконечный язык







**Программная реализация**

import numpy as np

import copy

def check\_dfa(q\_states, q\_start, rules, q\_allow):

# Создадим словарь, где будем отмечать, сколько раз было проверено правило

rules\_checked = dict(zip(rules, [0]\*len(rules)))

rules\_len = len(rules)

q\_states\_len = len(q\_states)

# Создадим словарь, где будем отмечать, сколько раз было посещено состояние

dict\_zeros\_ones = {'0': 0, '1': 0}

dict\_zeros\_ones\_list = [copy.deepcopy(dict\_zeros\_ones) for \_ in range(q\_states\_len)]

q\_states\_visited = dict(zip(q\_states, dict\_zeros\_ones\_list))

# print(q\_states\_visited)

i = 0

# print(q\_states\_visited)

q\_list = [q\_start]

# Зададим число итераций из расчета, что каждое правило должно быть

# проверено не больше двух раз

i\_crit = rules\_len\*3

while i < i\_crit:

if i >= len(q\_list):

break

# print(i, len(q\_list))

q\_curr = q\_list[i]

# print(q\_curr)

curr\_rules = []

for rule in rules:

if rule[0] == q\_curr: # and rules\_checked[rule] < 2:

# print(rule)

curr\_rules.append(rule)

q\_new\_list = []

for curr\_rule in curr\_rules:

q\_next = curr\_rule[2]

q\_tr = curr\_rule[1]

if not (q\_states\_visited[q\_next][q\_tr] > 2 and q\_next not in q\_allow and rules\_checked[curr\_rule] > 2):

q\_new\_list.append(process\_rule(curr\_rule, q\_states\_visited))

rules\_checked[curr\_rule] += 1

# print(rules\_checked)

# print(q\_states\_visited)

# print(q\_new\_list)

q\_list += q\_new\_list

i += 1

for q in q\_allow:

if (q\_states\_visited[q]['0'] >= 2) or (q\_states\_visited[q]['1'] >= 2):

print('ДКА ПОРОЖДАЕТ бесконечный язык')

break

else:

print('ДКА НЕ ПОРОЖДАЕТ бесконечный язык')

# print(q\_list)

def process\_rule(rule, q\_states\_visited):

q\_curr = rule[0]

q\_new = rule[2]

q\_tr = rule[1]

q\_states\_visited[q\_new][q\_tr] += 1

return q\_new

"""Теперь проведем 6 тестов"""

# Тест 1

# ДКА, который ПОРОЖДАЕТ бесконечный язык

#

q\_states\_t1 = ['q0', 'q1', 'q2', 'q3', 'q4']

q\_start\_t1 = 'q0'

q\_allow\_t1 = ['q4']

rules\_t1 = [

('q0', '0', 'q1'),

('q0', '1', 'q2'),

('q1', '0', 'q1'),

('q1', '1', 'q1'),

('q2', '0', 'q2'),

('q2', '1', 'q3'),

('q3', '0', 'q2'),

('q3', '1', 'q4'),

('q4', '1', 'q4'),

('q4', '0', 'q3')

]

check\_dfa(q\_states\_t1, q\_start\_t1, rules\_t1, q\_allow\_t1)

# Тест 2

# ДКА, который ПОРОЖДАЕТ бесконечный язык

#

q\_states\_t2 = ['q0', 'q1', 'q2', 'q3']

q\_start\_t2 = 'q0'

q\_allow\_t2 = ['q1']

rules\_t2 = [

('q0', '0', 'q0'),

('q0', '1', 'q1'),

('q1', '0', 'q2'),

('q1', '1', 'q3'),

('q2', '0', 'q2'),

('q2', '1', 'q2'),

('q3', '0', 'q3'),

('q3', '1', 'q3'),

]

check\_dfa(q\_states\_t2, q\_start\_t2, rules\_t2, q\_allow\_t2)

# Тест 3

# ДКА, который ПОРОЖДАЕТ бесконечный язык

#

q\_states\_t3 = ['q0', 'q1']

q\_start\_t3 = 'q0'

q\_allow\_t3 = ['q1']

rules\_t3 = [

('q0', '0', 'q0'),

('q0', '1', 'q1'),

('q1', '0', 'q1'),

('q1', '1', 'q0'),

]

check\_dfa(q\_states\_t3, q\_start\_t3, rules\_t3, q\_allow\_t3)

# Тест 4

# ДКА, который НЕ ПОРОЖДАЕТ бесконечный язык

#

q\_states\_t4 = ['q0', 'q1', 'q2', 'q3']

q\_start\_t4 = 'q0'

q\_allow\_t4 = ['q1']

rules\_t4 = [

('q0', '0', 'q2'),

('q0', '1', 'q1'),

('q1', '0', 'q3'),

('q1', '1', 'q3'),

('q2', '0', 'q2'),

('q2', '1', 'q2'),

('q3', '0', 'q3'),

('q3', '1', 'q3'),

]

check\_dfa(q\_states\_t4, q\_start\_t4, rules\_t4, q\_allow\_t4)

# Тест 5

# ДКА, который ПОРОЖДАЕТ бесконечный язык

#

q\_states\_t5 = ['q0']

q\_start\_t5 = 'q0'

q\_allow\_t5 = ['q0']

rules\_t5 = [

('q0', '0', 'q0'),

('q0', '1', 'q0'),

]

check\_dfa(q\_states\_t5, q\_start\_t5, rules\_t5, q\_allow\_t5)

# Тест 6

# ДКА, который НЕ ПОРОЖДАЕТ бесконечный язык

#

q\_states\_t6 = ['q0', 'q1', 'q2']

q\_start\_t6 = 'q0'

q\_allow\_t6 = ['q0']

rules\_t6 = [

('q0', '0', 'q2'),

('q0', '1', 'q1'),

('q1', '0', 'q2'),

('q1', '1', 'q2'),

('q2', '0', 'q1'),

('q2', '1', 'q1')

]

check\_dfa(q\_states\_t6, q\_start\_t6, rules\_t6, q\_allow\_t6)