|  |  |  |
| --- | --- | --- |
|  | | |
| Федеральное государственное бюджетное  образовательное учреждение высшего образования «Новосибирский государственный технический университет» | | |
|  | | |
| Кафедра теоретической и прикладной информатики | | |
| Практическая работа № 4 | | |
| по дисциплине «Теория вычислительных процессов и структур» | | |
| **Инвариант цикла** | | |
|  | | |
|  | Бригада №6 | Ерощенко Артем, Кауфман Яна, Цыренова Сарюна |
| Группа ПМИ-33 |  |
|  |  |
|  |  |
|  |  |
|  |  |
| Преподаватель | Дворецкая виктория константиновна |
|  |  |
| Новосибирск,2024 | | |

**1.** **Цель работы:** получить практические навыки по моделированию автоматов и рекурсивных схем.

**2. Задание:** дана строка из букв “a” и “b”. Разработать машину Тьюринга, которая переместит все буквы “a” в левую, а буквы “b” — в правую части строки. Автомат в состоянии q1 обозревает крайний левый символ строки. Кроме самой программы-таблицы, описать словами, что выполняется машиной в каждом состоянии.

Для этой задачи выполнить следующие условия:

**Задание 1.** Постройте протокол выполнения рекурсивной программы для следующей рекурсивной схемы:

RS3: F(x); F(x) = **if**  p(x) **then** x else f(F(g(x)), F(h(x))).

**Задание 2.**

Построить протокол выполнения рекурсивной схемы

F(a),

F(x) = если p(x) то f(x) иначе F(F(g(x)))

При интерпретации I: I(a) = 7,

I(p)(d) =

**Задание 3**.

- Построить двухголовочный автомат, допускающий слова в алфавите (а,х), в которых количество букв а и х одинаково.

- Построить соответствующий двухголовочному автомату двоичный автомат и ССП этого двоичного автомата.

**3. Анализ задачи:**

**Входные данные:** строка из букв “a” и “b”

**Выходные данные:** строка, где все буквы “a” слева, а буквы “b” справа.

**Метод решения:**

**4. Ход выполнения работы:**

1. Входное значение: x.

2. Вычисляем p(x):

- Если p(x) истинно, то возвращаем x.

- Иначе переходим к следующему шагу.

3. Вычисляем F(g(x)) и F(h(x)) по очереди:

3.1. Вычисляем F(g(x)):

- Выполняем шаги с 1 по 3 для g(x).

3.2. Вычисляем F(h(x)):

- Выполняем шаги с 1 по 3 для h(x).

4. Вычисляем F(x):

- Возвращаем результат, полученный на шаге 2, если p(x) истинно.

- Иначе возвращаем результат выполнения рекурсивной схемы RS3 для F(g(x)) и F(h(x)).

2. Построение протокола выполнения рекурсивной схемы S\_8.4:  
   - Напишем функцию F(x), которая будет рекурсивно вычислять значение согласно определению S\_8.4 и интерпретации I.  
   - Используем интерпретацию I для вычисления значений функций p(x), f(), g() и базового случая.  
   - Продолжим вычисления, пока не достигнем базового случая.

**5. Код программы:**

**Задачи 1 и 2**

def move\_a\_to\_start(input\_str):

tape = list(input\_str)

tape = sorted(tape, key=lambda x: x == 'а', reverse=True) # Сортируем символы, перемещая 'а' в начало

return ''.join(tape)

def move\_b\_to\_end(input\_str):

tape = list(input\_str)

tape = sorted(tape, key=lambda x: x != 'б') # Сортируем символы, перемещая 'б' в конец

return ''.join(tape)

def turing\_machine(input\_str):

tape = list(input\_str) # Преобразуем входную строку в список символов

state = 'q1' # Начальное состояние

head = 0 # Позиция головки на ленте

while True:

if state == 'q1':

if tape[head] == 'а':

tape.insert(0, tape.pop(head)) # Перемещаем 'а' влево

head += 1

elif tape[head] == 'б':

tape.append(tape.pop(head)) # Перемещаем 'б' вправо

head += 1

if head >= len(tape): # Проверка на достижение края строки

head = 0 # Возвращаемся к началу строки

state = 'q1' # Возвращаемся в начальное состояние

if all(char == 'б' for char in tape): # Если все символы стали 'б', завершаем работу

break

return ''.join(tape) # Преобразуем список обратно в строку

# Пример использования

input\_string = 'абааббб'

result\_a\_to\_start = move\_a\_to\_start(input\_string)

result\_b\_to\_end = move\_b\_to\_end(input\_string)

print("Перемещение всех 'а' в начало строки:", result\_a\_to\_start)

print("Перемещение всех 'б' в конец строки:", result\_b\_to\_end)

**Задача 3**

Для начала построим двухголовочный автомат, который будет допускать слова в алфавите {a, x}, в которых количество букв 'a' и 'x' одинаково.

Построим автомат:

Начальное состояние: q0

Состояния: q0, q1, q2

Алфавит: {a, x}

Переходы:

(q0, a, →, q1)

(q0, x, →, q2)

(q1, a, →, q0)

(q1, x, →, q1)

(q2, a, →, q2)

(q2, x, →, q0)

Допускающее состояние: q0

Этот автомат будет перемещаться по входной строке с двумя головками. Одна головка будет считать буквы 'a', а другая - 'x'. Если количество 'a' и 'x' окажется одинаковым, то автомат примет слово.

Теперь построим соответствующий данному автомату двоичный автомат и его ССП:

Двоичный автомат будет иметь три бита для представления состояний: q0 (00), q1 (01), q2 (10).

Для кодирования переходов используем функцию h: Q × Σ → Q, где Q - множество состояний, а Σ - алфавит.

Функция h задается следующим образом:

h(q0, a) = q1 (01)

h(q0, x) = q2 (10)

h(q1, a) = q0 (00)

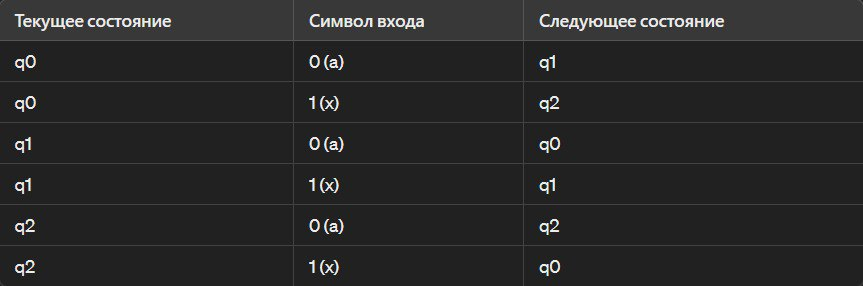
h(q1, x) = q1 (01)

h(q2, a) = q2 (10)

h(q2, x) = q0 (00)

По этой функции строится таблица переходов двоичного автомата.

ССП (состояние-символ-переход) данного автомата будет выглядеть так:



class TwoHeadedAutomaton:

def init(self):

self.states = {'q0', 'q1', 'q2'}

self.alphabet = {'a', 'x'}

self.transitions = {

('q0', 'a'): 'q1',

('q0', 'x'): 'q2',

('q1', 'a'): 'q0',

('q1', 'x'): 'q1',

('q2', 'a'): 'q2',

('q2', 'x'): 'q0'

}

self.start\_state = 'q0'

self.accept\_state = 'q1' # Изменение допускающего состояния

def accepts(self, word):

state = self.start\_state

count\_a = 0

count\_x = 0

for symbol in word:

if symbol not in self.alphabet:

return False

if symbol == 'a':

count\_a += 1

elif symbol == 'x':

count\_x += 1

state = self.transitions.get((state, symbol), None)

if state is None:

return False

return count\_a == count\_x and state == self.accept\_state # Проверка количества 'a' и 'x'

# Пример использования

automaton = TwoHeadedAutomaton()

words\_to\_test = ['aaxaax', 'xaxaax', 'ax', 'aaxaxx']

for word in words\_to\_test:

if automaton.accepts(word):

print(f"Слово '{word}' принято автоматом.")

else:

print(f"Слово '{word}' не принято автоматом.")

**6. Схема моделирующая автомат:**

Старт

1

0

0

0

0

0

1

Петля

1

0

0

1

0

0

0

1

1

0