Федеральное государственное автономное образовательное учреждение высшего образования «Национальный исследовательский университет ИТМО»

Факультет программной инженерии и компьютерной техники

# Разработка компиляторов

Домашнее задание №1

Вариант 13

Выполнил:

Маликов Глеб Игоревич

Группа № P3324

Преподаватель:

Лаздин Артур Вячеславович

Санкт-Петербург

2025

# Оглавление

Задание ..... 3
Задание 5 ..... 3
Задание 22 ..... 3
Задание 37 ..... 3
Выполнение ..... 5
Задание 5 ..... 5
Задание 22 ..... 5
Задание 37 ..... 6
Заключение ..... 11

## Задание

Для каждой грамматики из списка, соответствующему варианту, определить тип грамматики по классификации Хомского, для грамматик типа 2 и 3 необходимо привести вывод не менее двух предложений языка, принадлежащих языку, порождаемому грамматикой.

Укажите язык, порождаемый грамматикой, в множественно-теоретическом виде.

| Вариант | Задачи | Вариант | Задачи |
|---|---|---|---|
| 1 | 1, 10, 30 | 11 | 3, 20, 39 |
| 2 | 2, 12, 31 | 12 | 4, 21, 38 |
| 3 | 3, 11, 32 | 13 | 5, 22, 37 |
| 4 | 4, 13, 33 | 14 | 6, 23, 36 |
| 5 | 5, 14, 34 | 15 | 7, 24, 35 |
| 6 | 6, 15, 35 | 16 | 8, 25, 34 |
| 7 | 7, 16, 36 | 17 | 9, 26, 33 |
| 8 | 8, 17, 37 | 18 | 10, 27, 32 |
| 9 | 9, 18, 38 | 19 | 11, 28, 31 |
| 10 | 10, 19, 39 | 20 | 13, 2, 30 |

*Таблица 1 - Таблица вариантов*

## Задание 5

$S \rightarrow aSL \mid al$

$L \rightarrow Kc$

$cK \rightarrow Kc$

$K \rightarrow b$

## Задание 22

$S \rightarrow abc \mid aSQ$

$bQc \rightarrow bbcc$

$cQ \rightarrow Qc$

## Задание 37

Для алфавита A = {a, b, c} необходимо сконструировать регулярную грамматику и детерминированный конечный автомат для языка из символов этого алфавита:

Множество строк нечетной длины, которые содержат подстроку bb.

Допускается построение ДКА по регулярной грамматике, и регулярной грамматике по ДКА.

Для полученного ДКА необходимо написать функцию / программу, моделирующую поведение этого автомата. В качестве тестов нужно представить не менее четырех корректных входных цепочек и цепочек, не допускаемых данным автоматом.

## Выполнение

### Задание 5

S → aSL | aL

Справа количество терминальных и нетерминальных не соответствует третьему типу, соответственно грамматика уже второго вида.
L → Kc

Соответствует третьему типу.
cK → Kc

Не соответствует второму типу, так как слева два символа. При этом не укорачивает. Получается грамматика первого типа.
K → b

Соответствует третьему типу.

Так грамматика по худшему правилу, является грамматикой первого вида (Контекстно-зависимый язык).

Язык, порождённый данной грамматикой:

По первому и второму правилу получается: $a^n(Kc)^n$
Далее первый символ K всегда будет преобразовано в b, то есть $a^n b(cK)^n$. С помощью правила cK → Kc можно сделать перестановки этих символов, получая $a^n b(Kc)^{n-1}$. В итоге все $K$ будут «передвинуты» на левую сторону от $c$. Так язык определяется как $\{a^n b^n c^n \mid n \in \mathbb{N}\}$

### Задание 22

S → abc | aSQ

Второй тип. Слева только один нетерминал и справа не короче, чем слева.
bQc → bbcc

Первый тип. Справа не короче, чем слева.
cQ → Qc

Первый тип. Справа не короче, чем слева.

Грамматика является контекстно-зависимой, то есть первого типа.

Язык, порождённый данной грамматикой:

Начиная от символа S, можно заметить, что порождается язык $\{a^n b^n c^n \mid n \in \mathbb{N}\}$. Ниже приводится пример последовательной замены символов.

S
↳ abc
↳ aSQ
↳ aabcQ
↳ aabQc
↳ aabbcc
↳ aaSQQ
↳ aaabcQQ
↳ aaabQcQ
↳ aaabbccQ
↳ aaabbcQc
↳ aaabbQcc
↳ aaabbbccc

## Задание 37

Сначала построим детерминированный конечный автомат (ДКА). Всего в таком языке 6 возможных состояний при его чтении:

1. Ноль или чётное число символов;
2. Нечётное число символов;
3. Чётное число символов, последний символ был ‘b’;
4. Нечётное число символов, последний символ был ‘b’;
5. Чётное число символов, в тексте уже были два символа ‘bb’;
6. Нечётное число символов, в тексте уже были два символа ‘bb’;

Таким образом построен следующий ДКА (рисунок 1):

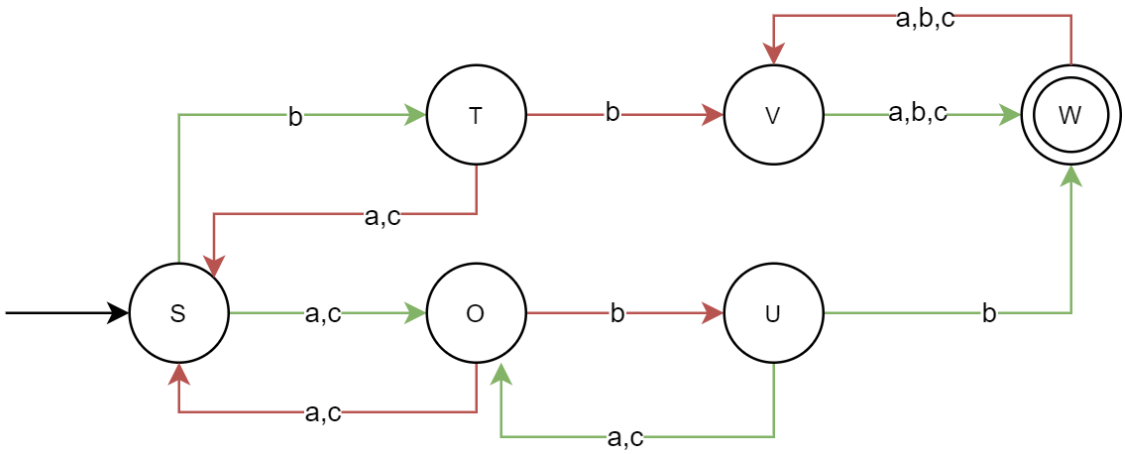


*Рисунок 1 - ДКА языка*

На схеме используется зелёный цвет для переходов, в которых будет нечётное количество символов, красный в переходах, где будет чётное.

Преобразуем данный конечный автомат в регулярную грамматику.

S – aO | bT | cO
O – aS | bU | cS
U – aO | bW | cO
T – aS | bV | cS
V – aW | bW | cW
W – aV | bV | cV | $\varepsilon$

Моделирование данного автомата сделано на языке Python. Реализация приведена ниже.

```
from transitions import MachineError
from transitions.extensions import GraphMachine


states = [
    'S',
    'O',
    'U',
    'T',
    'V',
    'W',
]

transitions_list = [
    {'trigger': 'a', 'source': 'S', 'dest': 'O'},
    {'trigger': 'b', 'source': 'S', 'dest': 'T'},
    {'trigger': 'c', 'source': 'S', 'dest': 'O'},
```

```
    {'trigger': 'a', 'source': 'O', 'dest': 'S'},
    {'trigger': 'b', 'source': 'O', 'dest': 'U'},
    {'trigger': 'c', 'source': 'O', 'dest': 'S'},

    {'trigger': 'a', 'source': 'U', 'dest': 'O'},
    {'trigger': 'b', 'source': 'U', 'dest': 'W'},
    {'trigger': 'c', 'source': 'U', 'dest': 'O'},

    {'trigger': 'a', 'source': 'T', 'dest': 'S'},
    {'trigger': 'b', 'source': 'T', 'dest': 'V'},
    {'trigger': 'c', 'source': 'T', 'dest': 'S'},

    {'trigger': 'a', 'source': 'V', 'dest': 'W'},
    {'trigger': 'b', 'source': 'V', 'dest': 'W'},
    {'trigger': 'c', 'source': 'V', 'dest': 'W'},

    {'trigger': 'a', 'source': 'W', 'dest': 'V'},
    {'trigger': 'b', 'source': 'W', 'dest': 'V'},
    {'trigger': 'c', 'source': 'W', 'dest': 'V'},
]


class DFA(GraphMachine):
    def __init__(self):
        super(DFA, self).__init__(states=states,
transitions=transitions_list, initial='S', auto_transitions=False,
graph_engine='graphviz')

    def accepts(self, input_string):
        """
        Attempt to consume the input_string symbol by symbol.
        Return True if we end in W, False otherwise.
        """
        # Reset to the start state (S)
        self.set_state('S')

        for symbol in input_string:
            if symbol not in ['a', 'b', 'c']:
                return False
            try:
                self.trigger(symbol)
            except MachineError:
                return False

        # Accept if (and only if) we end in W
        return self.state == 'W'


if __name__ == "__main__":
    dfa = DFA()
    dfa.get_graph().draw('dfa_transitions.png', prog='dot')
```

Данный конечный автомат был также нарисован с помощью класса *GraphMachine* и движка *graphviz*. Результат приведён ниже (рисунок 2).

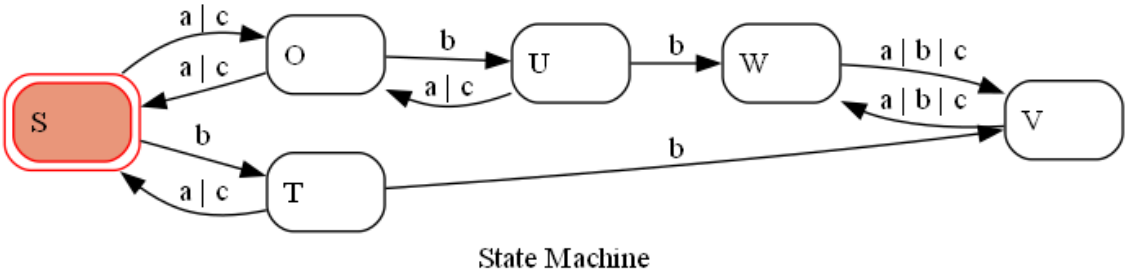


*Рисунок 2 - Конечный автомат, изображённый с помощью graphviz*

Для проверки корректности работы были сделаны следующие тесты:

```
import unittest
from main import DFA


class TestDFA(unittest.TestCase):
    def setUp(self):
        self.dfa = DFA()


    def test_trivial_accept(self):
        accepted = ["abb", "cbb", "bbb", "bbc"]
        for string in accepted:
            with self.subTest(input=string):
                self.assertTrue(self.dfa.accepts(string), f"Цепочка
'{string}' должна приниматься")

    def test_trivial_reject(self):
        rejected = ["", "a", "c", "bb"]
        for string in rejected:
            with self.subTest(input=string):
                self.assertFalse(self.dfa.accepts(string), f"Цепочка
'{string}' не должна приниматься")


    def test_nontrivial_accept(self):
        accepted = ["abbccab", "bcbbbac", "cabbabaaaccba", "babcaccabbc"]
        for string in accepted:
            with self.subTest(input=string):
                self.assertTrue(self.dfa.accepts(string), f"Цепочка
'{string}' должна приниматься")

    def test_nontrivial_reject(self):
        rejected = ["abccbabb", "abbb", "babab", "babb", "ccbcccbbaa"]
        for string in rejected:
            with self.subTest(input=string):
```

```
                self.assertFalse(self.dfa.accepts(string), f"Цепочка
'{string}' не должна приниматься")


if __name__ == '__main__':
    unittest.main()
```

## Заключение

В данной лабораторной работе были решены три задачи, каждая из которых позволила углубленно изучить различные аспекты формальных языков и автоматов.

Был проведён анализ грамматик по классификации Хомского, в результате которого удалось определить, что, несмотря на наличие смешанных правил, обе грамматики относятся к контекстно-зависимому типу. Порождаемым языком обеих грамматик является множество $\{a^n b^n c^n \mid n \in \mathbb{N}\}$.

Для языка строк нечетной длины, содержащих подстроку «bb», был сконструирован детерминированный конечный автомат, состоящий из шести состояний. На основе этого автомата была построена соответствующая регулярная грамматика.

Был смоделирован автомат на языке Python с использованием библиотеки *transitions*. Реализованная программа успешно показывает поведение ДКА, а проведённый модульный тестинг подтвердил корректность работы автомат.