Санкт-Петербургский национальный исследовательский университет
информационных технологий, механики и оптики

Кафедра информатики и прикладной математики

Домашняя работа №2
Дисциплина «Прикладная математика»

Вариант 21

Выполнил:
Съестов Дмитрий Вячеславович
Группа P3317

Преподаватель:
Лаздин Артур Вячеславович

Санкт-Петербург
2018

Регулярное выражение: ((a?b?)|(bc))*cb

Задание:

По заданному регулярному выражению
• Определить язык (регулярное множество), порождаемый регулярным выражением
• Построить недетерминированный КА
• По полученному НДА построить ДКА
• Минимизировать полученный ДКА
• Для мин. ДКА написать программу-распознаватель предложений языка, порождаемого регулярным выражением

Регулярное множество: { cb, acb, bcb, abcb, bccb, abccb, bbccb, abbccb… }

Автоматы: см. приложение

Программа-распознаватель (Python):

```
class State:
    transitions = {}

    def __init__(self, transitions):
        self.transitions = transitions


class InitState(State):
    pass


class TerminalState(State):
    pass


class FSM:
    states = {}
    initState = InitState({})

    def __init__(self, states):
        self.states = states

        state_values = dict.values(states)
        initStates = [s for s in state_values if isinstance(s, InitState)]
        terminalStates = [s for s in state_values if isinstance(s, TerminalState)]

        assert len(initStates) == 1
        assert any(terminalStates)

        self.initState = initStates[0]
```

```
    def doTransition(self, state, char):
        if state is not None:
            next_state = state.transitions.get(char)
            return self.states.get(next_state)
        else:
            return None

    def parse(self, sentence):
        state = reduce(self.doTransition, sentence, self.initState)
        return isinstance(state, TerminalState) if state is not None else False

fsm = FSM({
    'A': InitState({'a': 'A', 'b': 'B', 'c': 'D'}),
    'B': State({'a': 'A', 'b': 'B', 'c': 'C'}),
    'C': State({'a': 'A', 'b': 'F', 'c': 'D'}),
    'D': State({'b': 'E'}),
    'E': TerminalState({}),
    'F': TerminalState({'a': 'A', 'b': 'B', 'c': 'C'})
})
```

Модульные тесты:

```
assert fsm.parse("cb")
assert fsm.parse("bccb")
assert fsm.parse("bcbccb")
assert fsm.parse("acb")
assert fsm.parse("bcb")
assert fsm.parse("abcb")
assert fsm.parse("aabcb")
assert fsm.parse("bbbcb")
assert fsm.parse("aaaaaaababababbcbcababababcbbbbbcb")
assert not fsm.parse("c")
assert not fsm.parse("bcccb")
assert not fsm.parse("whatever")
```