

# ТМВ Домашнее задание №3

А-136-19 Головин Антон

29 мая 2022

## 2. Машины Тьюринга

### 2.1 Операции с языками и символами

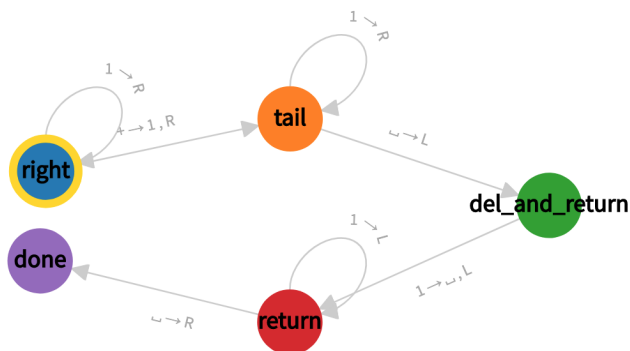
Реализуйте машины Тьюринга, которые позволяют выполнять следующие операции:

#### 1. Сложение двух унарных чисел (1 балл)

Алгоритм:

- Движемся вправо, пока не встретили '+'.  
• Заменяем '+' на 1 и движемся к хвосту.  
• Находим крайнюю единицу и удаляем её.  
• Движемся к голове.

```
# 2_1_1.yaml
input: '111+1'
blank: ' '
start state: right
table:
  right:
    1: R
    '+': {write: '1', R: tail}
  tail:
    1: R
    ' ': {L: del_and_return}
  del_and_return:
    1: {write: ' ', L: return}
  return:
    1: L
    ' ': {R: done}
  done:
```



## 2. Умножение унарных чисел (1 балл)

Копируем за знак '=' единицы первого множителя столько раз, сколько единиц во втором.

Алгоритм:

- Двигаемся вправо и находим '\*', потом помечаем первую найденную единицу второго множителя крестиком 'x'.
- Движемся влево и начинаем копирование - все единицы первого множителя помечаем крестиком и записываем после знака '='.
- Пока мы не встретили 'λ' в начале, продолжаем копирование.
- Встретили 'λ' - восстанавливаем единицы первого множителя (пока не встретим '\*').
- Повторяем шаги (помечаем 'x' единицы второго множителя, пока это возможно).
- Когда не осталось единиц, умножение завершено, восстанавливаем все единицы.

```
# 2_1_2.yaml
input: '111*11'
blank: ' '
start state: check-correct

table:

# проверка случаев 1* и *1
check-correct:
  1: {R: check-star}
  '*': {L: check-empty}
check-star:
  1: R
  '*': {R: check-empty}
check-empty:
  ' ': {R: done}
  1: {L: go-start}

go-start:
  [1, '*', '=']: {L: go-start}
  ' ': {R: preload}

preload:                                # добавляем =
  [1, '*']: R
  ' ': {write: '=', R: -end}
-end:                                   # добавляем в конец
  ' ': {write: '', L: -start}
  [1, '=', '*']: L
-start:                                 # добавляем в начало
  ' ': {write: '', R: start}
  ['', '=', '*', 1]: L

start:
  1: R
  ['x', '*']: {R: process-second}

process-second:
  'x': R
  1: {write: 'x', L: go-first}
```

```

'=': {L: clear--start}    # копировать нечего

remain-ones:              # проверяем, нужно ли нам копировать снова
[1, '=']: {L: go-first}

go-first:
['x', '*']: L
1: {write: 'x', R: copy}
': {R: restore}

restore:                  # очередное копирование завершено, восстанавливаем 1
'x': {write: 1, R: restore}
'*': {R: start}

copy:
[1, '*', 'x', '=']: R
': {write: 1, R: go-end}

go-end:
' ': {write: ' ', L: go-second}

go-second:
[1, 'x', '=']: L
'*': {L: go-first}

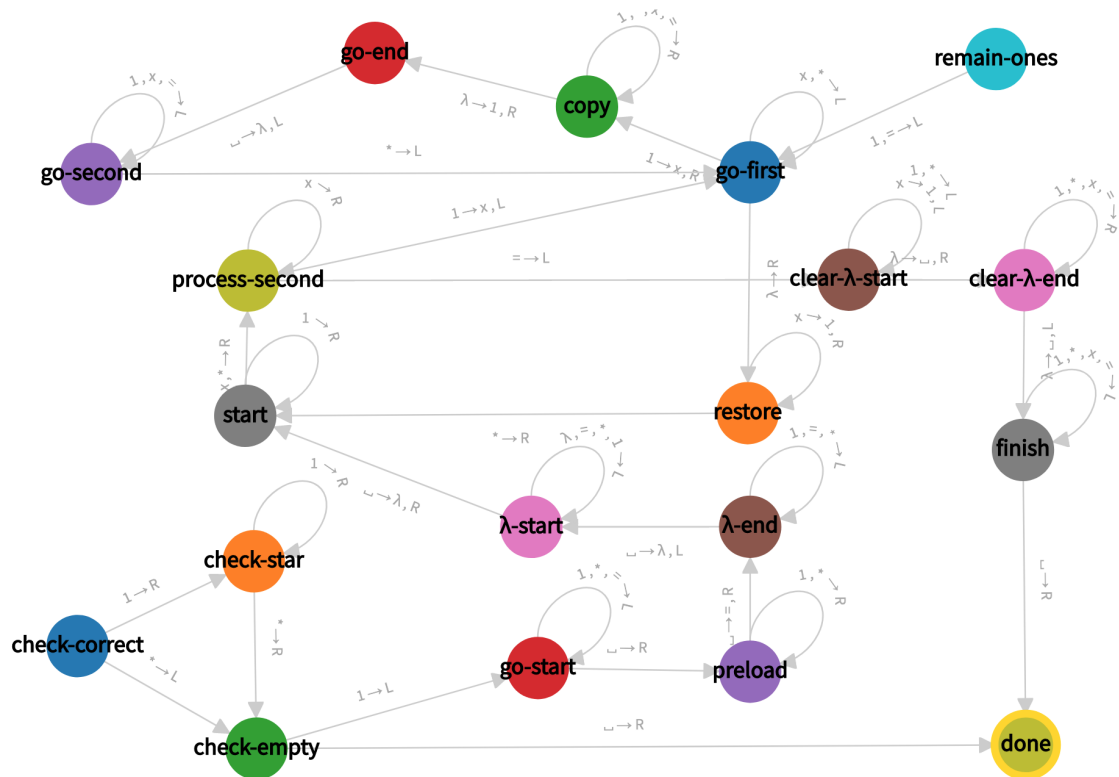
clear--start:
'x': {write: 1, L: clear--start}
[1, '*']: L
': {write: ' ', R: clear--end}

clear--end:
[1, '*', 'x', '=']: R
': {write: ' ', L: finish}

finish:
[1, '*', 'x', '=']: L
' ': {R: done}

done:

```



## 2.2 Операции с языками и символами

Реализуйте машины Тьюринга, которые позволяют выполнять следующие операции:

1. Принадлежность к языку  $L = \{0^n 1^n 2^n\}, n \geq 0$  (0.5 балла)

Алгоритм:

- Все первые вхождения 0, 1, 2 заменяем на 'x'. Возвращаемся в начало.
- Повторяем шаг выше, пока слово не будет заменено на все 'x' (иначе слово не принадлежит языку).
- 1 - слово принадлежит языку, 0 - нет
- По условию  $n$  может быть равно 0, поэтому пустое слово тоже принадлежит языку.

```
# 2_2_1.yaml
input: '012012'
blank: ' '
start state: start

table:
  start:
    ' ': {L: ok}      # слово принадлежит языку
    0: {write: 'x', R: go-one}
    [1, 2]: {R: not-ok}
    'x': R            # новый проход

  go-one:
    [0, 'x']: R
    1: {write: 'x', R: go-two}
    [2, ' ']: {L: not-ok}
```

```

go-two:
  ' ': {L: not-ok}
  [1, 'x']: R
  2: {write: 'x', L: go-start}

go-start:
  ' ': {R: start}
  [0, 1, 2, 'x']: L

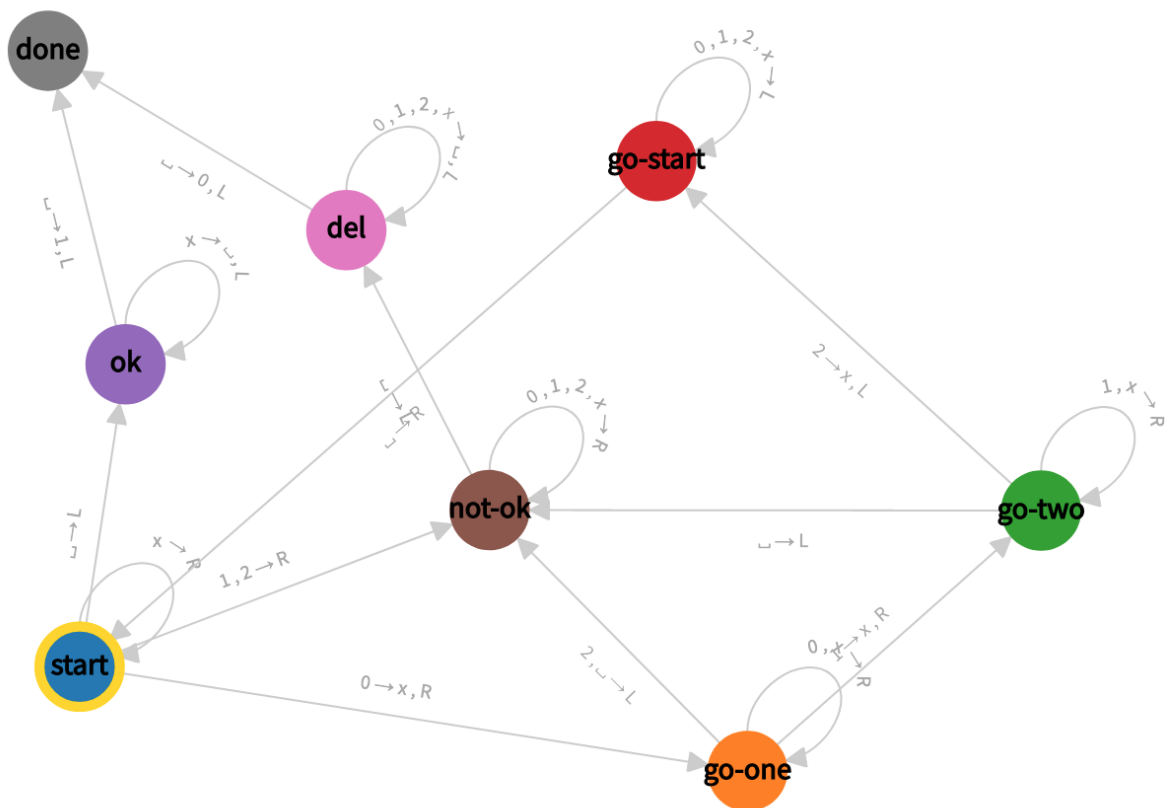
ok:
  ' ': {write: 1, L: done}
  'x': {write: ' ', L}

not-ok:
  ' ': {L: del}
  [0, 1, 2, 'x']: R

del:
  ' ': {write: 0, L: done}
  [0, 1, 2, 'x']: {write: ' ', L}

done:

```



2. Проверка соблюдения правильности скобок в строке (минимум 3 вида скобок) (0.5 балла)  
Алгоритм:

- Ищем первую закрывающуюся скобку. Заменяем её на 'x'. Возвращаемся в начало.
- Ищем открывающуюся скобку такого же вида. Заменяем её на 'x'. Возвращаемся в начало.
- 1 - слово принадлежит языку (все 'x'), 0 - нет
- Как и в предыдущем номере, пустое слово - правильная скобочная последовательность.

```
# 2_2_2.yaml
input: '([{}])'
blank: ' '
start state: start

table:
  start:
    ' ': {L: ok}      # пустая скобочная послед
    '[': {R: find-closed}
    '(': {L: not-ok}

  find-closed:
    ' ': {L: empty-or-ok}      # вышли за границы слова или не нашли закрывающуюся скоб
    '[': {R: find-closed}
    '(': {write: 'x', L: closed_1}
    ']': {write: 'x', L: closed_2}
    ')': {write: 'x', L: closed_3}

  closed_1:
    ' ': {R: not-ok}
    '[': {write: 'x', R: find-closed}
    '(': {L: not-ok}
    'x': L

  closed_2:
    ' ': {R: not-ok}
    '[': {write: 'x', R: find-closed}
    '(': {L: not-ok}
    'x': L

  closed_3:
    ' ': {R: not-ok}
    '[': {write: 'x', R: find-closed}
    '(': {L: not-ok}
    'x': L

  empty-or-ok:
    '[': {L: not-ok} # всё-таки есть необработанная скобка
    'x': L
    ' ': {R: ok}

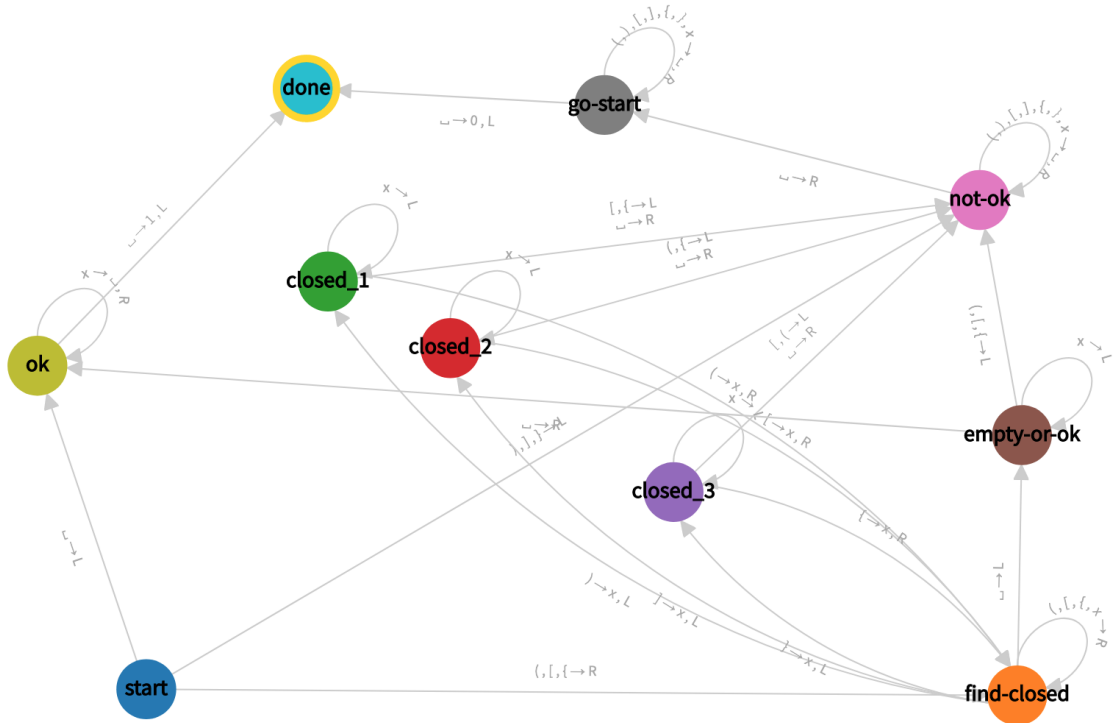
  not-ok:
    '(': {write: ' ', R}
    ' ': {R: go-start}
    # в начало, чтобы очистить ленту
  go-start:
    '(': {write: ' ', R: go-start}
    ' ': {write: 0, L: done}
```

```

ok:
  ' ': {write: 1, L: done}
  'x': {write: ' ', R}

done:

```



3. Поиск минимального по длине слова в строке (слова состоят из символов 1 и 0 и разделены пробелом) (1 балл)

Алгоритм:

- Заменяем в каждом слове один раз за проход 0 на 'a', 1 на 'b'.
- Справа налево аналогично
- Если в слове нечего заменять, считаем, что оно минимальное (ставим '+' после слова)
- Пустое слово - минимальное.

```

# 2_2_3.yaml
input: '110 01 11'
blank: ' '
start state: word
table:
  word:
    0: {write: 'a', R: next}
    1: {write: 'b', R: next}
    ['a', 'b', '']: R
    ' ': {write: '+', L: finish}

# один раз заменяем за проход
next:
  [0, 1, 'a', 'b']: R
  ' ': {write: ' ', L: go_0} # разворачиваемся

```

```

' ': {write: ' ', R: word}

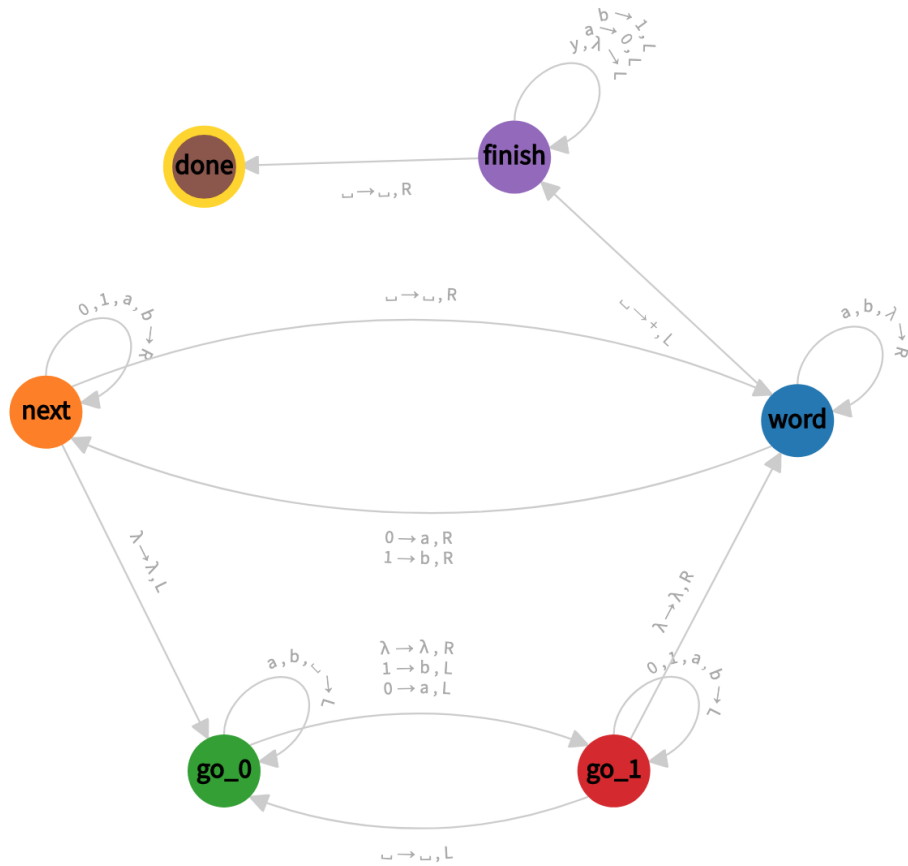
go_0:
['a','b', ' ']: L
0: {write: 'a', L: go_1}
1: {write: 'b', L: go_1}
' ': {write: ' ', R: go_1}

go_1:
[0, 1, 'a','b']: L
' ': {write: ' ', L: go_0}
' ': {write: ' ', R: word}

finish:
['y','']: L
'a': {write: '0', L}
'b': {write: '1', L}
' ': {write: ' ', R: done}

done:

```





## 3 Квантовые вычисления

### 3.1 Генерация суперпозиций 1 (1 балл)

Дано  $N$  кубитов ( $1 \leq N \leq 8$ ) в нулевом состоянии  $|0 \dots 0\rangle$ . Также дана некоторая последовательность битов, которое задаёт ненулевое базисное состояние размера  $N$ . Задача получить суперпозицию нулевого состояния и заданного.

$$|S\rangle = \frac{1}{\sqrt{2}}(|0 \dots 0\rangle + |\psi\rangle)$$

То есть, требуется реализовать операцию, которая принимает на вход:

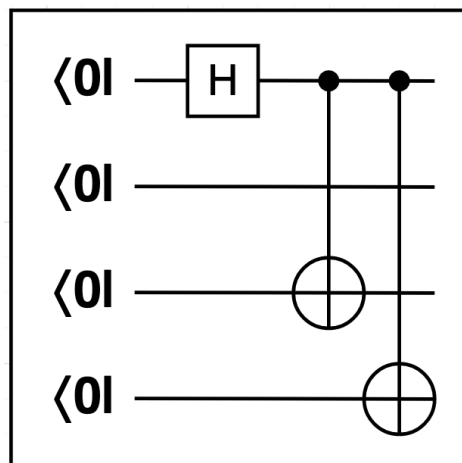
1. Массив кубитов  $q_s$
2. Массив битов  $bits$  описывающих некоторое состояние  $|\psi\rangle$ . Это массив имеет тот же самый размер, что и  $q_s$ . Первый элемент этого массива равен 1.

Пояснения:

- В начале у нас есть  $N$  независимых кубитов  $|0\rangle$
- Первые кубиты векторов различны, применим оператор Адамара к первому кубиту
- Все кубиты  $q_s$  равны 0,  $\Rightarrow$  если кубит  $bits[i] = 1$ , то нужно запутать  $i$ -ый кубит, а если кубит  $bits[i] = 0$ , то не нужно, т.к кубиты совпадают и равны 0.

Код

```
namespace Solution {
    open Microsoft.Quantum.Primitive;
    open Microsoft.Quantum.Canon;
    operation Solve (qs : Qubit[], bits : Bool[]) : Unit
    {
        body
        {
            H(qs[0]);
            for i in 1..Length(qs) - 1 {
                if (bits[i]) {
                    CNOT(qs[0], qs[i]);
                }
            }
        }
    }
}
```



### 3.2 Различение состояний 1 (1 балл)

Дано  $N$  кубитов ( $1 \leq N \leq 8$ ), которые могут быть в одном из двух состояний:

$$|GHZ\rangle = \frac{1}{\sqrt{2}}(|0\dots 0\rangle + |1\dots 1\rangle)$$

$$|W\rangle = \frac{1}{\sqrt{N}}(|10\dots 00\rangle + |01\dots 00\rangle + \dots + |00\dots 01\rangle)$$

Требуется выполнить необходимые преобразования, чтобы точно различить эти два состояния. Возвращать 0, если первое состояние и 1, если второе.

Пояснения:

- Чтобы измерить состояние системы надо измерить кубиты
- При  $N > 1$  состояние 1:  $N$  нулей, либо  $N$  единиц, состояние 2: 1 единица
- При  $N = 1$  состояния не различить (в обоих состояниях может выпасть вектор, который содержит одну единицу)

Код

```
namespace Solution {
    open Microsoft.Quantum.Primitive;
    open Microsoft.Quantum.Canon;
    operation Solve (qs : Qubit[]) : Int
    {
        body
        {
            mutable ones = 0;
            for i in 0..Length(qs) - 1 {
                if (M(qs[i]) == One) { // measurement
                    set ones += 1;
                }
            }
            if (ones == 1) {
                return 1;
            }
            return 0;
        }
    }
}
```

### 3.3 Различение состояний 2 (2 балл)

Дано 2 кубита, которые могут быть в одном из двух состояний:

$$|S_0\rangle = \frac{1}{2}(|00\rangle + |01\rangle + |10\rangle + |11\rangle)$$

$$|S_1\rangle = \frac{1}{2}(|00\rangle - |01\rangle + |10\rangle - |11\rangle)$$

$$|S_2\rangle = \frac{1}{2}(|00\rangle + |01\rangle - |10\rangle - |11\rangle)$$

$$|S_3\rangle = \frac{1}{2}(|00\rangle - |01\rangle - |10\rangle + |11\rangle)$$

Требуется выполнить необходимые преобразования, чтобы точно различить эти четыре состояния. Возвращать требуется индекс состояния (от 0 до 3).

**Код**

```
namespace Solution {
    open Microsoft.Quantum.Primitive;
    open Microsoft.Quantum.Canon;
    operation Solve (qs : Qubit[]) : Int
    {
        body
        {
            return
        }
    }
}
```