ТМВ Домашнее задание №3

А-13б-19 Головин Антон

29 мая 2022

2. Машины Тьюринга

2.1 Операции с языками и символами

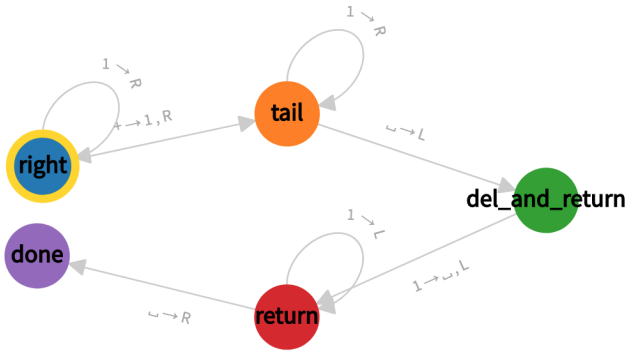
Реализуйте машины Тьюринга, которые позволяют выполнять следующие операции:

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.

```yaml
# 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)**

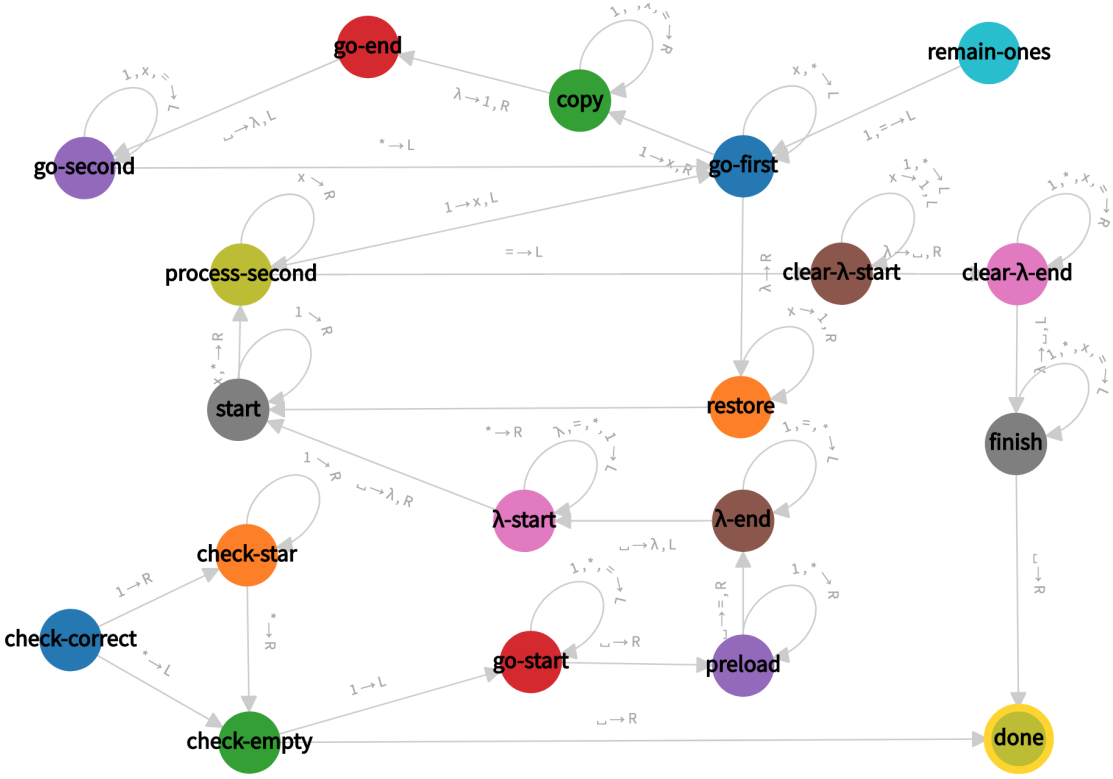
   Копируем за знак '=' единицы первого множителя столько раз, сколько единиц во втором.

   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, продолжам 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.

```yaml
# 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:

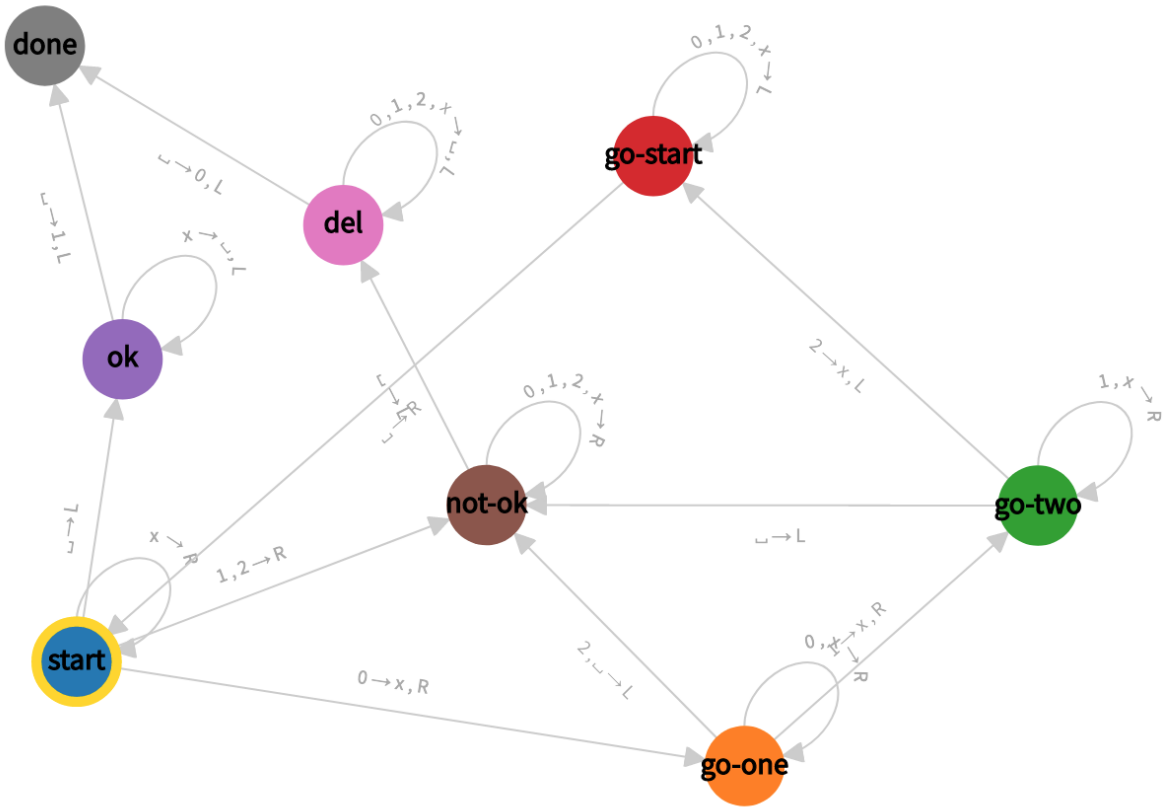
1. Принадлежность к языку $L = \{0^n 1^n 2^n\}, n \geq 0$ (0.5 балла)

   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 - слов принадлежит языку, 0 - нет
   - По усновию n может быть равно 0, поэтому пустое слово тоже принадлежит языку.

```yaml
# 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:
```

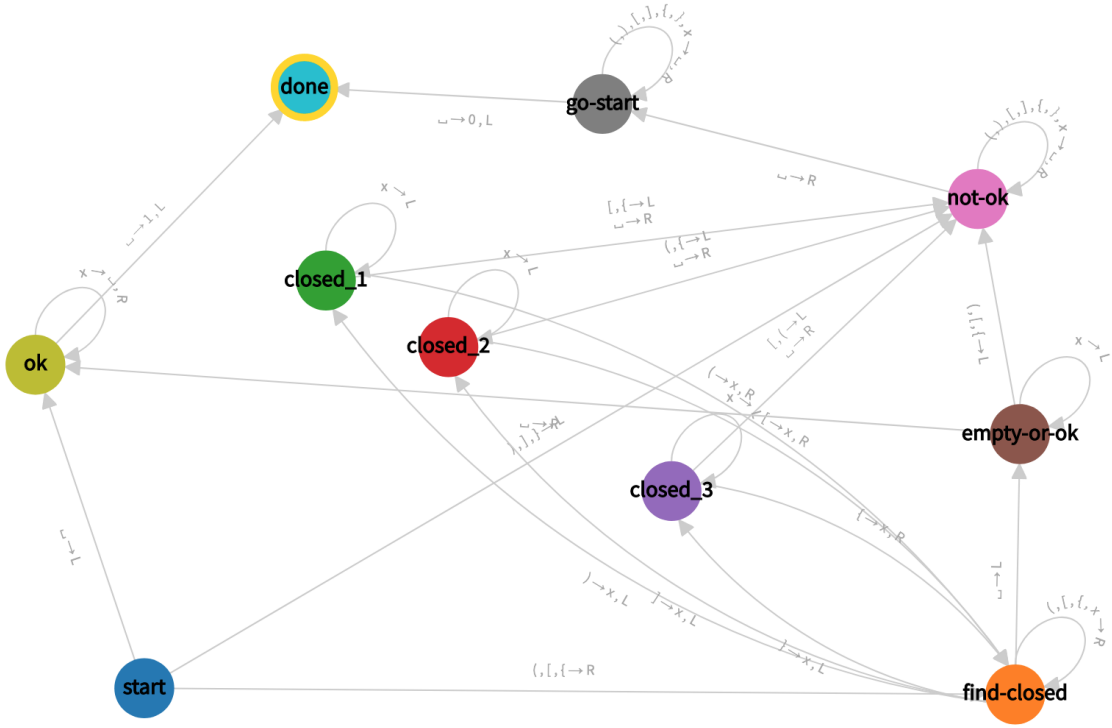


2. Проверка соблюдения правильности скобок в строке (минимум 3 вида скобок) (0.5 балла)
   Алгоритм:

- 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 - слов принадлежит языку (все 'x'), 0 - нет
- Как и в предыдущем номере, пустое слово - правильная скобочная последовательность.

```yaml
# 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 граицы слова или не нашли закрывающуюся скоб
    ['(', '[', '{', 'x']: R
    ')': {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:
    ['(', ')', '[', ']', '{', '}', 'x']: {write: ' ', R}
    ' ': {R: go-start}
  # в начало, чтобы очистить ленту
  go-start:
    ['(', ')', '[', ']', '{', '}', 'x']: {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)

3 Квантовые вычисления