Теоретические модели вычислений
ДЗ №3: Машины Тьюринга и квантовые вычисления

Расторгуев Александр, А-13а-19

1 июня 2022 г.

1 Машины Тьюринга

1.1 Операции с числами

1. Сложение двух унарных чисел

```
input: '11+1111'
blank: ' '
start state: lhs
table:
  lhs:
    1   : R
    +   : {R: rhs}
  rhs:
    ' ' : {L: cut}
    1   : {write: +, L}
    +   : {write: 1, R: lhs}
  cut:
    +   : {write: ' ', L: done}
  done:
```

2. Умножение двух унарных чисел

```
input: '*1'
blank: ' '
start state: start
table:
  start:
    '*' : {L: go_result_end}
    1   : {write: 1, R: go_rhs}
  go_rhs:
```

```
    [1, 1]   : R
    '*' : {R: go_rhs_one}
  go_rhs_one:
    r   : R
    ' ' : {L: go_lhs}
    1   : {write: r, L: drag_one}
  drag_one:
    [r, '*', 1, 1] : L
    ' '   : {write: 1, R: go_rhs}
  go_lhs:
    r   : {write: 1, L}
    '*' : {L: go_lhs_one}
  go_lhs_one:
    1   : L
    1   : {R: start}
  go_result_end:
    1   : L
    [1, ' '] : {R: clean_right}
  clean_right:
    [1, '*', 1] : {write: ' ', R}
    ' ' : {L: end}
  end:
```

1.2 Операции с языками и символами

1. Принадлежность к языку $L = \{0^n 1^n 2^n,\ n \geq 0\}$
   Если слово принадлежит языку, лента останется пустой.

```
input: '001122'
blank: ' '
start state: cutbegin
table:
  cutbegin:
    ' ' : {L: done}
    '*'   : {write: ' ', R}
    0   : {write: ' ', R: mark1}
  mark1:
    [0, '*']   : R
    1   : {write: '*', R: goend}
  goend:
    [1, 2]   : R
    ' ' : {L: cutend}
  cutend:
    2   : {write: ' ', L: gobegin}
  gobegin:
    [0, 1, 2, '*']   : L
```

```
    ' ' : {R: cutbegin}
  done:
```

2. Проверка соблюдения правильности скобок в строке (минимум 3 вида скобок)
   Если скобки расставлены правильно, лента останется пустой.

```
input: '{}[]({}{}{}))'
blank: ' '
start state: go_any_rp
table:
  go_any_rp:
    ['(', '[', '{', '*'] : R
    ')' : {write: '*', L: go_lp}
    ']' : {write: '*', L: go_slp}
    '}' : {write: '*', L: go_flp}
    ' ' : {L: clear}
  go_lp:
    '*' : L
    '(' : {write: '*', R: go_any_rp}
  go_slp:
    '*' : L
    '[' : {write: '*', R: go_any_rp}
  go_flp:
    '*' : L
    '{' : {write: '*', R: go_any_rp}
  clear:
    '*' : {write: ' ', L}
    ' ' : {L: end}
  end:
```

3. Поиск минимального по длине слова в строке (слова состоят из символов 0, 1 и разделены пробелом)
   Сравниваются первые 2 слова строки. Если первое меньше второго, второе слово стирается, первое передвигается на его место. Если второе слово меньше первого, первое слово стирается. Процесс повторяется, пока не останется одно наименьшее слово строки.

```
input: '1010 0 01 1111'
blank: ' '
start state: mark_l
table:
  mark_l:
    [o, l] : R
    0   : {write: o, R: go_fspace}
    1   : {write: l, R: go_fspace}
    ' ' : {R: rclear} # delete rhs, recover lhs
```

```
go_fspace:
  ' ' : {R: check_space}
  [1, 0] : R
check_space:
  ' ': {L: go_left_and_recover}
  [o, 1, 0, 1] : {L: go_one_right_and_mark_r} # delete rhs, recover lhs
go_one_right_and_mark_r:
  ' ': {R: mark_r}
mark_r:
  [o, 1] : R
  0   : {write: o, L: go_bspace}
  1   : {write: 1, L: go_bspace}
  ' ' : {L: rrecover} # delete lhs, recover rhs
go_bspace:
  [o, 1] : L
  ' ': {L: go_lbegin}
go_lbegin:
  [0, 1] : L
  [o, 1] : {R: mark_1}

rrecover:
  o   : {write: 0, L}
  1   : {write: 1, L}
  ' ' : {L: lclear}
lclear:
  [0, 1, o, 1] : {write: ' ', L}
  ' ' : {R: go_right}
go_right:
  ' ' : R
  [0, 1] : {L: go_one_right}
go_one_right:
  ' ' : {R: mark_1}
rclear:
  [0, 1, o, 1] : {write: ' ', R}
  ' ' : {R: check_space2}
check_space2:
  ' ': {L: go_left_and_recover}
  [o, 1, 0, 1] : {L: go_left}
go_left_and_recover:
  ' ' : L
  [o, 1, 0, 1] : {R: sedfsfsesgdgdg}
sedfsfsesgdgdg:
  ' ' : {L: recover_lhs_end}
recover_lhs_end:
  [0, 1] : L
  o : {write: 0, L}
```

```
  1 : {write: 1, L}
  ' ' : {R: end}
go_left:
  ' ' : L
  [0, 1, o, 1] : {L: go_mark_begin}
go_mark_begin:
  [o, 1, 0, 1] : L
  ' ' : {write: '*', R: go_lhs_end}
go_lhs_end:
  [o, 1, 0, 1] : R
  ' ' : {L: copy_lhs}
copy_lhs:
  [o, 0] : {write: ' ', R: begin_drag_0}
  [1, 1] : {write: ' ', R: begin_drag_1}
begin_drag_0:
  ' ' : R
  [0, 1] : {L: go_one_left_and_write_0}
go_one_left_and_write_0:
  ' ' : {L: lwrite_0}
lwrite_0:
  ' ' : {write: 0, L: continue_copy}
  #'*' : {write: ' ', R: go_restart}
go_restart:
  ' ' : R
  [0, 1] : {L: go_one_right_and_restart}
go_one_right_and_restart:
  ' ' : {R: mark_1}
continue_copy:
  ' ' : L
  [o, 0] : {write: ' ', R: drag_0}
  [1, 1] : {write: ' ', R: drag_1}
  '*' : {write: ' ', R: go_restart}
drag_0:
  ' ' : R
  [0, 1] : {L: lwrite_0}
begin_drag_1:
  ' ' : R
  [0, 1] : {L: go_one_left_and_write_1}
go_one_left_and_write_1:
  ' ' : {L: lwrite_1}
lwrite_1:
  ' ' : {write: 1, L: continue_copy}
  #'*' : {write: ' ', R: go_restart}
drag_1:
  ' ' : R
  [0, 1] : {L: lwrite_1}
```

```
end:
```

2 Квантовые вычисления

2.1 Генерация суперпозиций

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 qs. Первый элемент этого массива равен 1.

Решение
По условию задачи первый квант в векторах будет различаться. Применим к нему оператор Адамара, выбирем его в качестве управляющего кубита для гейта $CNOT$, который применим к управляемым кубитам - единичным кубитам вектора ψ.

```
operation Solve (qs : Qubit[], bits : Bool[]) : Unit
{
    H(qs[0]);
    for i in 1 .. Length(qs)-1 {
        if (bits[i]) {
            CNOT(qs[0], qs[i]);
        }
    }
}
```

2.2 Различение состояний 1

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.

Решение

Для определения состояния измерим кубиты. Если система находилась в состоянии $|GHZ\rangle$, при измерении получим N нулей, либо N единиц. Если система находилась в состоянии W, получим одну единицу и $N-1$ нулей. Поэтому при $N>1$ самого факта измерения будет достаточно для различения состояний. При $N=1$ мы могли бы применить матрицу поворота для повышения точности различения состояния системы, однако это не даст точности в 100%.

Код для $N>1$:

```
operation Solve (qs : Qubit[]) : Int
{
    mutable ones = 0;
    for i in 0 .. Length(qs)-1 {
        if(M(qs[i]) == One) {
            set ones += 1;
        }
    }
    if (ones == Length(qs) or ones == 0) {
        return 0;
    }
    else {
        return 1;
    }
}
```