Выполнил студент группы А-05-19
Трофимов Илья Сергеевич
27 мая 2022 г.

Домашняя работа №3

ЗАДАНИЕ 1. Операции с числами

Реализуйте машины Тьюринга, которые позволяют выполнять следующие операции:

1. **Сложение двух унарных чисел**
   Машина Тьюринга принимает две последовательности единиц, разделённых плюсом (например, 1111+11), после выполнения алгоритма машина установит каретку на начало результрующего числа. Машина будет удалять первую единицу и заменять знак + между числами на единицу, тем самым 'склеивая' аргументы. составить таблицу переходов (использую нотацию из курса мат. логики):

состояние	1	+	ε
first	toPlus, ε, R	done, ε, R	
toPlus	R	toStart, 1, L	
toStart	L		done, R
done			H

   Код для реализации машины Тьюринга доступен по этой ссылке.

2. **Умножение унарных чисел**
   Для того, чтобы составить машину Тьюринга определим умножение следующим образом:

$$\text{mult}(0,\, b) = 0$$
$$\text{mult}(a,\, b) = b + \text{mult}(a-1,\, b)$$

   Машина Тьюринга принимает две последовательности единиц, разделённых знаком умножения (например, 1111*11), после выполнения алгоритма машина установит каретку на начало результрующего числа. Суть алгоритма заключается в последовательном уменьшении a и копировании b на каждом шаге. Составим таблицу переходов:

состояние	1	*	ε
eachA	toB, ε, R	skip, *, R	
toB	R	eachB, *, R	
nextA	L	L	eachA, 1, R
skip	R		H
eachB	sep, ε, R		nextA, ε, L
sep	add, 1, R		R
add	R		1, sepL, ε, L
sepL	L		nextB, ε, L
nextB	L		eachB, 1, R

   Код для реализации машины Тьюринга доступен по этой ссылке.

ЗАДАНИЕ 2. Операции с языками и символами

Реализуйте машины Тьюринга, которые позволяют выполнять следующие операции:

1. **Принадлежность к языку $L = \{0^n 1^n 2^n\}, n \geq 0$.**
   Машина Тьюринга будет принимать слово и в конце совей работы записывать T, если слово принадлежит языку, и F - если не принадлежит.
   Будем помечать тройки из символов 0, 1, 2 буквами a, b, c, продвигаясь по слову вперёд. Как только пометим все буквы и достигнем пустого символа, можем считать, что исходное слово принадлежит языку L, если по какой-то причине этого не удалось сделать (например, раньше чем нужно достигли конца или встретили неожиданный символ), то слово не принадлежит языку.
   Составим таблицу переходов:

состояние	0	1	2	a	b	c	ε
q_0	$q_1 a R$	$q_{end} F R$	$q_{end} F R$	$q_{end} F R$	$q_{scan} b R$	$q_{end} T R$	
q_1	R	$q_2 b R$			R		
q_2		R	$q_{back} c R$			R	
q_{back}	L	L		$q_0 a R$	L	L	
q_{scan}	$q_{end} F R$	$q_{end} F R$	$q_{end} F R$	$q_{end} F R$	R	R	$q_{end} T R$
q_{end}							L

   Код для реализации машины Тьюринга доступен по этой ссылке.

2. **Проверка соблюдения правильности скобок в строке**
   Пусть машина Тьюринга принимает последовательность скобок и в конце своей работы устанавливает T, если последовательность правильная, F - если неправильная. Будем пользоваться следующим алгоритмом:

   - Движемся вправо до появления некоторой закрывающей скобки, пусть), заменяем её буквой A (другие скобки заменяем другими буквами).
   - Теперь возвращаемся назад, пока не найдём соответсвующую открывающую скобку, пропуская все помеченные скобки, если найдём открывающую скобку другого типа или пустой символ (т.е. вернёмся в начало), то слово неправильное.
   - Нужную открывающую скобку тоже заменяем на A и повторяем этот процесс.

   Если, выполняя данный процесс, достигли пустого символа (в данном случае конца слова), то слово правильное.
   Составим таблицу переходов:

состояние	(	⟨	}	)	⟩	}	A	B	C	ε
q_{right}	R	R	R	$q_a A R$	$q_b B R$	$q_c C R$	R	R	R	$q_{end} T R$
q_A	$q_{right} A R$	$q_{end} F R$	$q_{end} F R$				L	L	L	$q_{end} F R$
q_B	$q_{end} F R$	$q_{right} B R$	$q_{end} F R$				L	L	L	$q_{end} F R$
q_C	$q_{end} F R$	$q_{end} F R$	$q_{right} C R$				L	L	L	$q_{end} F R$
q_{end}							L	L	L	L

   Код для реализации машины Тьюринга доступен по этой ссылке.

3. **Поиск минимальной строки**
   Машина Тьюринга принимает последовательность строк из 0 и 1, разделённых тире (т.к. в turingmachine.io пробелом отмечается пустой символ) и в конце своей работы устаналивает каретку на начало наименьшей строки.
   Для нахождения минимальной строки будем помечать по одному символу каждой строки до тех пор пока не дойдём до разделителя-тире. После этого счтиаем, что минимальная строка найдена (она позади). Зачищаем ленту от ненужных строк и возвращаем каретку. Готово.
   Составим таблицу переходов:

(inputenc) Package inputenc Error: Invalid UTF-8 byte "81See the inputenc package documentation for explanation.The document does not appear to be in UTF-8 encoding.Try adding as the first line of the fileor specify an encoding such as [latin1]inputencin the document preamble.Alternatively, save the file in UTF-8 using your editor or another toolc

состояние	0	1	a	b	-	ε
mark	next, a, R	next, b, R	R	R	clear, R	reset, L
next	R	R			mark, R	back, L
back	L	L	L	L	L	mark, R
clear	ε, R	ε, R	ε, R	ε, R	ε, R	return, L
return					ε, R	reset, L
reset	0, L	1, L			done, R	done, R
done	H	H				H

Код для реализации машины Тьюринга доступен по этой ссылке.

ЗАДАНИЕ 3. Квантовые вычисления

Все функции для решений следующих заданий расположены в репозитории по этой ссылке.

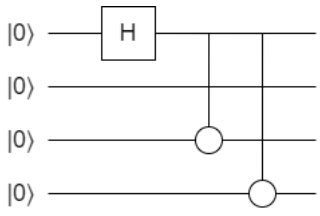
1. **Генерация суперпозиций.**
   Дано N кубитов ($1 \leq N \leq 8$) в нулевом состоянии $0 \dots 0$. Также дана некоторая последовательность битов, которое задаёт ненулевое базисное состояние размера N. Задача получить суперпозицию нулевого состояния и заданного.
   Решение на Q#:

```
operation Set (desired: Result, q1: Qubit) : () {
    body {
        let current = M(q1);
        if (desired != current) {
            X(q1);
        }
    }
}

operation Super(qs: Qubit[], bits: Bool[]) : () {
    body {
        // Предварительно зануляем кубиты
        for (q in qs) {
            Set(Zero, q);
        }

        H(qs[0]);
        for (i in 1..Length(qs) - 1) {
            if (bits[i]) {
                CNOT(qs[0], qs[i]);
            }
        }
    }
}
```

   Схема для случая $|0000\rangle$, $|1011\rangle$:



2. **Различие состояний 1.**
Дано N кубитов ($1 \le N \le 8$), которые могут быть в одном из двух состояний:

$$GHZ = \frac{1}{\sqrt{2}}(0\ldots0 + 1\ldots1)$$

$$W = \frac{1}{\sqrt{N}}(10\ldots00 + 01\ldots00 + \cdots + 00\ldots01)$$

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

```
operation Determine1(qs: Qubit[]) : Int {
    body {
        mutable ones = 0;
        for (q in qs) {
            if (M(q) == One) {
                set ones = ones + 1;
            }
        }
        if (ones == 1) {
            return 1;
        }
        else {
            return 0;
        }
    }
}
```

3. **Различие состояний 2.**
Дано 2 кубита, которые могут быть в одном из двух состояний:

$$S_0 = \frac{1}{2}(00 + 01 + 10 + 11),\ S_1 = \frac{1}{2}(00 - 01 + 10 - 11),$$

$$S_2 = \frac{1}{2}(00 + 01 - 10 - 11),\ S_3 = \frac{1}{2}(00 - 01 - 10 + 11$$

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

```
operation Determine2(q1: Qubit, q2: Qubit) : Int {
    if (M(q1) == Zero) {
        if (M(q2) == Zero) { return 0; }
        else { return 1; }
    }
    else {
        if (M(q2) == Zero) { return 2; }
        else { return 3; }
    }
}
```

4. **Написание оракула.**
Требуется реализовать квантовый оракул на N кубитах ($1 \le N \le 8$), который реализует следующую функцию: $f(\boldsymbol{x}) = (\boldsymbol{bx}) \mod 2$, где $\boldsymbol{b} \in \{0,1\}^N$ вектор битов и $\boldsymbol{x}$ вектор кубитов. Выход функции записать в кубит $\boldsymbol{y}$. Количество кубитов N ($1 \le N \le 8$).
Решение на Q#:

```
operation Oracle(x : Qubit[], y : Qubit, b : Int[]) : () {
    body {
        for (i in 0..Length(x) - 1) {
            if (b[i] == 1) {
                CNOT(x[i], y);
            }
            else {
                Set(Zero, y);
            }
        }
    }
}
```

Все функции для решений следующих заданий расположены в репозитории по этой ссылке.