Степанов Степан А-05-19

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, 111+111), 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:

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

   Код для реализации находится в файле 1_1.yaml

2. **Умножение унарных чисел**
   Для того, чтобы составить машину Тьюринга определим умножение следующим образом:
   mult(0, b) = 0
   mult(a, b) = b + 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 находится в файле 1_2.yaml

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_1aR	$q_{end}FR$	$q_{end}FR$	$q_{end}FR$	$q_{scan}bR$	$q_{end}TR$	
q_1	R	q_2bR			R		
q_2		R	$q_{back}cR$			R	
q_{back}	L	L		q_0aR	L	L	
q_{scan}	$q_{end}FR$	$q_{end}FR$	$q_{end}FR$	$q_{end}FR$	R	R	$q_{end}TR$
q_{end}							L

Код для реализации находится в файле 2_1.yaml

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:

   (a) Движемся вправо до появления некоторой закрывающей скобки, пусть), заменяем её буквой A (другие скобки заменяем другими буквами).

   (b) 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.

   (c) Нужную открывающую скобку тоже заменяем на A и повторяем этот процесс.

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

состояние	(	$\langle$	}	)	$\rangle$	}	A	B	C	ε
q_{right}	R	R	R	q_aAR	q_bBR	q_cCR	R	R	R	$q_{end}TR$
q_A	$q_{right}AR$	$q_{end}FR$	$q_{end}FR$				L	L	L	$q_{end}FR$
q_B	$q_{end}FR$	$q_{right}BR$	$q_{end}FR$				L	L	L	$q_{end}FR$
q_C	$q_{end}FR$	$q_{end}FR$	$q_{right}CR$				L	L	L	$q_{end}FR$
q_{end}							L	L	L	L

Код для реализации находится в файле 2_2.yaml

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.
Составим таблицу переходов:

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 находится в файле 2_3.yaml

ЗАДАНИЕ 3.Квантовые вычисления

1. **Генерация суперпозиций 1**
   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.

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

   То есть требуется реализовать операцию, которая принимает на вход:

   (a) Массив кубитов q_s

   (b) Массив битов $bits$ описывающих некоторое состояние ψ. Это массив имеет тот же самый размер, что и qs. Первый элемент этого массива равен 1.

   Так как первые кубиты векторов различны, то применяем оператор Адамара к первому кубиту

   Далее если bits[i] равно 1, то запутываем i-ый кубит (оператор CNOT)

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

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

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

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

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

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

```
                }
                if (countOnes == Length(qs) or countOnes == 0)
                {
                    return 0;
                }
                return 1;
            }
        }
    }
```