Quantum Algorithms
Homework 1

Patrick Canny

Due: 2019-01-29

1 Book Problems

1. Exercise 1.3
   Prove that there is no algorithm that determines, for given Turing machine and input string, whether the machine terminates at that input or not.

SOLUTION

Claim 1.1. *The above assertion holds, i.e. there is no algorithm to determine whether or not a machine halts for a given input.*

Proof of claim. Towards a contradiction, we assume that such an algorithm does exist. This algorithm would be defined as $f(\mathcal{M}, i)$ where $\mathcal{M}$ is a Turing Machine and i is the given input string. f will then return "1" if the machine halts on input i and "0" otherwise.

For f to return, it must simulate the execution of $\mathcal{M}$ on input i and determine the result. However, if $\mathcal{M}$ does not halt, the execution of f will not terminate. Since f must terminate and return an answer, this is a contradiction of the assumption above, proving that no such algorithm exists.

•

2 Additional Problems

1. Write a Turing machine that takes as input a binary number a (a string in "0" and "1") and outputs $a - 1$ (in binary). Be careful to specify the alphabet and all instructions.

SOLUTION

Here is the specification of a Turing machine that decrements a binary number:

$$\begin{aligned}
S =& \{0, 1, \_\} \\
A =& \{0, 1\} \\
Q =& \{q_0, q_1, q_2\} \\
\delta(q_0, 1) =& (q_0, 1, 1) \quad \delta(q_0, 0) = (q_0, 0, 1) \quad \delta(q_0, \_) = (q_1, \_, -1) \\
\delta(q_1, 0) =& (q_1, 1, -1) \quad \delta(q_1, 1) = (q_2, 0, -1) \\
\delta(q_2, 0) =& (q_2, 0, -1) \quad \delta(q_2, 1) = (q_2, 1, -1)
\end{aligned}$$

This machine is built with a few assumptions in mind:

- If the machine receives the number 0, it will return the maximum number for a binary number the length of the input. I.e., the machine operates in a number system modulo 2^n where n is the length of the input.
- The machine only works on positive binary numbers.
- Assume that initially the head of the Turing machine is pointed at the first character of the input string.

2. Let $\mathcal{T} = \Big\{ \mathcal{M} \mid \mathcal{M} \text{ is a Turing machine} \Big\}$.

   Find a function $f : \mathcal{T} \to \mathbb{N}$ such that if $\mathcal{N}$ and $\mathcal{M}$ are Turing machines and $f(\mathcal{N}) = f(\mathcal{M})$ then $\mathcal{N} = \mathcal{M}$ (that is, $\mathcal{N}$ and $\mathcal{M}$ have the same alphabet, same transition function, etc.).

SOLUTION

This problem seems a little tricky, since I don't think it is enough to check if two Turing machines accept the same language. You could have a number of Turing machines that accept the same language, but do not have the same transition function.

In order to solve this problem, we will need to find an f where f is a bijection(f is "one-to-one" and "onto"). This is more straightforward to do, since we can use the fact that $\mathcal{T}$ is countable.

f will convert the input Turing machine into a binary representation. This is possible, since all Turing machines can be encoded as binary strings. Then, f would take this string encoding, and return the natural number associated with that encoding. This is assuming that all of the Turing machines have been previously enumerated in this way.

To show then that f is a bijection, we must establish a diagonalizable matrix.

To do this is straightforward: establish the columns as $f(\mathcal{M})$ and the rows as the numbers of $\mathbb{N}$. Then, mark a 1 in the row corresponding to the output of f. This matrix will be a diagoinal, square matrix, since the Turing machines were assumed to be enumerated with $\mathbb{N}$. The matrix will then look as follows:

$$I = \begin{bmatrix} 1 & & \\ & \ddots & \\ & & 1 \end{bmatrix}$$

Since this matrix is diagonal, it shows that f is a bijection, meaning that it is a valid function for this problem.

Then, f can be defined explicity as follows:

$f(\mathcal{M})$:

- convert $\mathcal{M}$ into it's unique binary encoding, call it b.
- convert b into its proper numerical representation in $\mathbb{N}$, call it n.
- return n