MA3219 HW2

Tan Yee Jian (A0190190L)

July 13, 2021

Question 1

Solution. We use the diagonalization argument. Suppose there is a universal $\alpha : \mathbb{N}^2 \to \mathbb{N}$, such that for any total recursive function f, we must have some $n \in \mathbb{N}$ such that

$$\forall x \in \mathbb{N}, f(x) = \alpha(n, x).$$

Then let $g : \mathbb{N} \to \mathbb{N}$ be defined by

$$g(x) = \alpha(x, x) + 1.$$

It is easy to see that g is total recursive since α is total recursive. Suppose α is universal, then $g(x) = \alpha(n, x)$ for some $n \in \mathbb{N}$ by the definition of universal function. Then feeding n as the input to g,

$$g(n) = \alpha(n, n) \neq \alpha(n, n) + 1,$$

a contradiction to the definition of g. $\square$

Question 2

Solution

Our solution has the following idea:

1. The input has form 01^x01^y.
2. We replace the 1 at the beginning of both sections pair-by-pair.
3. If any of them run out but the other have not, then output 01, which represents `false`.
4. Otherwise, output 011, which represents `true`.

```
; find the first 1 to the right
0 * * R first-one

first-one 1 0 R next-one-search
first-one 0 * R first-one
first-one _ * L end-right

next-one-search 0 * R next-one-write
next-one-search 1 * R next-one-search ; hit a 0 first
next-one-search _ * L end-wrong

next-one-write 0 * R next-one-write
next-one-write 1 0 * back
next-one-write _ * L end-wrong

back _ * R first-one
back * * L back
```

```
end-right _ * R end-right-1
end-right * * L end-right

end-right-1 * 0 R end-right-2
end-right-2 * 1 R end-right-3
end-right-3 * 1 R end-right-4
end-right-4 * _ * halt-right

end-wrong _ * R end-wrong-1
end-wrong * * L end-wrong

end-wrong-1 * 0 R end-wrong-2
end-wrong-2 * 1 R end-wrong-3
end-wrong-3 * _ * halt-wrong
```

Question 3

Solution

We can encode the (ordered) alphabet using the binary digits, representing a_n as n in binary. For the tape, treat every $\lceil \log_2(n) \rceil$ cells as a unit, and operate on it.

It is possible to use $\{\square, 1\}$ as alphabet. We use 1^x to represent a_x, and use spaces to delimit. Two or more consecutive blank cells indicate the either end of the tape.

But I read online that Claude Shannon proved that one-symbol Turing machines are not universal.

Question 4

Part 4(i)

We assume knowledge on calling another Turing Machine as a subprogram - simply changing the initial and final states of the subprogram so it fits into the current program.

I. Primitive recursion: let the Turing Machines representing $g : \mathbb{N}^m \to \mathbb{N}, h : \mathbb{N}^{m+2} \to \mathbb{N}$ be M_g, M_h respectively. To implement $f(\overrightarrow{x}, y)$, we mark the number of times to the left of the input and build the result systematically:

   (a) First mark 1^y at the left of the input.

   (b) Calculate $f(\overrightarrow{x}, 0)$ by calling M_g on $\overrightarrow{x}$.

   (c) If to the left of the input is already cleared, return the result. Otherwise, remove a mark from the left of the input, and calculate the next iteration using M_h.

   (d) If the left to the input is cleared, return the result. Otherwise go to (c).

II. Minimization: suppose the partial recursive function $f : \mathbb{N}^m \to \mathbb{N}$ is defined as a minimization of some Turing Computable $g : \mathbb{N}^{m+1} \to \mathbb{N}$. To code a Turing Machine for f, we just need to literally search from $0, 1, 2, \ldots$, and at the end of each step increment a "counter". If the result is found, just return the "counter" as an output.

   (a) First let the input $01^{x_1}01^{x_2}\ldots$ be given.

   (b) Mark the counter (eg. to the left of the input) as 0.

   (c) Run M_g, the Turing machine representing g on $\overrightarrow{x}$ and counter.

   (d) If the result returned by M_g is 0, return this value of counter. Otherwise, go to (c).