Homework 7

Yutong Huang (yxh589)

Problem 1

Obviously, if $K(x)$ is computable, then `incompressible(n: integer)` will output every incompressible string in $\{0,1\}^n$. To create a short description for an incompressible string using `incompressible(n: integer)`, we can create the following procedure:

```
function description(S: string){
    string[] incompressibleList <- incompressible(|S|);
    if (S in incompressibleList):
        return S;
}
```

Proof. Assume `S` is an arbitrary incompressible string, then `S` is an incompressible string of length $|S|$. Therefore `S in incompressibleList` is true, and the procedure returns `S`;
$\therefore \forall s \in \{\textit{incompressible strings}\}$, `function description(S: string)` is a short description of s. $\square$

Problem 2

Let languages L_1, L_2 be arbitrary languages such that $L_1, L_2 \in P$. Then there must exist Turing machines M_1, M_2 that decide L_1 and L_2 in polynomial time respectively. Assume runtimes for M_1, M_2 are t_1, t_2 respectively.

Union: Consider language $L_1 \cup L_2$ and the following Turing machine:

Algorithm 1: $M_3(w)$

Run M_1 on w
if M_1 *accepts* **then**
  Accept and *halt*
Run M_2 on w
if M_2 *accepts* **then**
  Accept and *halt*
Reject

Then M_3 decides $L_1 \cup L_2$ in polynomial time.

Proof. Case 1: $w \in L_1 \cup L_2$
Case 1.1: $w \in L_1 \implies M_1 \textit{ accepts} \implies M_3$ accepts and halt; runtime is $t_1 + c, c$ is constant.
Case 1.2: $w \notin L_1 \implies M_1 \textit{ rejects} \implies M_2 \textit{ accepts} \implies M_3$ accepts and halt; runtime is $t_1 + t_2 + c, c$ is constant.
Case 2: $w \notin L_1 \cup L_2 \implies M_1 \textit{ rejects} \wedge M_2 \textit{ rejects} \implies M_3$ rejects; runtime is $t_1 + t_2 + c, c$.
$\therefore M_3$ correctly decides $L_1 \cup L_2$ in polynomial time. $\square$

$\therefore L_1 \cup L_2 \in P$.

Concatenation: Consider language $L_3 = \{\langle ab\rangle | a \in L_1 \wedge b \in L_2\}$ and the following Turing machine:

Algorithm 2: $M_4(w)$

Initialize M_1 with w on tape and head at the start
while M_1 *is not in an accept state* **do**
  Step through M_1 and copy its head movement to M_4
  Reject if current symbol is null
Initialize M_2 with w on tape and the head position of M_4
while M_2 *is not in an accept state* **do**
  Step through M_2 and copy its head movement to M_4
  Reject if current symbol is null
Accept

This machine correctly decides L_3 in polynomial time.

Proof. Case 1: $w \in L_3 \implies$ both loops would run without rejecting when M_1, M_2 find their corresponding substring and M_4 would accept; runtime is $t_1 + t_2 + c, c$ is constance;
Case 2: $w \notin L_3$
Case 2.1: w does not contain any substring belonging to $L_1 \implies M_4$ rejects while simulating M_1; runtime is $t_1 + c$;
Case 2.2: w contains a substring that belongs to L_1 but no substring belonging to $L_2 \implies M_4$ rejects whilw simulating M_2; runtime is $t_1 + t_2 + c$;
$\therefore M_4$ correctly decides L_3 in polynomial time.

□

$\therefore L_3 \in P$.

Complement: Consider the language $\overline{L_1}$ and the following Turing machine: This Turing machine correctly decides $\overline{L_1}$ in

Algorithm 3: $M_5(w)$

Run M_1 on w
if M_1 *accepts* **then**
  Reject and halt
else
  Accept and halt

polynomial time.

Proof. Case 1: $w \in \overline{L_1} \implies w \notin Ł_1 \implies M_1$ rejects $\implies M_5$ accepts; runtime is $t1 + c$;
Case 2: $w \notin \overline{L_1} \implies w \in Ł_1 \implies M_1$ accepts $\implies M_5$ rejects; runtime is $t1 + c$;
$\therefore M_5$ correctly decides $\overline{L_1}$ in polynomial time.

□

$\therefore \overline{L_1} \in P$.

Problem 3

Assume the size of t is $n, n \geq 1$; then t can be expressed in the form:

$$t = b_0 2^0 + b_1 2^1 + b_2 2^2 + \cdots + b_{n-1} 2^{n-1}$$

where $b_i \in 0, 1$. Then $q^t = q^{b_0 2^0 + b_1 2^1 + b_2 2^2 + \cdots + b_{n-1} 2^{n-1}}$.
Therefore, to compute q^t, we just need to compute each q^{2^i}.
To compute q^{2^i}, we first compute q^2 by applying q on q, which takes $O(1)$ time.

Then, each q^{2^i} can be computed in $O(1)$ time by applying $q^{2^{i-1}}$ on itself.
Therefore it takes $O(n)$ times to compute q^t.
Therefore $PERM - POWER \in P$