

## 1.

We have to show that  $\Sigma_1^0$ ,  $\Pi_1^0$  and  $\Delta_1^0$  are each closed under computable many-one reductions. A class  $\mathcal{C}$  is closed under many-one reductions if, for every pair of languages  $A, B$ , if  $B \in \mathcal{C}$  and  $A \leq_m B$ , then  $A \in \mathcal{C}$ .

$\Sigma_1^0$  is the class of computably enumerable languages. Let the language  $B \in \Sigma_1^0$ , and let the language  $A \leq_m B$ . This is equivalent to saying that there is a total computable function  $f : \Sigma^* \rightarrow \Sigma^*$  such that for every string  $x$ , we have  $x \in A$  if and only if  $f(x) \in B$ . Since  $B$  is computably enumerable, there is a Turing Machine  $M_B$  which accepts and halts on all inputs  $y$  which are in  $B$ , and may reject or run forever on other inputs.

We can construct an accepting Turing Machine for  $A$  as follows.

---

**Algorithm 1** The accepting machine for  $A$

---

```
1: Input  $x$ 
2:  $y \leftarrow f(x)$ 
3: if  $M_B$  accepts  $y$  then
4:   Accept  $x$ 
5: else
6:   Reject  $x$ 
7: end if
```

---

Since  $f$  is a computable function, the above algorithm describes an accepting Turing Machine. Also,

$$\begin{aligned} M_A \text{ accepts } x &\Leftrightarrow M_B \text{ accepts } f(x) \\ &\Leftrightarrow f(x) \in B \\ &\Leftrightarrow x \in A, \end{aligned}$$

so  $M_A$  accepts the language  $A$ , and therefore  $A \in \Sigma_1^0$ .

Now, we consider  $D \in \Pi_1^0$ , and  $C \leq_m D$ . If  $C \leq_m D$ , then there is a total computable function  $g : \Sigma^* \rightarrow \Sigma^*$  such that for all strings  $x$ ,  $x \in C$  if and only if  $g(x) \in D$ . This is equivalent to saying, for all strings  $x$ ,  $x \in \Sigma^* - C$  if and only if  $g(x) \in \Sigma^* - D$ , so we have that  $\Sigma^* - C \leq_m \Sigma^* - D$ .

If  $D \in \Pi_1^0$ , then  $\Sigma^* - D \in \Sigma_1^0$ . Since  $\Sigma_1^0$  is closed under computable many-one reductions, it follows that  $\Sigma^* - C \in \Sigma_1^0$ . Hence,  $C \in \Pi_1^0$ .

This proves the closure for  $\Pi_1^0$ .

Let  $F \in \Delta_1^0$ , and  $E \leq_m F$ . Since  $F \in \Delta_1^0 = \Sigma_1^0 \cap \Pi_1^0$ , by the closure under computable many-one reductions of  $\Sigma_1^0$  and  $\Pi_1^0$ , we can conclude that  $E \in \Sigma_1^0 \cap \Pi_1^0 = \Delta_1^0$ .

## 2.

We know that  $W$  and  $K$  are complete for  $\Sigma_1^0$ . i.e., Any language  $A \in \Sigma_1^0$  satisfies the properties  $A \leq_m W$  and  $A \leq_m K$ .

This implies, as in Question 1,  $\Sigma^* - A \leq_m W^c$ , and  $\Sigma^* - A \leq_m K^c$ . So any language in  $\Pi_1^0$  is computably many-one reducible to  $W^c$  and to  $K^c$ .

Also,  $W \in \Sigma_1^0$  is true if and only if  $W^c \in \Pi_1^0$ , and  $K \in \Sigma_1^0$  if and only if  $K^c \in \Pi_1^0$ , so these facts imply that both  $W^c$  and  $K^c$  are complete for  $\Pi_1^0$ .

### 3.

The following is a computable partial function with domain  $A$ , so  $A$  is acceptable.

$$\phi'(s_e) = \begin{cases} 1 & \text{if } \phi_e(0) = 0 \\ \uparrow & \text{otherwise .} \end{cases}$$

Analogously, the following is a computable partial function with domain  $B$ , making  $B$  acceptable.

$$\phi''(s_e) = \begin{cases} 1 & \text{if } \phi_e(0) = 1 \\ \uparrow & \text{otherwise .} \end{cases}$$

### 4.

$A = \{e \mid \phi_e(0) = 0\}$ , and  $B = \{e \mid \phi_e(0) = 1\}$ .

Let  $D$  be a decidable set with witness machine  $M_D$ , which satisfies the following properties:

1.  $D \supseteq A$ .
2. There is a deciding Turing Machine  $M_D$  for  $D$  which halts on every input.

We prove that  $D$  has to intersect  $B$ , which implies that  $D$  cannot be a solution to the Promise Problem  $(A, B)$ .

$f : \Sigma^* \rightarrow \Sigma^*$  a total computable function is defined as follows. For every string  $s_e$ ,  $f(s_e)$  is a Gödel number of a machine computing the following function:

$$\phi_{\text{index}(f(s_e))}(x) = \begin{cases} 1 & \text{if } s_e \in D \\ 0 & \text{if } s_e \notin D \end{cases}$$

This function  $\phi_{\text{index}(s_n)}$  is total computable since  $D$  is decidable.

Since  $f$  is total computable, by the recursion theorem, there is an  $n$  such that  $\phi_{\text{index}(f(s_n))} = \phi_n$ .

$\phi_{\text{index}(f(s_n))}(0) = 1 \equiv s_n \in D$ . Since  $\phi_{\text{index}(f(s_n))}(0) = \phi_n(0)$ , we have  $\phi_n(0) = 1 \equiv s_n \in D$ . But by the definition of  $B$ ,  $s_n \in B$ . Thus no decidable  $D \supseteq A$  can be a solution to the promise problem  $(A, B)$ .

## 5.

Recall that  $K$  the diagonal halting problem is c.e. but not co-c.e. We prove that  $K \leq_m EQ$  and  $K^c \leq_m EQ$ . First, we convince ourselves that this proves that  $EQ$  is neither c.e. nor co-c.e.

If  $K \leq_m EQ$  and  $EQ$  is co-c.e., then by Q1,  $K$  would be co-c.e., which is not true. Also,  $K^c \leq_m EQ$  implies that  $EQ$  cannot be c.e.

We produce the computable many-one reduction  $f : \Sigma^* \rightarrow \Sigma^*$  as follows.

$$\phi_{\text{index}(f(e))} = \begin{cases} \phi_e(x) & \text{if } x \neq s_e \\ 0 & \text{otherwise.} \end{cases}$$

It is now easy to verify that  $f$  is total computable. Also,  $e \in K \Leftrightarrow \langle e, f(e) \rangle \in EQ$ . This follows, because,  $\phi_{\text{index}(f(e))}$  is equal to  $\phi_e$  on all arguments except  $s_e$  – therefore, these two functions are equal exactly when  $\phi_e(s_e) = 0 \Leftrightarrow e \in K$ .

Now, we consider  $g : \Sigma^* \rightarrow \Sigma^*$  defined as:

$$\phi_{\text{index}(g(e))} = \begin{cases} \phi_e(x) & \text{if } x \neq s_e \\ \uparrow & \text{otherwise.} \end{cases}$$

As above,  $g$  is a total computable function. Also, we can show that  $e \in K^c \Leftrightarrow \langle e, g(e) \rangle \in EQ$ . This follows, because,  $\phi_{\text{index}(g(e))}$  is equal to  $\phi_e$  on all arguments except  $s_e$  – therefore, these two functions are equal exactly when  $\phi_e(s_e) = \uparrow \Leftrightarrow e \in K^c$ .

## 6.

We have to show that a partial function  $f : \Sigma^* \dashrightarrow \Sigma^*$  is computable if and only if its graph  $\{\langle x, f(x) \rangle \mid x \in \text{dom}(f)\}$  is computably enumerable.

Suppose the partial function  $f$  is computable. Then there is a Turing Machine  $M_f$  such that if  $x$  is in the domain of  $f$ , then  $M_f(x)$  halts outputting the value  $f(x)$ , otherwise  $M_f(x)$  runs forever. We prove that in this case, the graph of  $f$  is computably enumerable by building a Turing Machine that accepts exactly the paired strings  $\langle x, f(x) \rangle$  in the graph of  $f$ .

Given a paired string  $\langle x, y \rangle$ , the machine runs  $M_f(x)$ , and if the output of  $M_f(x)$  is equal to  $y$ , then it accepts  $\langle x, y \rangle$  and halts. This machine accepts the graph of  $f$ , because, for all paired strings  $\langle x, y \rangle$ , it halts in an accepting state if and only if  $M_f(x)$  halts with output  $y$ , in which case  $y = f(x)$ .

Conversely, let the graph of  $f$ ,  $\text{Graph}(f)$ , be computably enumerable. Then  $\text{Graph}(f)$  is either empty, or there is a total computable function  $g : N \rightarrow \Sigma^*$  such that  $\text{range}(g) = \text{Graph}(f)$ . If  $\text{Graph}(f)$  is empty, then it follows that the domain of  $f$  is empty, hence  $f$  is partial computable. Now, assume that the domain of  $f$  is non-empty.

We can build a Turing Machine  $M$  which computes the partial function  $M$  as follows. On input  $x$ , it runs  $g$  on inputs  $i = 0, 1, \dots$  until it finds a  $g(i)$  such that  $\pi_1(g(i)) = x$ . Then,  $M$  outputs  $\pi_2(g(i))$  and halts. We can see that the machine's computation is correct as follows. The function  $g$  enumerates exactly the pairs  $\langle y, f(y) \rangle$  in the graph of  $f$ . Hence, if  $x$  is in the domain of  $f$ , then eventually there is an  $i$  at which  $g(i) = \langle x, f(x) \rangle$ . At this  $i$ , the machine will output  $\pi_2(x, f(x))$ , which is  $f(x)$ , and halt.

Hence,  $f$  is partial computable.

## 7

Since  $A$  and  $B$  are disjoint sets, every string is in exactly one of the three sets:  $A, B, (A \cup B)^c$ . Since  $A$  and  $B$  are c.e., there are Turing Machines  $M_A$  and  $M_B$  which accept the languages  $A$  and  $B$ , respectively. We can show that  $A \leq_T A \cup B$  by building the following Oracle Turing Machine  $M^{A \cup B}$ :

---

**Algorithm 2** The oracle machine  $M^{A \cup B}$

---

```

1: Input x
2: if  $A \cup B(x) = 0$  then
3:   Reject  $x$ , halt.
4: else
5:    $i = 0$ 
6:   while true do
7:     if  $M_A(x)$  accepts in at most  $i$  steps then
8:       Accept  $x$ , halt
9:     else if  $M_B(x)$  accepts in at most  $i$  steps then
10:      Reject  $x$ , halt
11:    end if
12:     $i = i + 1$ 
13:  end while
14: end if

```

---

If  $x \in (A \cup B)^c$ ,  $x$  will be rejected in line 3. If  $x \in B$ , then  $x \notin A$ , so the machine  $M_A$  will never accept  $x$  and the machine  $M_B$  will eventually accept  $x$ . Thus  $x$  will be rejected in line 10. If  $x$  is in  $A$ , then machine  $M_B$  will not accept  $x$ , and  $M_A$  will eventually accept  $x$ , so  $M^{A \cup B}$  will accept  $x$ . Thus  $M^{A \cup B}$  correctly decides  $A$ .

In a similar way, we can show that  $B \leq_T A \cup B$ , by constructing an oracle Turing Machine interchanging the decisions in lines 8 and 10.

## 8.

An oracle  $B$  is a set which can be taken as given. Every set is equivalently described by a characteristic function  $\chi_B : N \rightarrow \{0, 1\}$ , described as follows:  $\chi_B(e) = 1$  if  $s_e \in B$ , and  $\chi_B(e) = 0$  if  $s_e \notin B$ . Intuitively, the characteristic function is a “bit-map” of the set  $B$ , such that the bit at position  $i$  is 1 if  $s_e \in B$ , and 0 otherwise.

The intuition is that the oracle  $B$  is represented by an infinite “bit-map” which has all the information

of the halting problem, but hidden at positions uncomputably far apart from each other. Essentially the only way to get information about the membership of any particular string  $s_e$  in  $K$  is to start querying  $B$  from the first bit until we get replies to the membership queries of all strings  $s_0, s_1, \dots, s_e$ .

Let the oracle  $B$  be described as follows. For every string  $s_e$ , if  $s_e \in K$ , then the bit at  $e + G(e)$  is set to 1, and the bit at position  $e + G(e) + 1$  is set to 0. For every string  $s_e \notin K$ , the bit at position  $e + G(e)$  is set to 0, and the bit at position  $e + G(e) + 1$  is set to 1. All other bits of  $B$  are zero. (The bit at position  $e + G(e)$  serves as a “signal bit” that indicates that the *next* bit represents whether  $s_e \in K$  or not.)

We now make this intuition formal.

The set  $B$  is

$$\{x0^{G(\text{index}(x))} \mid x \in \Sigma^*\} \cup \{s_e0^{G(e)}1 \mid s_e \in K\}.$$

Then  $K \leq_T B$  by the oracle Turing Machine  $M^B$  given in Algorithm 3. The algorithm always halts, because for every  $s_e$ , it will eventually find its membership information from  $B$  at position  $e + G(e) + 1$ . Also, when it halts, it would correctly decide  $K$ : when the loop in lines 4–8 breaks,  $j = e + G(e) + 1$ , so  $B(j) = 1$  if and only if  $s_e \in K$ .

We sketch a proof showing that there can be no computable time-bounded Turing Reduction from  $K$  to  $B$ . Let  $t : \mathbb{N} \rightarrow \mathbb{N}$  be a computable function that upper bounds the running time of some oracle Turing Machine  $N^B$  which decides  $K$ . This would imply that for all large  $e$ , whether  $s_e \in K$  is decidable by  $N^B$  in time upper bounded by  $t(|s_e|)$ . With the cost model which takes one step to write one bit on the query tape, we can say that  $N^B$  can query at most  $t(|s_e|)$  initial bits of  $B$  to decide the membership of  $s_e$ . But the information of the membership of  $s_e$  itself is not available at that position in  $B$ .

Let  $e'$  be the smallest number such that for any string  $s_k$  which follows  $s_{e'}$  in the standard enumeration of binary strings, the membership of  $s_k$  can be decided by  $N^B$  in at most  $t(|s_k|)$  steps. Then there is a computable function  $f(s_0, s_1, \dots, s_{e'-1})$  which decides the membership of  $s_{e'} \in K$ . By strong induction, we can see that  $f$  has the property that, for all strings beyond  $s_{e'}$  in the standard enumeration of strings, given the membership information of strings from  $s_0$  to  $s_{e'-1}$ , it can compute their membership information. This means that for all large  $e$ , the membership of  $s_e$  in  $K$  is decidable in  $t(|s_e|)$  steps given information about the membership finitely many initial strings in  $K$ .

But this is a contradiction, since the membership of  $s_0$  to  $s_{e'-1}$  is only a finite amount of information, and the existence of such a function  $f$  would imply that  $K$  can be decided by a Turing Machine.

---

**Algorithm 3** The turing reduction from  $K$  to  $B$ 

---

```
1: Input  $x$ 
2:  $n = \text{index}(x)$ 
3:  $i, j = 0, \text{isMember} = B(G(0) + 1)$  { $G(0)$  is a hard-coded constant.}
4: while  $i < n$  do
5:   if  $B(j) = 1$  then {Signal Bit found}
6:      $i = i + 1$ 
7:      $j = j + 1$ 
8:      $\text{isMember} = B(j)$ 
9:   end if
    $j = j + 1$ 
10: end while
11: if  $\text{isMember} = 1$  then
12:   Accept  $x$ .
13: end if
14: Reject  $x$ .
```

---

*“Beware of bugs in the above code. I have only proved it correct; not tried it.”* - Donald E. Knuth,  
“Notes on the van Emde Boas construction of priority deques: An instructive use of recursion”.