# Exercise 1

Use diagonalisation to show that the membership problem is undecidable. That is, show that the following language is not recursive: MP def = { M#x | x ∈ L(M) }.

## Diagonalisation proof

Let’s suppose that the set MP is recursive. Thus, there must exist a Total Turing machine K such that L(K) = MP

In other words, K behaves in the following way:

* On input M#x:
  + If x belongs to L(M) then accept
  + Otherwise reject

Let Turing machines be encoded as strings over the alphabet {0,1} and have as input alphabet Σ = {0,1}.

Thus, Mx is a Turing machine whose encoding is x.

Therefore, for every Turing machine Mx, K knows whether Mx#y is in MP or not.

Let there be a Turing machine N that behaves in the following way:

* On input x:
  + Constructs Mx
  + Runs K on Mx#x
  + If k accepts then reject
  + Otherwise accept

In other words, this machine N behaves differently from every other Turing machine Mx.

Thus, there is a contradiction because K is supposed to know whether every possible Mx#y belongs to MP or not. Following this, there cannot exist any Total Turing machine K that decided MP. Consequently, MP is not recursive.

# Exercise 2

Prove that LP def = { M#x | M loops on x } is not recursively enumerable.

LP = ~HP as HP = {M#x | M halts on x}.

We have proved in some other lecture the following:

A is recursive enumerable and ~A is recursive enumerable iff A is recursive

Since HP is recursive enumerable and HP is not recursive then it follows that ~HP cannot be recursive enumerable otherwise HP would be recursive.

Thus, ~HP is not recursive enumerable.

# Exercise 3

Construct a reduction from LP = { M#x | M loops on x } to { M | L(M) = ∅ } to prove that the latter is not recursively enumerable.

We know the following:

If A <m B and A is not recursive enumerable then B is not recursive enumerable.

I need to build a reduction function σ(M#x) such that

* M#x belongs to LP then L(σ(M#x)) = ∅
* M#x does not belongs to LP then L(σ(M#x)) ≠ ∅

Thus, let σ(M#x) be the following Turing machine:

* On input y:
  + Run machine M on string x
  + If M halts on x then accept

As we can see the language of σ(M#x) is the following:

* If M halts on x then L(σ(M#x)) = Σ\*
* If M does not halt on x then L(σ(M#x)) = ∅

Thus, we have built a correct reduction from the set LP to the set { M | L(M) = ∅ }.

As all member of LP will be mapped to Turing machines belonging to { M | L(M) = ∅ }. All strings belonging to ~LP are mapped to {M | L(M) ≠ ∅}.

# Exercise 4

Show that the language A = { M | L(M) = L(a\*) } is not recursive by constructing a reduction from HP.

In other words, we need to construct a reduction function σ in the following way:

* If M#x is in HP then L(σ(M#x)) = L(a\*)
* If M#x is not in HP then L(σ(M#x)) ≠ L(a\*)

We can define σ(M#x)) in the following way:

* On input y:
  + Run M with string x
  + Checks whether y is in L(a\*) simulating its corresponding DFA.
  + If it is then accept otherwise reject

As we can see the following holds:

* If M#x is in HP then L(σ(M#x)) = L(a\*)
* If M#x is in HP then L(σ(M#x)) = ∅ which it is different from L(a\*)

Thus, we have created a correct reduction from HP to A. Therefore, A is not recursive.

# Exercise 5

Show that the language { M | M loops on some input } is not recursive enumerable.

Thus, I will create the following reduction: ~HP <m { M | M loops on some input }

The reduction must obey the following rules:

* If M#x is in ~HP then σ(M#x) is in { M | M loops on some input }
* If M#x is in HP then σ(M#x) is not in { M | M loops on some input }

In other words, the reduction function σ outputs a Turing machine.

Let σ(M#x) be defined in the following way:

* On input y:
  + Run M with input string x
  + If M halts then accept

Thus, the language accepted by σ(M#x) is the following:

* If M#x is in ~HP then L(σ(M#x)) = ∅ which means that σ(M#x) loops on all inputs
* If M#x is in HP then L(σ(M#x)) = Σ\* which means that σ(M#x)) never halts

Since, we have built a reduction from ~HP to { M | M loops on some input } then { M | M loops on some input } is not recursive enumerable because ~HP is not recursive enumerable.

# Exercise 6

1. given TMs M and N, does L(M) = L(N)? Hint: Use Question 4.
2. given TMs M and N, does L(M) ⊆ L(N)? Hint: Use Question 3.
3. given TMs M and N, is L(M) ∩ L(N) = ∅? Hint: Use Question 3

**(i)** We know from exercise 4 this: { M | L(M) = L(a\*) } is not recursive. In other words, there is no Total Turing machine that either accept or reject any input string x.

If the set {(M,N) | L(M) = L(N)} were recursive (its Total Turing Machine is K) then the previous set { M | L(M) = L(a\*) } would be recursive as well because we could build the following Total Turing machine M’ accepting { M | L(M) = L(a\*) }:

* On input x:
  + Run machine K on input (x,P) //P is a trivial Turing machine accepting L(a\*)
  + If K accept then accept, otherwise reject

As we can see then such a Turing machine K cannot exist otherwise the problem in exercise 4 would be recursive. Therefore, the set {(M,N) | L(M) = L(N)} is not recursive.

**(ii)** We know from exercise 3 the following: {M | L(M)= ∅} is not a recursive enumerable.

If the set {(M,N) | L(M) ⊆ L(N)} were recursive (its Total Turing Machine is K) then the set {M | L(M)= ∅} would be recursive as well as we could create the following Total Turing Machine M’:

* On input x:
  + Run K with input (x,P) //where P is a trivial Turing machine such that L(P) = ∅
  + If K accepts then accept, otherwise reject

As we can see then such a Turing machine K cannot exist otherwise the problem in exercise 3 would be recursive while it is not even recursive enumerable. Therefore, the set {(M,N) | L(M) ⊆ L(N)} is not recursive.

**(iii)** We know from exercise 3 the following: {M | L(M)= ∅} is not a recursive enumerable.

If the set {(M,N) | L(M) ∩ L(N) = ∅} were recursive (its Total Turing Machine is K) then the set {M | L(M)= ∅} would be recursive as well as we could construct the following Turing machine:

* On input x:
  + Run K with input (x,P) // where P is a trivial Turing machine accepting Σ\*
  + If K accepts then accept otherwise reject

As we can see then such a Turing machine K cannot exist otherwise the problem in exercise 3 would be recursive while it is not even recursive enumerable. Therefore, the set {(M,N) | L(M) ∩ L(N) = ∅ } is not recursive.