Let set A be r.e.
Let set B be r.e.

Prove \( A \cup B \) is r.e.

Since A is r.e., there exists an algorithm that will eventually print all members of A (alg A)

\[ \text{Run Alg A for 1 hr} \]

\[ \text{Run Alg B for 1 hr} \]

print the elements in common

\[ \text{Run Alg A for 2 hrs} \]

\[ \text{Run Alg B for 2 hrs} \]

print the elements in common

keeping running the algorithms in this fashion until all elements have been generated.

3) Since B is decidable it is also recursively enumerable. Proof:

\[ \text{int } n = 0 \]
\[ \text{while(true) } \]
\[ \text{if (n \in B)} \]
\[ \text{print n} \]
\[ \text{\text{Therefore B is r.e.}} \]

Since A is r.e. and B is r.e. then \( A \cup B \) is r.e. (the proof is in problem 4 this text)

Is \( A \cup B \) always decidable? **NO**, Counter example:

Let A be the halting set which is r.e.
Let B = \( \mathbb{N} \) which is decidable

\( A \cup B = \mathbb{N} \) which is not decidable

3) I will prove that a n+1-checker program doesn't exist by showing that if it existed we could build a zero-checker out of it (and we have already proven that zero-checker does not exist)

\( \text{n+1-checker} \( \text{p(n) = } \begin{cases} 1 & \text{if } \forall n \ p(n) \text{ holds and } p(n) = n+1 \\ 0 & \text{if } \forall n \ p(n) \text{ holds and } \exists n (p(n) \neq n+1) \end{cases} \)

To build a zero-checker we need an auxiliary function \( q \) which given input \( n \), its output is \( n+1 \) and also that \( p(n) = 0 \), such that \( q(n) = n+1 \iff p(n) = 0 \)

The specific auxiliary program is described as follows:

\( q(n) = p(n) + n+1 \)
\[
\text{Zero}_{\mu}(t) = \begin{cases} 
1 & \text{if } \forall n (p(n) = 0 \text{ and } p(n) \downarrow \text{ on } n \text{ before time } t) \\
0 & \text{if } \exists n (p(n) \neq 0 \text{ and } p(n) \downarrow \text{ on } n \text{ before time } t) 
\end{cases}
\]

\[
\text{zero-checker}(p) = \mathcal{T}_{\mu+1}\text{-checker}(\text{zero}_{\mu}(t))
\]

If \((q(n) = n+1)\) then

\(p(n)\) must be zero.

If \((q(n) \neq n+1)\) then

\(p(n)\) must be different than zero.

Therefore, a \(\mu+1\)-checker cannot exist or a \(\text{zero-}\text{checker}\) could be built from it, which is impossible.

\[4-5\]

Turing machine for \(\sigma\):

- \(\text{start} \# \rightarrow \text{forward}, \text{R}\)
- \(\text{forward}, 1 \rightarrow \text{forward}, \text{R}\)
- \(\text{forward}, \# \rightarrow \text{returning}, 1, \text{L}\)
- \(\text{returning}, 1 \rightarrow \text{returning}, \text{L}\)
- \(\text{returning}, \# \rightarrow \text{halt}\)

Algorithm for composition \(f(q(n))\):

1. Rename the state \(\text{halt}\) of \(q(n)\), in this case the Turing machine for \(\tau^n_q\) to \(\text{start}\).
2. Append a 2 to the name of all states of \(f(n)\), in this case the Turing machine for \(\sigma\), except the state \(\text{halt}\).
3. Run \(q(n)\) followed by \(f(n)\).

Trace of \((\tau^n_q, 1, 11)\):

- \(\#1\#1\) followed by \(\#\) as expected.
\[ S(0, b) = 0 \]
\[ S(m+1, b) = S(m) - b \]
\[ g(0, n_1) = 0 \]
\[ f(m+1, n_1) = f(m) - n_1 \rightarrow h(m, n_1, f(m, n_1)) \]

\[ \text{PR} \left( O, \text{minus}(\Pi^3_T, \Pi^3_T) \right) \]

What is the value of this function when \( b = 2 \) and \( a = 3 \)?

\[ j = 1 \quad s = -2 \]
\[ i = 2 \quad s = -1 \]
\[ i = 3 \quad s = -6 \]

\[ s = -6 \]

\[ S = 0; \]
\[ \text{while}(S > a) \]
\[ S = S - b \]

Step 1: While-loop to for-loop

\[ S = 0; \]
\[ \text{for}(i = 1; i < c; i++) \]
\[ S = S - b \]

Step 2: For-loop to for-loop

\[ S(0, b) = 0 \]
\[ S(m+1, b) = S(m) - b \]
\[ g(0, n_1) = 0 \]
\[ f(m+1, n_1) = f(m) - n_1 \rightarrow h(m, n_1, f(m, n_1)) \]

\[ \text{PR} \left( O, \text{minus}(\Pi^3_T, \Pi^3_T) \right) \]

Step 3: PR to \( \mu \)

\[ P(m, b) = Z(m, b) > a \]
\[ C = \mu m, (1 (Z(m, b) > a)) \]
\[ F(a, b) = Z(a, b) \]
What is the value of the function when b = 2 and a = -3?

Initial value: s = 0  a = -3
After 1 loop: s = -2  s > -3 ✓
After 2 loops: s = -4  -4 > -3 ✓

s = -4

8) Kolmogorov complexity is the shortest length of a program to generate the string. In python, the following code concatenates the string '01' 2015 times:

'01' * 2015

the code has 9 characters therefore the Kolmogorov complexity of the sequence is ≤ 9. It is understandable that it is so low because the Kolmogorov complexity is a measure of randomness and this sequence is not random at all, it has a very simple pattern.

9) Time 1 0 0
Time 2 0 4 0
Time 1 0 0

Misunderstood the problem

s₁₁₂₈ = 0
s₁₂₂ = s₁₂₁
s₂₁₂₈ = 0
s₂₂₂ = s₂₂₁
s₃₁₂ = 1
s₃₂₂ = s₃₂₁
s₁₁₃ = 0
s₁₂₃ = Run out of time
P: class of problems that have a polynomial-time algorithm to solve them.

NP: class of problems whose solutions can be verified in polynomial time.

NP-hard: A problem is NP-hard if every problem from the class NP can be reduced to it.

NP-complete: A problem from the class NP that is also NP-hard.

- P: the class P contains the problem of sorting a list since it can be done in $n^2$ time.

- P does not contain the propositional satisfiability problem since it cannot be solved in polynomial time.

- NP: the class NP contains the problem of sorting a list since its solution can be checked in polynomial time.

- The class NP contains the problem of propositional satisfiability since a given solution can be checked in polynomial time.

- NP-hard: The class NP-hard does not contain the problem of sorting a list unless $P = NP$.

- The class NP-hard contains the propositional satisfiability since every problem from the class NP can be reduced to it.

- NP-complete: This class does not contain the problem of sorting a list because it is not NP and NP-hard unless $P = NP$.

- This class contains the problem of propositional satisfiability since it is both NP and NP-hard.