Things to do: define time predicate, function

Please check the validity of this

1. We want to prove that a function that checks if a function *x* given an input *y* will halt does not exist, in other words:
2. Let be a function such that x and y are functions and
3. Let us assume exists
4. Let be a function such that x is a function and
5. We let such that
6. Let us assume
   1. by (3.1)
   2. by (2)
   3. This is a contradiction
8. Let us assume
   1. by (3.1)
   2. by (2)
   3. This is a contradiction
10. from (3.4) and (3.6)
11. This is a contradiction
12. does not exist.

2 pages Turing machines (&& other definitions)

2 pages of formal, 1 page of pseudo, 2 pages of case samples and practical applications.

What are the practical applications?

Modern programming languages are Turing complete. (i.e. they can be used to simulate a single taped turing machine.)

Making the halting problem undecidable / unsolvable

Thus, in order to deterministically program a function that will halt, we need to impose restrictions to the programming languages. (i.e programming disciplines)

Such an example of a type of programming language would be “total functional programming” which differs from most “weak” forms of functional programming in the following 2 conditions.

Restricted recursion which operates only on reduced forms of its arguments.

Functions must be total (as opposed to partial) -> it must have a definition for everything inside its domain.

These 2 conditions are necessary for the solvability of the halting problem in the language of programming.

What are the constraints for this???

improve on functional programming, we can restrict the turing machine to accept “ticker tapes of finite length” by restricting the form of recursion -> operates only upon