Chapter 1

Time Stone

1.1 Extracts from wiki

"The Time Stone was one of the six Infinity Stones... ...the user is able to physically control and redirect the flow of time, and can specifically select the exact area to manipulate without affecting those outside its selected range. The stone can alter targets as small as an apple or as wide in scope as the timeline of the universe... ...Due to the selective nature of the Time Stone's power, it could be used to individually alter the timeline of individual objects or events, reversing them to a previous state or sending the object forward into a future state. This occurs regardless of any potential breaches in causality... ...the Stone could send objects forward to a potential future state that does not necessarily have to occur in the current timeline."

1.2 Abilities

- Can alter an arbitrary space.
- Can reverse such a space to a previous state.
- Can send the space forward into a future potential state (that does not necessarily have to occur in the current timeline.)
- Does not cause issues in causality.

1.3 Formal Definition

A TS is a tuple T = (R, q_0, qend)

- R is a deterministic run of some automata which is assumed to halt.
- q_0 is the initial step of the run.

- qend the final step of the run.
- direction is the desired flow of time, either forward or backwards.

When activated, the TS has the capability to alter an automaton, sending it either forward in time to qend, or back in time to q_0, depending on the TS's given input. This process is assumed to take no computational power and is a constant O(1) in time complexity.

We define a new MCU complexity class, which in this case defines the number of times the TS must be activated.

1.4 P = NP when using a TS

We create a proof by doing the following:

1. Prove using a TS on an arbitrary deterministic TM that is assumed to halt will result in a time complexity of O(1).
2. Create an algorithm that simulates any non-deterministic TM on a deterministic TM.
3. Use steps 1 and 2 to show when using the TS, NP problems have a time complexity of O(1), hence P = NP.

1.4.1 Step 1: TS and deterministic TM interactions

Use the time stone to send a deterministic Turing machine forward in time to its end state such that its time complexity is reduced to O(1).

Get any arbitrary deterministic TM.

We assume the TM halts. The TM therefore has some arbitrary running time.

We get the TS components:

- R is the deterministic run of the TM.
- q_0 is the initial step of the run.
- There exists a qend step, as the algorithm is assumed to halt.
- We place the TM into its initial step. We activate the TS, altering the automaton and sending it forward in time to qend, where it either halt-accepts or halt-rejects. This takes O(1) time complexity.

The arbitrary deterministic TM, using the TS with an MCU complexity of O(1), has now halted in O(1) time.

We conclude that all deterministic TMs - given the assumption that the TM halts - have a time complexity of O(1) and MCU complexity of O(1) when using a TS.

1.4.2 Step 2: Simulating a non-deterministic TM using a deterministic TM

Have an external input dictate which path to take at any non-deterministic choice using a DFS, where eventually all paths are run. Therefore, the entire process is now deterministic in nature.

We construct a TM containing three tapes.

The Computer

The first tape, the computer, is designed to do an arbitrary run of some non-deterministic algorithm. It is assumed we know the worst-case running time of the algorithm.

At each non-deterministic step, when facing an x number of choices, each choice is labelled from 0 to x-1 such that the step can be chosen deterministically via an external input.

If the input path does not exist, the TM will halt-reject with an additional output specifying that the input path did not exist.

The tape, otherwise, can halt-reject or halt-accept as normal.

The Echo

The second tape, the echo, simply retains the initial step of the computer. It can copy its contents to the computer, such that it can return to its initial step after leaving it. The third tape shares the arbitrary tape language of the computer.

The Decider

The third tape, the decider, is designed to determine which route the computer will take given a non-deterministic step.

The input language consists of the set of natural numbers, where the ith cell determines the ith choice the first tape will make. I.e., if the first cell contains a '0', the first non-deterministic step will choose the first choice presented. If the second cell then contained a '3', the next step will choose the 4th choice presented. The decider is also the final output of the entire TM.

Usage

The decider will iterate through all the possible combinations of non-deterministic choices via the following recursive definition:

1. The decider will be initialised with x amount of 0's, where x is the maximum number of steps the algorithm will require in a run. The tape head is moved to the last 0.

2. The tape will output its content to the computer and waits for it to accept or reject.

3. The decider then reads the computer's output, such that:

   (a) If the computer outputs a halt-reject:

      i. increment the current cell by 1.
      ii. activate the echo to return the computer to its initial step.
      iii. go to step 2.

   (b) Else if the computer outputs a halt-reject, specifying the input path did not exist:

      i. mark the current cell.
      ii. set all non-empty cells to the right to 0.
      iii. move back to the marked cell, set it to 0.
      iv. move left.
      v. if a blank symbol is encountered, halt-reject, else increment the cell by 1 and move to the last cell.
      vi. activate the echo to return the computer to its initial step.

   (c) Else, if the computer outputs a halt-accept:

      i. halt-accept

The non-deterministic algorithm has now been simulated using a deterministic multi-tape TM. Multi-tape TMs area equivalent to single-tape TMs, therefore we conclude all non-deterministic algorithms that are assumed to halt can be converted to deterministic algorithms that are assumed to halt.

1.4.3 TS and non-deterministic TM interactions

As we have concluded that all non-deterministic algorithms that are assumed to halt can be converted to deterministic algorithms that are assumed to halt, we conclude that all non-deterministic TMs which are assumed to halt, when using a TS, have a time complexity of O(1) and MCU complexity of O(1).

Therefore, as all P and NP problems halt, we conclude that assuming we know the time complexity of all NP problems, given access to a TS, P = NP.