Solution to Exercise III.2.19 a) of Odifreddi

Duarte Maia

May 25, 2024

1 Problem Statement

Construct directly an example of a set which is simple but not effectively simple.

2 Solution

We perform a marker construction, in the following sense. We assume that countably many moving markers have been placed on the natural numbers, one for each number, and we will describe a simulation in which the markers are moved around. Once a natural number is left unmarked, it will never be marked again. This shall determine a c.e. set, by enumerating the numbers which are left unmarked. We will ensure that the resulting set is coinfinite by ensuring that each marker can be moved finitely many times. This leaves us with two tasks to fulfil:

P Ensure that the resulting set is simple, by making sure that for every infinite c.e. set W_e there is an element $x \in W_e$ which is left unmarked,

N Ensure that the resulting set is not effectively simple, by making sure that for every φ_e there is x such that either $\varphi_e(x) \uparrow$ or, if $\varphi_e(x) \downarrow$, W_x is a finite set, all of whose elements are marked, but $\#W_x > \varphi_e(x)$.

To ensure that all tasks P_e are fulfilled, we simply dovetail over the c.e. sets W_e and, upon finding large enough elements of W_e, we push any marker with label $> e$ that may be in their place. We do this only (at most) once for each W_e.

To ensure that N_e is fulfilled is quite tricker, and the way I've found to do it requires some strong usage of the recursion theorem. First, let P be the code for the simulation that we are programming right now. Moreover, define $x(e)$ as code for a program that does the following (this again requires usage of the recursion theorem):

```
Let x be this program
Let ν be φ_e(x)
Let P be the code for the simulation
Run P for as long as it takes for the simulation to compute ν
List the position of the first ν markers at this stage.
```

Once this is defined, we set our simulation to dovetail processes which compute $\nu = \varphi_e(x(e))$ for every $e \in \mathbb{N}$; once such a computation is complete, we place a "lock" on the first ν markers so that they can't be moved again. This would fulfill N_e, with $x = x(e)$.

I don't think that this quite works, unfortunately, because it could be the case that the requirements N_e create locks at an overwhelmingly fast rate, such that e.g. P_{73} is never fulfilled. (Also, it absolutely *couldn't* work; if it did, the union of all $W_{x(e)}$ would be an infinite c.e. set contained in the complement of the final set...) To get around this, we allow P_e to break through some locks, which will in turn mess up some of the N_i, but we ensure that P_e only breaks locks created by large enough N_i, and when it does we will make a note to re-fulfill N_i. Thus, each N_i will be broken a finite number of times (at most i), and after the last time it will be fulfilled forever.

We are now almost ready for the final construction. Define $x(e, s)$ as the following program:

```
Let x be this program
Let ν be φ_e(x)
Let P be the code for the simulation
Run P for s steps, then continue running it until it is about to fulfill N_e
List the position of the first ν + 1 markers at this stage.
```

Now, we define the simulation P as follows. Dovetail processes P_e and N_e, for $e \in \mathbb{N}$. Here is what these processes do:

N_e Compute $\nu = \varphi_e(x(e, 0))$. Once this computation halts, immediately place "locks" labeled e on the first $\nu + 1$ markers and halt this process.

P_e Enumerate elements from W_e, stopping when you find some $x \in W_e$ in an unmarked position, or in a position with an unlocked marker, or in position with a marker all of whose locks have labels $i > e$.

- If x is in an unmarked position, P_e has been fulfilled; halt this process.
- If x is in a position with an unlocked marker $i > e$, "kick" this marker forward (i.e. push it to the position of the next marker, and kick that one and so forth). P_e has now been fulfilled; halt this process.
- If x is in a position whose marker has some number of locks labeled $i_1, \ldots, i_n > e$, kick the marker forward and restart the processes $N_{i_1}, \ldots, N_{i_n}$, albeit with ν defined as $\varphi_e(x(e, s))$, with s equal t the current time of execution. Finally, P_e has been fulfilled; halt this process.

We claim that the set A given by the numbers which are eventually unmarked is a simple, but not effectively simple, set. We now verify this.

- (c.e.) When a position is unmarked, it is never marked again. Moreover, the simulation of the markers is computable. Thus, we enumerate the set by running the simulation indefinitely and printing out every number which is unmarked.

- (coinfinite) It suffices to verify that every marker is kicked a finite number of times. This is the case because the i-th marker will only ever be kicked by the processes $P_0, \ldots, P_{i-1}$, and once by each at most.

- (simple) Let W_e be an infinite c.e. set. Suppose for the sake of contradiction that condition P_e is never fulfilled, in which case the process P_e will run indefinitely.

  Let N be a natural number larger than all the following:

  - The final positions of the markers numbered 0 to e,
  - The locks that will ever have been placed by processes N_0 to N_e.

  Such an N exists because each marker has a finite final position, and each process N_i places finitely many locks each time it runs, and runs at most i times.

  Now, eventually the process P_e enumerates an element of W_e which is larger than N, and by inspection of the simulation it is clear that at this stage P_e will be fulfilled, a contradiction. Thus, A is simple.

- (not effectively simple) Suppose that A is effectively simple with function $f = \varphi_e$. We use the definition of "effectively simple" that presupposes that f is a total function. Thus, whenever the process N_e is made to run, it will finish executing in a finite amount of time. We also know that it is made to run a finite number of times, and that the last time that it is executed the locks that it places will never be removed, and the corresponding markers will stay in place forever. Thus, the corresponding set $W_{x(e,s)}$ shall be contained in A^c. Moreover, it contains $\nu + 1 > \nu = f(x(e, s))$ elements, and so f is not a witness to A being effectively simple. Since this argument holds for every total computable f, A cannot be effectively simple and the proof is complete.