A proof of non-computability

Let us verify that the Busy Beaver Function is not Turing-Computable. (The result is due to Hungarian mathematician Tibor Radó.) As in the case of the Halting Function, we'll proceed by *reductio*. More specifically, we'll assume that there is a Turing Machine M^{BB} that computes the Busy Beaver function, and show that that would allow us to construct an "impossible" Turing Machine M^{I}. Here is how M^{I} works, when given an empty input:

Step 1
For a particular number k (to be specified below), M^{I} starts by printing a sequence of k ones on the tape, and then brings the reader to the beginning of the resulting sequence.

Step 2
M^{I} duplicates the initial sequence of k ones, and brings the reader to the beginning of the resulting sequence of $2k$ ones.

Step 3
M^{I} applies M^{BB}'s program as a subroutine, which results in a sequence of $BB(2k)$ ones.

Step 4
M^{I} adds an additional one to the sequence of $BB(2k)$ ones, returns to the beginning of the sequence, and halts, yielding $BB(2k)+1$ as output.

We will now prove that our impossible machine M^{I} is indeed impossible. Here is the intuitive idea behind the proof. At Stage 3 of its operation, M^{I} produces as long a sequence of ones as a machine with $2k$ could possibly produce. But we'll see that M^{I} is built using no more than $2k$ states. So at Stage 3 of its operation, M^{I} produces as long a sequence of ones as it itself could possibly produce. So when, at Stage 4, M^{I} adds a one to the sequence, it produces a *longer* string of ones than it itself could possibly produce, which is impossible.

Formally speaking, we show that M^{I} is impossible by verifying that its existence would allow us to prove each of the following inequalities (where $\overline{M^{I}}$ is the number of states in M^{I}'s program):

(E1)

$$BB(\overline{M^I}) \geq BB(2k) + 1$$

(E2)

$$BB(\overline{M^I}) < BB(2k) + 1$$

Proof of (E1)

The construction of M^I guarantees that M^I delivers an output of $BB(2k) + 1$, given an empty input. So we know that the productivity of M^I is $BB(2k) + 1$. But, by definition, $BB(\overline{M^I})$ is the productivity of the most productive Turing Machine with $\overline{M^I}$ states or less. Since M^I has $\overline{M^I}$ states, and since its productivity is $BB(2k) + 1$, it must therefore be the case that $BB(\overline{M^I}) \geq BB(2k) + 1$. In other words: (E1) must be true.

Proof of (E2)

We will now show that, when k is chosen appropriately, (E2) must be true. We start with some notation:

- Let a_k be the number of states M^I used to implement Step 1 of its procedure.

  (As you'll be asked to verify below, it is possible to build a Turing Machine that outputs k on an empty input using k states or less. So we may assume that $a_k \leq k$.)

- Let b be the minimum number of states M^I uses to implement Step 2 of its procedure.

  (Since a fixed number of states can be used to duplicate sequences of ones of any length, b can be assumed to be a constant, independent of k.)

- Let c be the number of states M^I uses to implement Step 3 of its procedure.

  (c is simply the number of states in M^{BB}, and is therefore a constant, independent of k.)

- Let d be the number of states M^I uses to implement Step 4 of its procedure.

  (Since a fixed number of states can be used to add an extra one at the end of a sequence of any length, d can be assumed to be a constant independent of k.)

This allows us to count the number of states required by M^I: $a_k + b + c + d$. Since we know that $a_k \leq k$, the following must be true:

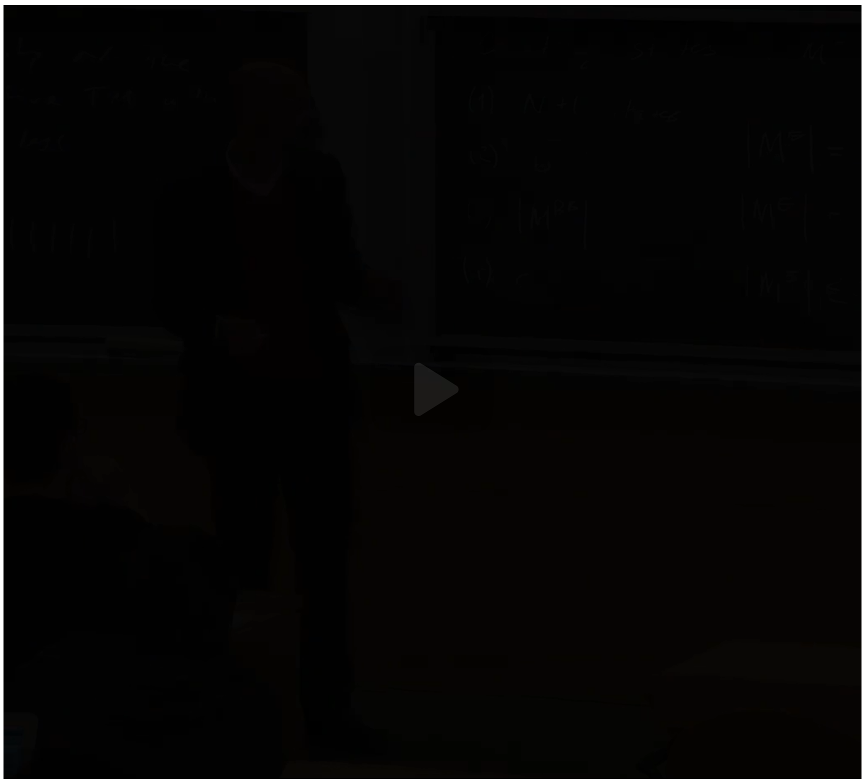
$$\overline{M^I} \leq k + b + c + d$$

Let us now assume that k is chosen so that it is at least as big as $b+c+d$. We then have $\overline{M^I} \leq 2k$. From this it follows that $BB\left(\overline{M^I}\right) \leq BB\left(2k\right)$, and therefore that $BB\left(\overline{M^I}\right) < BB\left(2k\right) + 1$. In other words: (E2) must be true.

Summary

By looking at M^I's output we showed that (E1) must be true. But by analyzing M^I's program we came to the contrary conclusion that (E2) must be true. So M^I cannot exist. So M^{BB} cannot exist. So the Busy Beaver Function isn't Turing-computable.

Video Review: If the Busy Beaver Machine Exists, the Evil Machine Exists



8:08 / 8:08 1.50x

But our assumption is that it does

exist so you would be able to calculate it

and that would be enough to know what beta is.

And that would be enough to know what n is.

And then you just build that into the specification

of your machine.

End of transcript. Skip to the start.

Video

Download video file

Transcripts

Download SubRip (.srt) file

Download Text (.txt) file

Video Review: The Evil Machine Does Not Exist

3:27 / 3:27 1.50x

What in the universe could be cooler than that?

So there can be no such thing as the Evil Turing Machine.

So there can be no such thing Turing Machine computing

the Busy Beaver Function.

So the Busy Beaver Function is not computable.

End of transcript. Skip to the start.

Video
Download video file

Transcripts
Download SubRip (.srt) file
Download Text (.txt) file

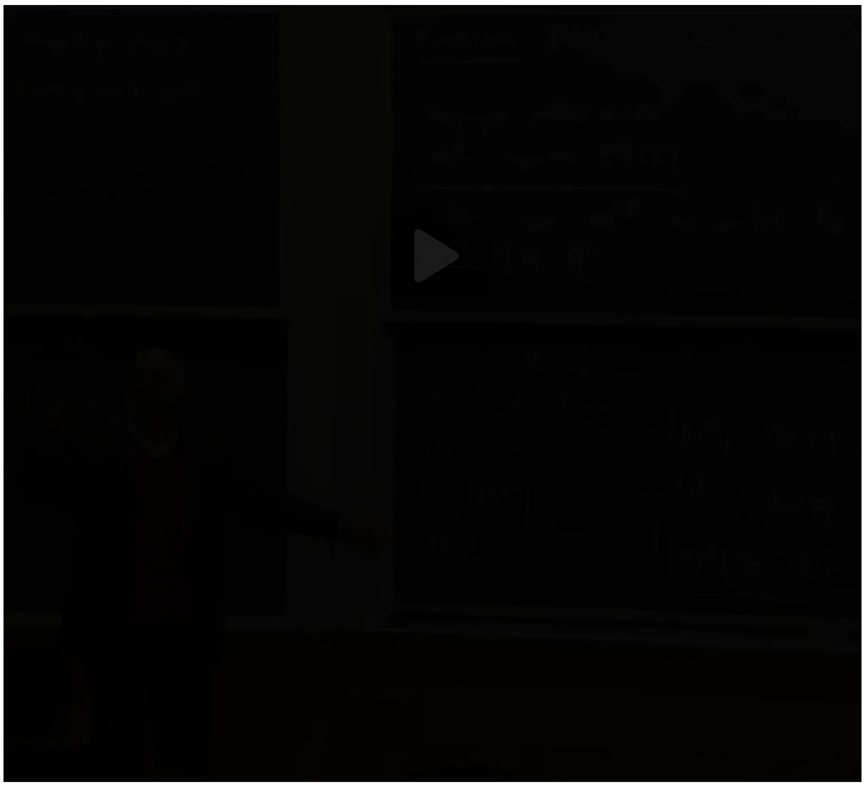
Video Review: The Evil Machine, Step by Step

have no idea what that number is,

but on our assumption, that number exists.

And again, it's constant.

It does not depend on how big n is going to be.



And then, finally, c is just add 1.

Again, some small number of states.

End of transcript. Skip to the start.

1:20 / 1:20 1.50x

Video

Download video file

Transcripts

Download SubRip (.srt) file
Download Text (.txt) file

Problem 1

1/1 point (ungraded)
True or false?

For any natural number k, it is possible to build a Turing Machine that outputs k on an empty input, using k states or less.

● True

○ False

✓

Explanation

When k is 0, the problem is straightforward. When $k \geq 1$, the following Turing Machine gives us what we want:

$$\begin{array}{ccccc} 0 & \_ & 1 & l & 1 \\ 1 & \_ & 1 & l & 2 \\ \dots & & & & \\ (k-1) & \_ & 1 & * & \text{halt} \end{array}$$

Submit

Answers are displayed within the problem

Problem 2

1/1 point (ungraded)

True or false?

For some fixed constant c, it is possible to build a one-symbol Turing Machine that uses at most $\lfloor \log_2 k \rfloor + c$ states and outputs k^2, on an empty input.

[✓ True] False

✓

Explanation

We begin with some observations:

- By using the technique of the previous exercise, one can show that it is possible to build a one-symbol Turing Machine that outputs k in binary notation, on an empty input, and uses at most $\lfloor \log_2 k \rfloor + 1$ states.
- As shown earlier, it is possible to build a one-symbol Turing Machine that outputs k, given k's binary notation as input, and uses only 6 states.
- It is easy (if laborious) to build a one-symbol Turing Machine that outputs k^2, given input k. Assume that this can be done using only d states, for some constant d.

These three observations entail that it is possible to build a one-symbol Turing Machine that outputs k, on an empty input, using at most $\lfloor \log_k \rfloor + 7 + d$ states.

There are much more efficient ways of generating big outputs, but efficient machines tend to rely on clever tricks that are not easy to describe succinctly. Here is a famous example of a two-state machine that halts after producing a sequence of four ones:

$$\begin{array}{ccccc} 0 & \_ & 1 & r & 1 \\ 0 & 1 & 1 & l & 1 \\ 1 & \_ & 1 & l & 0 \\ 1 & 1 & 1 & l & \text{halt} \end{array}$$

Submit

Answers are displayed within the problem

Problem 3

2/2 points (ungraded)

Determine the productivity of each of the following Turing Machines:

$$0 \quad \_ \quad 1 \quad r \quad 0$$

0 ✓

0

$$0 \quad \_ \quad 1 \quad * \quad 1$$

1 ✓

1

Submit

✓ Correct (2/2 points)

Problem 4

1/1 point (ungraded)

True or false?

$BB(n)$ grows faster than any Turing-computable function. (More specifically: for any Turing-computable function $f(n)$, there is an m such that, for every $n > m$, $BB(n) > f(n)$.)

[✓ True]

False

✓

Explanation
Since $f(n)$ is Turing-computable, it is computed by some Turing Machine M^f. By using M^f's program as a subroutine, one can program a Turing Machine that computes $f(2k) + 1$ on the basis of $k + c$ states, for c a constant that does not depend on k. So as long as $k > c$, $BB(2k) \geq f(2k) + 1$. It follows that for every even number n above a certain bound, $BB(n) > f(n)$. A similar argument could be used to show that for every odd number n above a certain bound, $BB(n) > f(n)$.

Submit

Answers are displayed within the problem

Discussion

Hide Discussion

Topic: Week 9 / A proof of non-computability

Add a Post

Show all posts by recent activity

There are no posts in this topic yet.

✖