

**Problem 1: The Busy Beaver Function** (50 pts.)**Background**

Let  $\beta(n)$  be the Busy Beaver function from class. We mentioned that  $\beta$  is not computable, but have given no proof so far. At the heart of the issue lurks the Halting problem, as usual. However, in the case of  $\beta$  it is easier to show that the function grows faster than any computable function.

**Task**

- A. Assume  $f : \mathbb{N} \rightarrow \mathbb{N}$  is a strictly increasing computable function. Show that for some sufficiently large  $x$  we must have  $f(x) < \beta(x)$ . Note that  $f$  is computable by some Turing machine with a fixed number of states.
- B. Conclude that  $\beta$  is not computable.
- C. Herr Prof. Dr. Wurzelbrunft is selling a device on eBay called HaltingBlackBox™ that supposedly solves the Halting Problem. Explain how Wurzelbrunft's gizmo could be used to compute  $\beta$ . Also explain why it is not a good idea to bid on Wurzelbrunft's device.
- D. Show that  $\beta(5) = 4098$ .

**Comment**

Relax, part D is a joke, you don't need to do this.

Pinning down  $\beta(5)$  would be an excellent project for this class. Look at the Marxen-Buntrock example and think hard about what is needed to realize that this machine is not in a loop. Likewise, think about how you could eliminate machines that are "looping" (i.e., machines that will never halt and just write more and more 1's on the tape).

**Problem 2: Write-First Turing Machines** (50 pts.)**Background**

As already mentioned in class, there is nothing particularly sacred about the way we defined Turing machines – many variations ultimately produce the same notion of computability. Here is one.

It is customary to define Turing machines via a transition function of the form

$$\delta : Q \times \Gamma \rightarrow \Gamma \times \Delta \times Q$$

Here  $Q$  is the set of states,  $\Gamma$  the tape alphabet including a blank symbol, and  $\Delta = \{-1, 0, +1\}$  indicates movement of the head. An instruction  $\delta(p, a) = (b, d, q)$  indicates that the machine, when in state  $p$  and reading symbol  $a$  on the tape, will write symbol  $b$ , move the head by  $d$  and go into state  $q$ . Informally the machine cycles through phases

read — write — move — goto

Every action after the read depends on the symbol on the tape. This seems fairly natural, but there are other possibilities. For example, the machine could use a basic cycle

write — move — read — goto

The corresponding transition function has the format

$$\gamma : Q \rightarrow \Gamma \times \Delta \times (\Gamma \rightarrow Q)$$

Thus the machine writes a symbol and moves the head according to the current state. Only then will it read the tape (in a new position) and determine which state to move into. For the sake of clarity we refer to these machines as write-first Turing machines; their traditional counterparts will be called read-first Turing machines.

**Task**

1. Give a precise definition of what it means for a write-first Turing machine to compute a function.
2. Show that every write-first Turing machine can be simulated by a read-first machine.
3. Show that every read-first Turing machine can be simulated by a write-first machine.
4. How do the machines compare in size?

**Comment**

Assume for the sake of simplicity that the tape alphabet is  $\Gamma = \{0, 1\}$ .