

Problem 1: Write-First Turing Machines (30 pts.)**Background**

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, Γ 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 .

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 $\Gamma = \{0, 1\}$.

Problem 2: The Busy Beaver Function (30 pts.)**Background**

The Busy Beaver function β is a famous example of a non-computable function. For our purposes, let's define $\beta(n)$ as follows. Consider all register machines P with n instructions and no input (so all registers are initially 0). Executing such a machine will either produce a diverging computation or some output x_P in register R_0 . Define $\beta(n)$ to be the maximum of all x_P as P ranges over n -instruction programs.

It is intuitively clear that β is not computable: we have no way to eliminate the non-halting programs from the competition. Alas, it's not so easy to come up with a clean proof. One line of reasoning is somewhat similar to the argument that shows that the Ackermann function is not primitive recursive: one shows that β grows faster than any computable function.

Task

1. Show that for any natural number m there is a register machine without input using $O(\log m)$ instructions that outputs m .
2. Assume $f : \mathbb{N} \rightarrow \mathbb{N}$ is a strictly increasing computable function. Show that for some sufficiently large x we must have $f(x) < \beta(x)$.
3. Conclude that β is not computable.
4. Prof. Dr. Blasius Wurzelbrunft sells a device called HaltingBlackBoxTM that allegedly solves the Halting Problem for register machines. Explain how Wurzelbrunft's gizmo could be used to compute β .

Comment

The bound in part (A) is far from tight in special cases: some number m have much shorter programs. But in general $\log m$ is impossible to beat (Kolmogorov-Chaitin program-size complexity). Part (D) says that β is K -computable.

Problem 3: K -Semidecidability (40 pts.)**Background**

Recall that we defined two versions of the Halting set:

$$K = \{ e \mid P_e(e) \downarrow \},$$
$$K_1 = \{ \langle e, x \rangle \mid P_e(x) \downarrow \}.$$

While K_1 seems slightly more complicated, the two versions have the same degree of difficulty: given one as an oracle, it is easy to decide the other. Here are two more complicated properties of register machines: convergence on only finitely many inputs and totality:

$$\text{FIN} = \{ e \mid \exists x \forall y (P_e(y) \downarrow \Rightarrow y \leq x) \}$$
$$\text{TOT} = \{ e \mid \forall x \exists y (P_e(x) = y) \}$$

Task

- A. Show that K_1 is K -decidable.
- B. Show that FIN is K -semidecidable.
- C. Show that TOT is FIN-decidable.
- D. Show that FIN is TOT-decidable.

Comment

Parts (C) and (D) mean that from the perspective of relative decidability FIN and TOT are the same.