15-455: UCT — K. Sutner

Assignment 1 — Due: **TBA**.

1. Fractran GCD (30)

Background

For the following problem, we want a Fractran program F thus uses the standard coding convention: $3^a 5^b \rightsquigarrow 2^{\mathrm{gcd}(a,b)}$. Try to make F as short as ever possible.

Task

1. Construct a Fractran program F that implements gcd.
2. Explain in detail why your program is correct.

2. Write-First Turing Machines (30)

Background

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

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

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. So the inner loop looks like this:

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

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

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 $\Gamma = \{0, 1\}$ with 0 the blank symbol.

3. Minimal Machines (40)

Background

All models of computation can be associated with a natural size function. This is particularly obvious for machine-based models: the machine is just a finite data structure, and has a canonical size. For example, we could define the size of a Turing machine M to be the product $|Q||\Sigma|$, or the number of bits needed to specify its transition function. Or we could think of the index $\widehat{M}$ as a natural number, and use that number.

Fix one such measure, and call M minimal if no smaller machine is equivalent to M. Here equivalent means that $\forall z\,(M(z) \simeq M'(z))$: the computations may unfold in a different way, but the final result has to be the same for all inputs.

Task

1. Explain intuitively why minimality of TMs should not be semidecidable. You might want to start with decidability.
2. Assume that minimality of TMs is semidecidable. Show that there is an effective enumeration (N_e) of all minimal TMs.
3. Show that minimality of TMs fails to be semidecidable using the recursion theorem and part (A).

Comment

This can be proven without the recursion theorem, but the argument is much easier using the theorem.