# 15-455: Undergraduate Complexity Theory Assignment 1

Zhikun Wang[*]

February 1, 2021

## 1 Fractran GCD

### 1.1 Program construction

The program is [87/41, 145/23, 47/435, 17/29, 26/51, 26/85, 17/13, 1/17, 795/47, 133/795, 53/19, 31/53, 55/93, 31/11, 37/31, 129/259, 37/43, 29/37, 41/3, 23/5].

The following Python program simulates the Fractran program.

```
l = [(3*29,41),(5*29,23),(47,3*5*29),(17,29),(2*13,3*17),(2*13,5*17),(17,13),\\
(1,17),(53*3*5,47),(7*19,53*5*3),(53,19),(31,53),(5*11,3*31),(31,11),(37,31),\\
(3*43,7*37),(37,43),(29,37),(41,3),(23,5)]
a = int(input())
while 1:
    na = a
    for i in l:
        if na%i[1] == 0:
            na = na*i[0]//i[1]
            break
    if na == a:
        break
    else:
        a = na
print(a)
```

### 1.2 Explanation for correctness

All primes greater than ”7” serves as a flow control prime, primes 3, 5 and 7 are variables. The last two and the first two fractions serves to add a ”flow control prime”. The program runs as follows:

- Multiply the input with the control prime.
- If the number is not dividable by 15, output(change all powers of 3 and 5 to powers of 2).
- Change power of 7 to min(power of 3, power of 5), and change powers of 3 and 5 to max(power - other power,0).
- Move all powers of 3 to powers of 5, and then change all powers of 7 to powers of 3
- Return to the 1st step.

[*]nocrizwang@gmail.com

# 2 Write-First Turing Machines

## 2.1 Precise definition

The definition of computation is precisely the same as read-first Turing machines.

## 2.2 Every write-first Turing machine can be simulated by a read-first machine.

For every state in the write-first TM, construct two states in the read-first TM. One is the moving state, and the other is the reading state. In the moving state the TM moves its head in the corresponding position regardless of the symbol and changes into the corresponding reading state. Then, in the reading state the read-first TM selects which moving state to change to, without changing what is just read.

## 2.3 Every read-first Turing machine can be simulated by a write-first machine.

For every state in the read-first TM, construct two states for each symbol('0' and '1') that may be read in the write-first TM. The initial state for the write-first TM will be the state constructed from the initial state of the read-first TM and symbol '0', since it will always first read '0'. Each state writes and moves the same as the read-first TM, and with the new symbol read, changes to the corresponding state.

## 2.4 Machine size comparison

Simulation from both directions takes 2 times the states.

# 3 Minimal Machines

## 3.1 Explain intuitively why minimality of TMs should not be semidecidable. You might want to start with decidability.

The non-minimality is semidecidable. If minimality is also semidecidable then minimality will be decidable, which is extremely counter intuitive.

## 3.2 Assume that minimality of TMs is semidecidable. Show that there is an effective enumeration $(N_e)$ of all minimal TMs.

Let $M$ be the TM that accepts a TM if it is minimal. The enumerator runs as follows:

Enumerate x from 1 to infinity. Enumerate y from 1 to x, for each y, run $M$ on y for x steps, if $M$ halts after exactly x steps, output y.

## 3.3 Show that minimality of TMs fails to be semidecidable using the recursion theorem and part (A).

Let M be the TM that accepts a TM if it is minimal, otherwise does not halt.

Let D be a TM that runs as follows:

- Get D's representation using the recursion theorem
- Use the enumerator, let the first TM greater than D be E
- Return E(x)

Then E would not be minimal.