On The Undecidability of the Generalized Collatz Problem

Julien Malka
École Normale Supérieure

Abstract

This work present the Collatz conjecture, one natural generalization and the proof of its undecidability. We also discuss the implications of the result on the hardness of the initial Collatz conjecture.

1 Introduction : The Collatz Conjecture

The Collatz conjecture, or as french people often call it, the Syracuse conjecture, is a famous mathematical conjecture that despite its apparent simplicity, remain unsolved to this day.

Let's consider a function f defined as follow :

$$f(n) = \begin{cases} \frac{n}{2} & \text{if } n \equiv 0 \pmod 2 \\ 3n+1 & \text{if } n \equiv 1 \pmod 2. \end{cases}$$

Now we can form a sequence (a_i) by starting with a number n and recursively computing the elements of the sequence using the function f.

$$a_i = \begin{cases} n & \text{if } i = 0 \\ f(a_{i-1}) & \text{if } i > 0 \end{cases}$$

The Collatz (or Syracuse) conjecture states that $\forall n, \exists i$ such that $a_i = 1$, that is for any starting point n, the sequence (a_i) will eventually reach 1.

Example. Taking 14 as a starting point, we get the sequence 14, 7, 22, 11, 34, 17, 52, 26, 13, 40, 20, 10, 5, 16, 8, 4, 2, 1, 4, 2 that indeed falls into the cycle $1, 4, 2, 1$.

This conjecture has been verified experimentally for n up to 87×2^{60} but is yet to be proven right or wrong.

2 A Natural Generalization

We are going to present a natural generalization of the Collatz problem that was introduced by Conway in [1].

Definition. Collatz Function
A function g is said to be a *Collatz function* if there is $m \in N$ and two sequences (a_i) and (b_i) such that $\forall x \equiv i \pmod{m}$, we have $g(x) = a_i x + b_i$ and $g(x)$ is an integer.

Example. The function h defined below is a Collatz function.

$$h(n) = \begin{cases} \frac{2}{3}n & \text{if } n \equiv 0 \pmod 3 \\ \frac{4}{3}n - \frac{1}{3} & \text{if } n \equiv 1 \pmod 3 \\ \frac{4}{3}n + \frac{1}{3} & \text{if } n \equiv 2 \pmod 3 \end{cases}$$

Remark. One can see this indeed generalize the original conjecture because the Collatz function with $a_0 = 1/2, b_0 = 0, a_1 = 3, b_1 = 1$ is the function f defined earlier.

Definition. Generalized Collatz Problem
Given g, a Collatz function, and n an input, the Generalized Collatz Problem is the decision problem that is true iff the sequence reaches 1.

The goal of the paper is to prove the following theorem.

Theorem. The Generalized Collatz Problem is undecidable.

3 Some preliminary tools

The proof of the above theorem requires prior knowledge of several models of computation that might be unknown, the next part is made to introduce them and to prepare the reduction that will be the core of the proof.

3.1 Counter Machines

3.1.1 Minsky machine

A counter machine is an abstract model of computation introduced in [4] by Minsky. It is a machine equipped by a finite number of counters containing a non negative number and controlled by a finite sequence of instructions. There are several kind of counter machines, but the one introduced by Minsky works with three instructions. The input to the machine is put in the counter 0 and the output is also found in this counter once the machine has halted.

- INC(c, l) that increments counter c and then jumps to line l
- DEC$(c, l_{success}, l_{failure})$ that tries to decrement counter c. If it was originally 0 then it jumps to $l_{failure}$ else it jumps to $l_{success}$
- HALT that halts the machine

Theorem. Counter machines can simulate Turing machines.

Proof. First, we are going to show that 2-PDAs can simulate Turing machines and then that counter machines can simulate 2-PDAs.
To simulate a Turing machine on a 2-PDA, the trick is rather easy, one stack contains the left part of the tape, the other stack contains the item under the head and the right part of the tape.

To simulate a 2-PDA with stack alphabet $X_1, X_2, ..., X_k$, we use a 3-counter machine and store the content of the 2 stacks in 2 counters. To encode the content of the stacks we use a $k-$ary representation of their content. The last counter is used to implement the stack operations. For example to push an element X_i in stack 0, we have to update the associated counter by multiplying to by k and adding i. This can be done with the additional counter.

□

3.1.2 Tweaked Minsky Machine

In our reduction, we are going to use a simplified version of the Minsky machine with these added restrictions :

- The machine takes its input in counter 0 and output in counter 1
- No instruction can branch to itself
- The only halting instruction is the last one
- Before halting, the machine set all its counters but counter 1 to 0.

One can verify that these restrictions are made without losing any generality by reducing Minsky's counter machine to this tweaked machine.

3.2 FracTran

FracTran is a Turing-complete programming language created by Conway in [2].
A FracTran program is represented by a sequence (q_i) of rational numbers.
From an input n, the program is run by updating n by the first element of the sequence (nq_i) that is an integer. If no such element exist the program halts. Else, it continue to do the same operation.

Example. A FracTran program computing the prime numbers
$\left(\frac{17}{91}, \frac{78}{85}, \frac{19}{51}, \frac{23}{38}, \frac{29}{33}, \frac{77}{29}, \frac{95}{23}, \frac{77}{19}, \frac{1}{17}, \frac{11}{13}, \frac{13}{11}, \frac{15}{14}, \frac{15}{2}, \frac{55}{1}\right)$

4 The reduction

The proof of the theorem is essentially a reduction from the Halting problem to the Generalized Collatz Problem. The reduction has originally been proposed by Conway and is separated in two steps : a reduction from Counter Machines to FracTran, and one from FracTran to the Generalized Collatz Problem.

4.1 From Counter Machines to FracTran

We are going to show that we can represent a counter machine (that can itself represent a Turing machine) with a FracTran program. We are going to present a reduction from our tweaked Minsky machine to FracTran. We want that for any computable function K computed by a counter machine, we are able to cook up a FracTran program P such that $P(2^{x+1}) = 3^{H(x)+1}$. The trick essentially resides in a clever encoding of the state of the counter machine in the input of the FracTran program. Let p_i denote the i_{th} prime number. Suppose we want to represent a k-counter machine with l instructions with a FracTran program. We are going to encode the state of the machine in the input n. We encode the states of the counters using the k first primes and storing the state of the counters in the exponents. Similarly, we use the next l primes to represent in which state the machine is : if the machine is in state 2, p_{k+l} has exponent 1 and all the other primes exponent 0.

Example. $n = 2^0 3^4 5^0 7^0 11^1$ is an input encoding a 2-counter machine with counter $2 = 4$ and that is currently in state 3.

Now that we have an encoding for the state of the machine we need to be able to translate the 3 instructions of the counter machine in FracTran language.

- INC(c, l) on line h : We first multiply by p_c to increment the counter and then multiply by $\frac{p_{k+l}}{p_{k+h}}$ to go to state l. Hence the instruction is $\frac{p_c}{p_{k+l} \times p_{k+h}}$
- $DEC(c, l_{success}, l_{failure})$ on line h : We first divide by p_c to decrement the counter and then multiply by $\frac{p_{k+l_{success}}}{p_{k+h}}$ to go to state $l_{success}$. In case of failure (the counter is already at 0), only multiply by $\frac{p_{k+l_{failure}}}{p_{k+h}}$ Hence the instruction is $\frac{p_{k+l_{success}}}{p_{k+h} \times p_c} \times \frac{p_{k+l_{failure}}}{p_{k+h}}$.
- HALT The halt instruction does not need translation.

We still need to take care of the initialization and ending of our program. To compensate for the added exponent and set up the first state we add $\frac{p_{k+1}}{2}$ at the beginning of the program. To add the missing exponent and remove the last state we add $\frac{3}{p_{k+l}}$ at the end of the program.
This concludes our reduction.

4.2 From FracTran to the Generalized Collatz Problem

We can easily see that the transition function of a FranTran program indeed define a Collatz function k_f (indeed, FranTran has been designed by Conway exactly for that reason). Let's denote d_i the denominators of the (q_i) and call m the least common multiple of the (d_i). It will be the modulus of our Collatz function. Now we take the (q_i) in order, and for each of the residues r mod m, if r mod $d_i = 0$, then $a_r = q_i$. At the end of the procedure all the unset a_i are set to 1 and the (b_i) are set to 0.

Example. The FracTran program $(\frac{1}{3}, \frac{5}{2})$ would give this Collatz function :

$$j(n) = \begin{cases} \frac{1}{3}n & \text{if } n \equiv 0 \text{ or } n \equiv 3 \pmod 6 \\ \frac{5}{2}n & \text{if } n \equiv 2 \text{ or } n \equiv 4 \pmod 6 \\ n \text{ else} \end{cases}$$

Let's define $F(x)$ the output of the FracTran program with input x. We have that $F(x)$ is defined if $\exists i$ such that $k_f^{(i)}(2^{x+1}) = 3^{z+1}$ for some z. We would like that $F(x)$ is defined iff $\exists i$ such that $k_f^{(i)}(2^{x+1}) = 1$. To solve this last problem, we tweak k_f so that $a_r = \frac{1}{3}$ for $r = 3^k \pmod m$, concluding the reduction.

We now have that from a Turing machine T, we can create a Collatz function that verifies the Collatz conjecture with input 2^{x+1} iff T halts. As the halting problem is undecidable, the Generalized Collatz Problem is undecidable too.

5 Implications and Conclusion

We've proven that there exist at least one Collatz function for which the Generalised Collatz problem is undecidable for some input.

A 2007 paper from Kurtz and Simon [3] proved a result that is even closer than the original Collatz conjecture, that is given a Collatz function, it is undecidable to know if the Collatz conjecture is true for this function. They also proved that this problem is Π_2^0-complete. In clear, there is no Turing machine that takes a Collatz function in input, halts and outputs true iff the Collatz conjecture is verified for the function. It is important to understand that this last result doesn't imply that the original Collatz conjecture is false, or undecidable, but it can give us insights on the hardness of the problem. Indeed, the fact that the general problem is undecidable doesn't mean that all of its instances are.

References

[1] J. Conway. Unpredictable iterations. 1972.

[2] J. H. Conway. Fractran: A simple universal programming language for arithmetic. In *Open Problems in Communication and Computation*, pages 4–26. Springer, 1987.

[3] S. A. Kurtz and J. Simon. The undecidability of the generalized collatz problem. In *International Conference on Theory and Applications of Models of Computation*, pages 542–553. Springer, 2007.

[4] M. Minsky. Recursive unsolvability of post's problem of "tag" and other topics in theory of turing machines. *Annals of Mathematics. Second Series*, 74, 12 1966.