Can the Collatz conjecture be proven, or not?

Hans-Dieter A. Hiep

November 1, 2023

1 Introduction

In 1937, shortly after the mathematician Lothar Collatz obtained his doctorate, he wrote down a problem in his notebook that later became known as *Collatz' problem* or *the* $(3x+1)$*-problem*. The problem is remarkable since it is easy to state, but for more than eighty years no solution had been found.

Before the problem can be stated, we first need two ingredients: we define a function, and we recall what is repeated function application. The function f is defined on the positive natural numbers (1, 2, 3, et cetera) with the following specification:

$$f(x) = x \div 2 \text{ if } x \text{ is even,}$$
$$f(x) = 3 \times x + 1 \text{ if } x \text{ is odd.}$$

Now take an arbitrary positive natural number n. We can repeatedly apply the function f starting with n, i.e. $f(n)$, $f(f(n))$, $f(f(f(n)))$, et cetera. We write $f^i(n)$ to mean that the function f is applied i times starting with n, so that we have $f^1(n) = f(n)$ and $f^{i+1}(n) = f(f^i(n))$. The superscript notation, an operation on a function, should not be confused with exponentiation, an operation on a number. With this in mind, we can state the problem: for every positive natural number n, is there an i such that $f^i(n) = 1$?

We can first try out a number of examples, to gain some intuition about the problem. If we take $n = 1$, can we find an i? Clearly, $f(1)$ is 4, since 1 is odd and so the first clause of the definition of f applies, and $3 \times 1 + 1$ is 4. Then we evaluate $f(f(1))$, which is $f(4)$: since 4 is even the second clause applies so we have $f(4) = 2$. Finally, we evaluate $f(f(f(1)))$, which is $f(2)$: since 2 is even we have $f(2) = 1$. So $f^3(1) = 1$, hence we can take i to be three. For $n = 1$ the problem can be solved.

The process we tried is called iteration. We find i by starting with the smallest value and try out successively larger values for i until we reach our desired destination. This process works efficiently, since the computations we did in the past to find out whether i was a solution can be reused to find out whether $i + 1$ is a solution. In the case of Collatz' problem, we see that the outcome of the previous try, i.e. $f^i(n)$, is what we feed back as input in the next try, i.e. $f(f^i(n))$ computes $f^{i+1}(n)$.

Such iterative processes can be written down neatly. Say we start with $n = 5$. How often do we need to apply the function f until we reach the destination 1? We write the following sequence, where each number is separated by an arrow:

$$5 \longrightarrow 16 \longrightarrow 8 \longrightarrow 4 \longrightarrow 2 \longrightarrow 1.$$

the computation sequence starts with 5 and we get 16 after applying f once ($3 \times 5 + 1 = 16$), we get 8 after applying f again (so twice from our starting point), and so on, until we reach our destination after applying f five times from where we started. In other words, we have $f^5(5) = 1$. So for $n = 5$, we know there exists an i, namely 5, such that $f^i(n) = 1$. But meanwhile we also discovered for $n = 16$, $n = 8$, $n = 4$ and $n = 2$ a solution, since these were the intermediary numbers that turned up in the computation starting in 5.

As another example, take $n = 3$. We then have $f(3) = 3 \times 3 + 1 = 10$, and $f(10) = 5$, and from that point onward we already know what happens. Starting with the number 5, we see that we can extend the computation sequence towards the right (each time computing the function f), but we can *also* extend the computation sequence towards the left:

$$\ldots \longrightarrow 3 \longrightarrow 10 \longrightarrow 5 \longrightarrow \ldots$$

What would be the number before 3? And how would the computation sequence starting from 7 look like? The reader may try answering these questions, to gain some intuition about Collatz' problem.

For some numbers n, the solution is not immediately obvious. The reader may try out $n = 27$ (but, beware, the computation takes more than 100 steps). In fact, for large enough n, the intermediary numbers that the computation goes through can be used to generate pseudo-random numbers that passes standard tests for randomness [9]. This fact may give us the impression that the numbers involved in the computation do not give much insight into solving the problem.

Now, the conjecture states that *there is* a solution for every n. This is what is known as the 'Collatz conjecture'. Although the problem is quite old, more recently a new interest for the problem emerged—as witnessed by the many published articles, including scientific articles (e.g. [3, 2, 1]) and articles in the popular press (e.g. *The Simple Math Problem We Still Can't Solve* in Quanta-Magazine). Also on social media such as MathOverflow, unanswered questions are raised, such as "How to know when the Collatz conjecture has been proved?"

In 2013, the late John H. Conway wrote about the problem in The American Mathematical Monthly, in an article entitled *On unsettleable arithmetical problems* [5]. In that article, Conway also speaks of well-known results such as Turing's unsolvable halting problem or Gödel's incompleteness theorem. The halting problem is the question whether an algorithm halts (which, as we shall see, has Collatz' problem as an instance) and Turing showed that it is in general unsolvable. The incompleteness theorem states that there are true statements in a formal system called Peano arithmetic that cannot be proven from Peano's axioms. The two results are closely related: there is a proof of the incompleteness theorem by reduction to the halting problem [8].

Conway argues that it is very unlikely that the Collatz problem is settleable—a technical term he introduces for true assertions, as phrased in a set theoretical language, that can also be proven. A problem is unsettleable if it is true, but cannot be proven. There surely are unsettleable (or, unsettling?) assertions: the axioms of set theory are insufficient for proving all true assertions, by Gödel's incompleteness theorem. But I find Conway's argument, that Collatz conjecture is unlikely settleable, too difficult to grasp—it is probably because I lack the background knowledge and intuition behind his argument. However, Conway is not sure, and still leaves open the possibility for the conjecture to be proven:

> "I don't want readers to take these words on trust but rather to encourage those who don't find them convincing to try even harder to prove the Collatz Conjecture!" [5]

A quick search on Google Scholar shows articles, that claim to have proven the Collatz conjecture. These articles are not published, but are available on pre-print servers or institutional repositories on the Web. How can we evaluate such articles? Are they really proofs? Is it worth our time to review them?

By accident, I stumbled upon the pre-print paper *Collatz conjecture becomes theorem* by Mirkowska and Salwicki [7]. Grażyna Mirkowska is a professor emeritus of Warsaw University, and she is an expert on mathematical logic, program semantics, and formal verification. Andrzej Salwicki is also professor emeritus, founder of the journal Fundamenta Informaticae, and he is an expert on the mathematical foundations of computer science. Both worked together on the 1978 book on *Algorithmic Logic* [6], among other works. Algorithmic logic is closely related to my field of expertise, Hoare's logic and dynamic logic.

Maybe I can understand what is written in their article? In the final remarks, the authors write:

> "We know that our presentation is clumsy (we are of age, English is not our native language)." [7]

It is not a good idea to reject this paper because of a clumsy presentation: the authors have a track record in the field, and their work must be taken seriously. Maybe I can figure out whether their paper makes sense to me? The adventure thus begins... And in this article, I do not present my own work, but I present what I could distill from what Mirkowska and Salwicki wrote.

This article assumes the reader has knowledge of Hoare's logic. See, for reference: *A Discipline of Programming* by Edsger Dijkstra (1976), *Mathematical Theory of Program Correctness* by Jaco de Bakker (1980), *The Science of Programming* by David Gries (1981), *Program Verification* by Nissim Francez (1992), *Verification of Sequential and Concurrent Programs* by Krzysztof Apt, Frank de Boer & Ernst-Rüdiger Olderog (2009). See the survey paper by Apt and Olderog [4] for more references.

```
while x ≠ 1 do
  if even(x) then
    x := x ÷ 2
  else
    x := 3 × x + 1
  fi
od
```

Figure 1: A program representation of the Collatz conjecture.

2 Problem statement

First, we revisit the conjecture. We can understand the conjecture in a different way, by studying the program in Figure 1. Looking at the program as given, we have the following primitive operations and tests:

- the test $x \neq 1$ for deciding whether the number x is not equal to 1,
- the test $even(x)$ for deciding whether x is even,
- the primitive operation $x := x \div 2$ for dividing the number by two—but this operation is only executed in the context where we know that the old value of x is even and not equal to 1,
- the primitive operation $x := 3 \times x + 1$ that multiplies the old value of x by the constant three and adds the constant one—this operation is only executed in the context where we know that x is odd and not equal to 1.

If the program terminates on every input $x \geq 1$, then the Collatz conjecture is true. From the terminating execution of the program we can then extract the computation sequence, simply by looking at the values that x take over time. If, however, the program runs infinitely for some input $x \geq 1$, then we have a counterexample to the Collatz conjecture.

Let us abbreviate the program in Figure 1 by S. We reformulate the question, whether the program S terminates or not, as follows:

- If $\{x \geq 1 \wedge \phi\}\ S\ \{\mathbf{false}\}$ is provable for some ϕ in Hoare's logic for partial correctness such that the precondition is satisfiable, then we know that the Collatz conjecture is false.
- If $\{x \geq 1\}\ S\ \{\mathbf{true}\}$ is provable in Hoare's logic for total correctness, then we know that the Collatz conjecture is true.

This formulation raises a number of question, as often is the case in Hoare's logic. What is the language we use in assertions? What is the program theory? And what is the background theory?

What is the logical language we use in assertions? We restrict ourselves to a first-order language consisting of addition only. This language consists of first-order formulas with respect to a signature with:

- the constant individual symbol 0,
- the constant individual symbol 1,
- the binary function symbol $+$.

We can also introduce abbreviations: for every natural number n we have the numeral $\underline{n}$. We have $\underline{0} = 0$, and for any $n \geq 0$ we have $\underline{(n+1)} = \underline{n} + 1$. In other words, we have the numerals:

$$\underline{0} = 0,\ \underline{1} = 1,\ \underline{2} = 1 + 1,\ \underline{3} = (1 + 1) + 1,\ \underline{4} = ((1 + 1) + 1) + 1,\ \ldots$$

Note that these numerals are terms constructed from constants and function symbols, hence do not depend on the value of variables: the numerals are all ground. Also, we implicitly used the fact that $0 + 1 = 1$ since this follows from the background theory we introduce later.

We introduce the abbreviation $x < y$ to stand for $(\exists z)(z \neq 0 \wedge x + z = y)$ where z is fresh (so not equal to either x or y). $y > x$ abbreviates $x < y$, and $x \geq y$ abbreviates $y < x \vee y = x$.

In a similar way as the numerals, can also introduce abbreviations for multiplication by a numeral: $\underline{0} \times x = 0$, $\underline{(n+1)} \times x = x + (\underline{n} \times x)$ for $n \geq 0$. In other words, we have the abbreviations

$$\underline{0} \times x = 0,\ \underline{1} \times x = x,\ \underline{2} \times x = x + x,\ \underline{3} \times x = x + x + x,\ \ldots$$

where we implicitly assume that $+$ is associative, and $x + 0 = x$ (again, we shall introduce the background theory later, from which these properties follow).

What is the program theory? This question amounts to showing how we axiomatize the primitive operations and tests. We introduce a predicate symbol for the test $even(x)$, where x is a variable. We can define this predicate symbol in our language as follows:

$$even(x) \equiv_{\text{def}} (\exists y)(x = y + y).$$

It is necessary that tests are decidable. Otherwise, if one would execute the program one can not make the case distinction in the **if**-statement.

Given that we have a formal understanding of the test, how do we axiomatize the two updates? We introduce the following axiom schemes.

$$\{(\exists y)(x = y + y \wedge \phi[x := y])\}\ x := x \div 2\ \{\phi\} \text{ where } y \text{ is fresh}$$

The precondition of the division-by-two update states that the original value of x must be even before executing the operation. The witness of evenness, y, is substituted for x in the postcondition—and we require the variable y to be fresh, that is, not already occurring in ϕ and different from the variable x.

$$\{\phi[x := (\underline{3} \times x) + 1]\}\ x := 3 \times x + 1\ \{\phi\}$$

The precondition of the times-three-plus-one update is the weakest precondition given the postcondition ϕ. Notice how we are able to express the new value of x in terms of the old value of x only by using the constant symbol 1 and function symbol + from our signature (recall that our multiplication by a numeral is an abbreviation, viz. $x + x + x$).

Now that we have axiomatized the test and primitive operations, we turn to the last question. What is the background theory? Until now, one may freely interpret what the language means. By means of a background theory we restrict the possible interpretations. From the background theory, we can derive consequences which must hold in every program state.

As background theory we take Presburger arithmetic. Presburger arithmetic is a restriction of Peano arithmetic, as it speaks only of the addition operation on numbers. By Gödel's incompleteness theorem, the formal theory of Peano arithmetic can be shown to be incomplete, that is, there are valid sentences that are not consequences of the axioms. What is remarkable about Presburger arithmetic, however, is that it's theory is complete: all valid sentences are also consequences of the axioms.

Furthermore, the axiomatization of Presburger arithmetic is recursive, that is, there exists a procedure to decide what are the axioms of Presburger arithmetic. From this, we also obtain that the formal theory of Presburger arithmetic is decidable: for any sentence, either we can use the standard proof system of first-order logic and prove that the sentence is valid, or we can effectively find a model that satisfies all axioms but in which the sentence is false.

There are different presentations of the same theory. We shall give one that is short on paper, but it is presented by using the unary function s. The unary function s is defined by $s(x) = x + 1$, and we have that $1 = s(0)$. The other axioms of Presbuger arithmetic are the following:

1. $(\forall x)(s(x) \neq 0)$,
2. $(\forall x)(\forall y)(s(x) = s(y) \to x = y)$,
3. $(\forall x)(x + 0 = x)$,
4. $(\forall x)(\forall y)(s(y) + x = s(y + x))$,
5. $\phi(0) \wedge (\forall x)(\phi \to \phi(s(x))) \to (\forall x)\phi$.

The last is an axiom scheme for any formula ϕ, and where $\phi(t)$ is the result of replacing the free variable occurrences of x by the term t. From these axioms, the usual properties of addition follow. For example:

- $(\forall x)(\forall y)(x + y = y + x)$,
- $(\forall x)(\forall y)(\forall z)(x + (y + z) = (x + y) + z)$,
- $(\forall x)(x \neq 0 \to (\exists y)(x = y + 1))$,
- $(\forall x)(\forall y)(\forall z)(x + z = y + z \to x = y)$.
- et cetera.

3 Different interpretations

Now, we turn to the semantics of the program S, of which the halting problem captures the Collatz conjecture. A program is nothing but a piece of text: a finite string of symbols. One may think of the intended meaning of a program, but nothing prevents two people looking at the same string of symbols and interpret it differently. To illustrate this concept, we shall look at two interpretations of S. The fact that this simple program can be interpreted in different ways, somewhat delighted me when I first read the pre-print paper by Mirkowska and Salwicki [7].

The standard interpretation is given by taking the following data structure:

- Take the natural numbers $\mathbb{N} = \{0, 1, 2, 3, \ldots\}$ as domain.
- Interpret the symbols in the usual way: $+$ is addition of natural numbers, and $<$ is less than.

This standard interpretation satisfies all the axioms of Presburger arithmetic.

Alternatively, one could give a complex interpretation, as follows:

- Take the following subset of the complex numbers $\mathbb{C}$ as domain:

  $$\{k + w\iota \mid k \in \mathbb{Z} \text{ and } w \in \mathbb{Q}^+ \text{ and if } w = 0 \text{ then } k \geq 0\}$$

  where k is an integer, w is a non-negative rational, and ι is $\sqrt{-1}$. (We use the Greek ι instead of the Latin i, to avoid confusion with the natural number i we used earlier in function iteration f^i.) The condition implies that we have no negative real numbers in our domain.
- The constant individual symbols 0 and 1 are interpreted as the complex numbers $0 + 0\iota$ and $1 + 0\iota$, respectively.
- The binary function symbol $+$ is interpreted as usual:

  $$(k + w\iota) + (k' + w'\iota) = (k + k') + (w + w')\iota.$$

  Note in this equation alone, the symbol $+$ has four different meanings: the function symbol $+$ in our language, we have the $+$ on complex numbers, we have the $+$ on integers, and we have the $+$ on non-negative rationals!

This complex interpretation satisfies all the axioms of Presburger arithmetic.

In the complex interpretation, we have the so-called *reachable* elements and the *unreachable* elements. The reachable elements are the complex numbers $k + w\iota$ in our domain with a zero imaginary part (and so, by the condition, we know that $k \geq 0$). The unreachable elements are the complex numbers in our domain with a non-zero imaginary part. Any operation performed on only reachable elements, gives us back a reachable element. However, an operation performed with at least one unreachable element, results in an unreachable element. This follows from the fact that the imaginary part is non-negative.

In the complex interpretation, the order of elements is defined by $x < y$, which abbreviates $(\exists z)(z \neq 0 \wedge x + z = y)$. We can understand the order relation on complex numbers $(k + w\iota) < (k' + w'\iota)$ as it were a lexical order on pairs: $\langle w, k\rangle < \langle w', k'\rangle$. This means that all reachable elements (where $w = 0$) are below all unreachable elements (where $w > 0$). Further, in the complex interpretation we have that an element is even iff the real part is even.

With our complex interpretation, we can actually give an infinite run of the program! The following is a demonstration of a computation sequence:

$$(8 + \frac{1}{2}\iota) \longrightarrow (4 + \frac{1}{4}\iota) \longrightarrow (2 + \frac{1}{8}\iota) \longrightarrow (1 + \frac{1}{16}\iota) \longrightarrow$$
$$(4 + \frac{3}{16}\iota) \longrightarrow (2 + \frac{3}{32}\iota) \longrightarrow (1 + \frac{3}{64}\iota) \longrightarrow (4 + \frac{9}{64}\iota) \longrightarrow \ldots$$

The computation sequence can be mapped back to an execution of the program (this takes a bit of work, but it is not difficult to see that the number generated each step is the value of the variable x right at the start of each iteration of the **while**-loop). Notice that, since we start with an unreachable value (the imaginary part is non-zero), the test of the loop never fails, and so the body of the loop is always taken. Any of the two operations (division-by-two or multiply-by-three-add-one) on an unreachable value result in an unreachable value again, which can never be equal to the reachable value of 1.

However, this infinite run cannot be used to argue that the Collatz conjecture is false, since the conjecture speaks of the positive natural numbers and not the complex numbers we have introduced. So, coming back to our reformulated question: in Hoare's partial correctness logic, can we prove

$$\{x \geq 1 \wedge \phi\}\ S\ \{\mathbf{false}\}$$

for some ϕ, and show that the precondition is satisfiable? No, we can not prove this *based on our example above*. The reason why is that the complex interpretation is elementarily equivalent to the standard interpretation. This means that the same first-order sentences are true in both the standard interpretation as in the complex interpretation. Since our assertion language uses a first-order language, we cannot express in ϕ that the starting value of x has to be an unreachable element. Namely, the ability to express that x is an unreachable element (which would be false in the standard interpretation, but true in the complex interpretation under existential quantification) contradicts that the standard and complex interpretations have the same first-order theories.

Note, however, that the above correctness formula may still be provable—showing the Collatz conjecture is false. If one gives a proof of

$$\{x \geq 1 \wedge \phi\}\ S\ \{\mathbf{false}\}$$

in Hoare's logic for partial correctness, then the proof must be sound for the standard interpretation too. If one then also shows that ϕ is satisfiable in the standard interpretation, then the Collatz conjecture is settled (to be false).

On the other hand, if we look at proving

$$\{x \geq 1\}\ S\ \{\mathbf{true}\}$$

in Hoare's logic for total correctness, are we sure that a proof means that the Collatz conjecture is true? In Hoare's total correctness logic, we employ a proof rule for reasoning about the termination of the **while**-loop by giving two ingredients: the invariant (an assertion that holds before and after the loop and also before and after the loop body) and the variant (a term denoting a value that must decrease each iteration, and it must be shown that this term is larger than or equal to zero assuming the invariant).

There may be a problem with non-standard interpretations. For example, in the complex interpretation, what does the variant denote? It may no longer express a quantity that ensures that the loop body is executed finitely many times. In fact, we already know that every unreachable element is larger than 0, and every unreachable element has infinitely many predecessors: so requiring that during the loop body the variant decreases no longer yields an argument that the computation must be finite. Yet, this is not a problem, for we can always ignore non-standard interpretations. The proof rule is sound: interpreting total correctness with respect to the standard interpretation works fine. In that interpretation, the variant expresses that, before executing the loop, we can predict the maximum number of iterations that will be taken based on the values of program variables in the initial state.

Mirkowska and Salwicki suggest that the Collatz conjecture, formulated as questions of provability in Hoare logic as we did above, is not precise. They then start working on the level of the semantics in the complex interpretation, and restrict attention to those states in which the program variables have reachable values. However, here I disagree. That there exists non-standard interpretations is not an excuse to not give a proof in Hoare's logic: although such a proof can also be interpreted in a non-standard interpretation, there is always the standard interpretation one can look at.

What I thus expected to see was a clear description from which I can extract a proof in Hoare's total correctness logic. In particular, to perform such an extraction, I need to know the answers to the following questions:

- What is the loop invariant of S?
- Is the loop invariant expressible as an assertion?
- What is the loop variant of S?
- Is the loop variant expressible?

Unfortunately, the paper does not clearly give answers to these questions. When I read the paper, I was not able to verify that there actually is a proof. The paper presents many interesting ideas, though, but I feel that I am doing original research to see how the ideas presented are related to me trying to answer the questions above. I should not have the feeling I am doing original research when my task and only task is to verify a proof!

4 Conclusion and closing remarks

Unfortunately, I am unable to extract a proof from the pre-print paper by Mirkowska and Salwicki [7]. Thus, the Collatz conjecture still remains unsettled—at least in my mind—and thus remains unsettling. The lesson, if there is a need for one, is that non-standard interpretations of Hoare's logic exists, and may shed light on a problem from a different angle.

At some point during reading, I got the following intuitions. I share them here—just for fun—but these closing remarks may not be valuable to anyone and may only cause confusion.

If we revisit the computation sequence above, that yielded an infinite run, we see the following pattern (we call these the *even step* and the *odd step*):

$$(k + \frac{3^o}{2^e}\iota) \longrightarrow (\frac{k}{2} + \frac{3^o}{2^{e+1}}\iota) \text{ if } k \text{ is even,}$$
$$(k + \frac{3^o}{2^e}\iota) \longrightarrow ((3 \times k + 1) + \frac{3^{o+1}}{2^e}\iota) \text{ if } k \text{ is odd.}$$

For simplicity, assume k is a positive natural number, and o, e are natural numbers. Forgetting about the imaginary part, we see simply that the function f we defined in the introduction is applied on k. However, in the imaginary part, we see that the rational number is 'counting' how many times which part of the function was applied. If k was even, we increase the denominator from 2^e to 2^{e+1}. If k was odd, we increase the numerator from 3^o to 3^{o+1}.

Now, for any computation sequence on natural numbers of the domain of the standard interpretation, there exists infinitely many corresponding computation sequences on the complex numbers of the domain of the complex interpretation. We are free, so to say, to choose the imaginary part of the complex number. So, let us look at the complex interpretation. From now onward, my thoughts are a bit cloudy and imprecise—so we proceed in unclear territory.

Since in computations of the complex interpretation, 'we are free' to choose the imaginary part, could we treat the imaginary part as a ghost count of the number of times we see an even number and the number of times we see an odd number? If the even numbers tend to outnumber the odd numbers then we are going towards the exit—but at any moment we are doubtful, we may freely reset the counters. Even when we reach the destination, we may still continue (here $\longrightarrow^*$ means multiple steps, even or odd):

$$(1 + \iota) \longrightarrow^* (1 + \frac{3}{4}\iota) \longrightarrow^* (1 + \frac{9}{16}\iota) \longrightarrow^* (1 + \frac{27}{64}\iota) \longrightarrow \ldots$$

Here the denominator grows faster than the numerator: the imaginary part tends towards zero in the limit if we never reset the ghost counter back to ι. Does this intuition help? I do not know. Maybe one can show that there is a computation sequence in the complex interpretation in which the imaginary part tends to zero if and only if there is a corresponding computation sequence in the standard interpretation? If so, can we then also show that a computation sequence tends to zero if and only if it contains a number with 1 as real part?

The intuition here is that we want to jump ahead a potential computation sequence starting in some number, as if we could travel in time. The target of our jump is a number with the same real part—that would be evidence of a loop. If the loop also exists in the standard interpretation, then all the intermediary numbers must be natural numbers. Suppose we are at an arbitrary section of the computation sequence:

$$(f^0(k) + w_0\iota) \longrightarrow (f^1(k) + w_1\iota) \longrightarrow \ldots \longrightarrow (f^i(k) + w_i\iota)$$

where $f^0(k) = k$. The sequence $w_0, w_1, \ldots$ are the values of the ghost counters. We now want to reset the ghost counters back to ι, but keep the real part. Note that the imaginary part of ι is the fraction $3^0 \div 2^0 = 1$, meaning both even and odd counters are zero. After resetting the ghost counters, we have the section that starts with $(k + \iota)$:

$$(f^0(k) + \frac{3^{o_0}}{2^{e_0}}\iota) \longrightarrow (f^1(k) + \frac{3^{o_1}}{2^{e_1}}\iota) \longrightarrow \ldots \longrightarrow (f^i(k) + \frac{3^{o_i}}{2^{e_i}}\iota)$$

where the sequences $o_0, o_1, o_2, \ldots$ and $e_0, e_1, e_2, \ldots$ are running ghost counts with $o_0 = e_0 = 0$. For any index in the sequence $0 \leq j \leq i$, we have that $o_j + e_j = j$.

Is it possible to have $f^0(k) = f^i(k)$ but where the numerator grows faster than the denominator, i.e. $3^{o_i} > 2^{e_i}$? If so, can it then also have 1 occurring in the sequence? Note that 1 is never reached as a result of $3 \times x + 1$ (only x with zero real part would give 1, but that never occurs since $0 + w\iota$ is even and thus stays even), so from that point onward the denominator would dominate the numerator.

And what about the non-repeating computation sequences? Where there is no loop, but in which the imaginary part does tend to zero?

So many questions are still floating around in my mind, like butterflies...

References

[1] Paul J. Andaloro. The $3x + 1$ problem and directed graphs. *Fibonacci Quarterly*, 40(1):43–54, 2002.

[2] Ştefan Andrei, Manfred Kudlek, and Radu Ştefan Niculescu. Some results on the Collatz problem. *Acta Informatica*, 37:145–160, 2000.

[3] Ştefan Andrei and Cristian Masalagiu. About the Collatz conjecture. *Acta Informatica*, 35:167–179, 1998.

[4] Krzysztof R. Apt and Ernst-Rüdiger Olderog. Fifty years of Hoare's logic. *Formal Aspects of Computing*, 31:751–807, 2019.

[5] John H. Conway. On unsettleable arithmetical problems. *The American Mathematical Monthly*, 120(3):192–198, 2013.

[6] Grażyna Mirkowska and Andrzej Salwicki. *Algorithmic logic*. Springer, 1987.

[7] Grażyna Mirkowska and Andrzej Salwicki. Collatz conjecture becomes theorem. 2023. https://arxiv.org/abs/2310.13035.

[8] George Tourlakis. Gödel's first incompleteness theorem via the halting problem. In *Computability*, pages 265–280. Springer, 2022.

[9] David Xu and Dan E. Tamir. Pseudo-random number generators based on the Collatz conjecture. *International Journal of Information Technology*, 11(3):453–459, 2019.