Induction and Code for Collatz Conjecture or 3x+1 Problem

Wei Ren

School of Computer Science,
China University of Geosciences, Wuhan, China 430074
Hubei Key Laboratory of Intelligent Geo-Information Processing
(China University of Geosciences (Wuhan)), Wuhan, Hubei, China
Email: weirencs@cug.edu.cn

Abstract. Collatz conjecture (or 3x+1 problem) has not been proved to be true or false for about 80 years. The exploration on this problem seems to ask for introducing a totally new method. In this paper, a mathematical induction method is proposed, whose proof can lead to the proof of the conjecture. According to the induction, a new representation (for dynamics) called "code" is introduced, to represent the occurred $3*x+1$ and $x/2$ computations during the process from starting number to the first transformed number that is less than the starting number. In a code $3*x+1$ is represented by 1 and $x/2$ is represented by 0. We find that code is a building block of the original dynamics from starting number to 1, and thus is more primitive for modeling quantitative properties. Some properties only exist in dynamics represented by code, but not in original dynamics. We discover and prove some inherent laws of code formally. Code as a whole is prefix-free, and has a unified form. Every code can be divided into code segments and each segment has a form $\{10\}^{p\geq 0}0^{q\geq 1}$. Besides, p can be computed by judging whether $x \in [0]_2$, $x \in [1]_4$, or computed by $t = (x-3)/4$, without any concrete computation of $3*x+1$ or $x/2$. Especially, starting numbers in certain residue class have the same code, and their code has a short length. That is, $CODE(x \in [1]_4) = 100$, $CODE((x-3)/4 \in [0]_4) = 101000$, $CODE((x-3)/4 \in [2]_8) = 10100100$, $CODE((x-3)/4 \in [5]_8) = 10101000$, $CODE((x-3)/4 \in [1]_{32}) = 10101001000$, $CODE((x-3)/4 \in [3]_{32}) = 10101010000$, $CODE((x-3)/4 \in [14]_{32}) = 10100101000$. The experiment results again confirm above discoveries. We also give a conjecture on $x \in [3]_4$ and an approach to the proof of Collatz conjecture. Those discoveries support the proposed induction and are helpful to the final proof of Collatz conjecture.

Keywords: Collatz Conjecture; 3X+1 Problem; Computational Algebra; Algorithmic Number Theory;

1 Introduction

The Collatz conjecture is a mathematical conjecture that is first proposed by Lothar Collatz in 1937. It is also known as the 3x+1 conjecture, the Ulam

conjecture, the Kakutani's problem, the Thwaites conjecture, or the Syracuse problem [1].

Simply speaking, the conjecture can be stated as follows. Take any positive integer number x. If x is even, divide it by 2 to get $x/2$. If x is odd, multiply it by 3 and add 1 to get $3*x+1$. Repeat the process again and again. The Collatz conjecture is that no matter what the number (i.e., x) is taken, the process will always eventually reach 1.

Although the statement of this problem is simply, the exploration of the conjecture in existing methods seems to have experienced remarkable difficulties for further new advances. Thus, new research object (variables or functions) and corresponding methods may be asked for bringing new approaches to the proof of the conjecture.

The contributions and results of the paper are listed as follows:

1. We propose an induction method whose proof can lead to the proof of Collatz conjecture. A new variable called code is introduced to represent dynamics of Collatz transformation procedure and is taken as research object. Code is a new measurement for modeling properties and a more primitive element for exploring Collatz conjecture.
2. We prove that code as a whole called CODE is prefix-free. $x/2$ (denoted as 0) always occur after $3*x+1$ (denoted as 1). That is, 10 always consecutively occurs in CODE. We prove that CODE has a unified format. Any code can be divided into code segments and each segment has the form $\{10\}^p0^q$, $p, q \in \mathbb{N}$. Dynamics represented by code is a building block of original dynamics from starting number to 1.
3. We also discover that the code of starting numbers in certain residue class is the same. Especially, ceratin residue classes have codes with short length (i.e., ≤ 7), when and only when $x \in [0]_2 \cup [1]_4$, and especially $(x-3)/4 = t \in [0]_4 \cup [2]_8 \cup [5]_8 \cup [1]_{32} \cup [3]_{32} \cup [14]_{32}$).
4. We prove that $CODE(\{x|x \in [1]_4\}) = 100$, $CODE(\{x|(x-3)/4 \in [0]_4\}) = 101000$, $CODE(\{x|(x-3)/4 \in [2]_8\}) = 10100100$, $CODE(\{x|(x-3)/4 \in [5]_8\}) = 10101000$, $CODE(\{x|(x-3)/4 \in [1]_{32}\}) = 10101001000$, $CODE(\{x|(x-3)/4 \in [3]_{32}\}) = 10101010000$, $CODE(\{x|(x-3)/4 \in [14]_{32}\}) = 10100101000$.
5. The analysis results on short code can shorten the verification time for Collatz conjecture to 10% as before. That is, only 10% numbers are left and need to be verified. We also point out an approach and a conjecture on $t = (x-3)/4 \in [i]_m$ whose proof can lead to the proof of Collatz conjecture.

Next, we state the Collatz conjecture ($3x+1$ conjecture) more formally. For any positive integer x, after finite times of computations of Computation I or Computation II, x will always become 1. The Computation I is $x \Leftarrow 3*x+1$ when x is odd, which is called Triple Plus One, denoted as TPO; The Computation II is $x \Leftarrow x/2$ when x is even, which is called Half, denoted as H. We call these two types of computation (TPO and H) as Collatz Transformation. Specifically, the Collatz conjecture can be formulated as follows:

Definition 1. *Collatz Transformation (denoted as $CT(x)$). $\forall x \in \mathbb{N}$, $CT(x) = TPO(x) = 3*x+1$ when x is odd, and $CT(x) = H(x) = x/2$ when x is even.*

Here, $\mathbb{N}$ is a set of positive integer numbers. That is, $\mathbb{N} = \{a | a \in \mathbb{Z}, a \geq 1\}$. Note that, in this paper we call positive integer numbers as natural numbers.

That is,

$$CT(x) = \begin{cases} TPO(x) = 3 * x + 1 & (TPO) \quad (x \in [1]_2) \\ H(x) = x/2 & (H) \quad (x \in [0]_2) \end{cases} \tag{1}$$

Here, $[1]_2$ denotes odd natural numbers, $[0]_2$ denotes even natural numbers. That is, $[1]_2 = \{a | a \mod 2 = 1, a \in \mathbb{N}\}$, $[0]_2 = \{a | a \mod 2 = 0, a \in \mathbb{N}\}$.

$TPO(x)$ can be simply denoted as TPO, and $H(x)$ can be simply denoted as H.

Definition 2. *The Collatz Conjecture.* $\forall x \in \mathbb{N}$. *After finite times of Collatz Transformation* $x \Leftarrow CT(x)$, x *will become 1. (Here "*$\Leftarrow$*" is assignment symbol and "*$x \Leftarrow y$*" means to assign value* y *to* x*.)*

Example 1. The dynamics consisting of each Collatz Transformation from a starting value (at head, e.g., 7) to 1 (at rear) are as follows:

(1) $3 \to 10 \to 5 \to 16 \to 8 \to 4 \to 2 \to 1$;

(2) $5 \to 16 \to 8 \to 4 \to 2 \to 1$

(3) $7 \to 22 \to 11 \to 34 \to 17 \to 52 \to 26 \to 13 \to 40 \to 20 \to 10 \to 5 \to 16 \to 8 \to 4 \to 2 \to 1$.

2 Results

2.1 Induction

To simplify the statement for conjecture, we define "Returnable" as follows.

Definition 3. *Returnable.* $x \in \mathbb{N}$ *is Returnable, if* x *will become 1 after finite times of Collatz Transformation* $x \Leftarrow CT(x)$.

If x is Returnable, we denote it as $x \in \mathcal{RTN}$ for simplicity.

To verify (or prove further) Collatz conjecture, the basic idea in this paper is mathematical induction.

The Collatz conjecture is True, if following induction can be proved.

Induction (for Collatz conjecture):

(1) $x = 1 \in \mathcal{RTN}$ (see Example 1(1)).

(2) If $x \leq k$ $(x, k \in \mathbb{N})$ is Returnable, $x = k + 1$ will be Returnable. That is, if $x \in \mathcal{RTN}$, $(x \leq k, x, k \in \mathbb{N})$, $x = k + 1 \in \mathcal{RTN}$ can be proved.

In shorthand, the induction is as follows:

$$x \leq k \in \mathcal{RTN} \Rightarrow x = k + 1 \in \mathcal{RTN},$$

where $x, k \in \mathbb{N}$.

If we call $x = k + 1$ as a "starting number", and a number after Collatz Transformation (i.e., $CT(x)$) as a "transformed number". It is worth to note

that we ONLY need to check whether transformed number is less than starting number. Once transformed number is less than starting number, the starting number will be Returnable (i.e., $x = k + 1 \in \mathcal{RTN}$) due to the induction assumption ($x \leq k \in \mathcal{RTN}$).

Fig. 1 illustrates the rationale in our Induction.

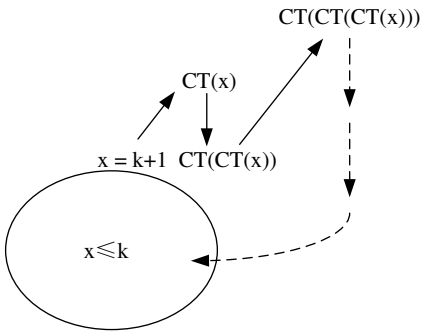

Fig. 1. Induction Rationale. Once transformed number is smaller than the starting number, the starting number will be called "Returnable". That is, once $CT(CT(...CT(x))) < x$, $x \in \mathcal{RTN}$

Proposition 1. *If the Induction can be proved, Collatz Conjecture is True.*

Proof. Straightforward. □

In the following of the paper, we only discuss natural number x, thus we omit $x \in \mathbb{N}$ for simplicity. We will also omit "*" for multiplication representations (e.g., $3x + 1$ is $3 * x + 1$ for short).

Besides, it is trivial to check that $k = 1, x = k = 1 \in \mathcal{RTN}, x = k + 1 = 2 \in \mathcal{RTN}$ (see Example 1(1)).

Proposition 2. *If in the Induction k is odd, the Induction is trivial to be proved.*

Proof. Suppose $x \leq k \in \mathcal{RTN}, k > 1$. k is odd, thus $k+1$ is even. That is, when $x = k + 1$, $CT(x) = (k + 1)/2 < k$. Thus, $x = k + 1 \in \mathcal{RTN}$. □

Therefore, for the proof of Induction we only need to prove the case that k is even.

Proposition 3. *If in the Induction k is even with $k = 4t, t \in \mathbb{N}$, the Induction is straightforward to be proved.*

Proof. Suppose $x \leq k = 4t$ is Returnable. That is, $x \leq k = 4t \in \mathcal{RTN}, t \in \mathbb{N}$. Thus, $k + 1 = 4t + 1$. Next, let's check whether $x = k + 1 \in \mathcal{RTN}$. $CT(k + 1) = (3(4t + 1) + 1) = 12t + 4 \in [0]_2, CT(CT(x)) = (12t + 4)/2 = 6t + 2 \in [0]_2, CT(CT(CT(x))) = (6t + 2)/2 = 3t + 1 \leq 4t$. Thus, $x = k + 1 \in \mathcal{RTN}$. □

Therefore, we only need to prove the case that k is even with $k = 4t + 2, t \in \mathbb{Z}, t \geq 0$ in the Induction due to above propositions (proposition 2 and proposition 3).

Note that, above analysis on induction should be distinguished with following discussion on reduced induction.

Induction (**Reduced Induction** for Collatz conjecture):

(1) $x = 1, 2, 3, 4, 5, 6 \in \mathcal{RTN}$ (see Example 1(1)).

(2) If $x \leq k = 2t+1$ $(x, k, t \in \mathbb{N})$ is Returnable, $x = k+1$ will be Returnable. That is, if $x \in \mathcal{RTN}$, $(x \leq k = 2t + 1, x, k, t \in \mathbb{N})$, $x = k + 1 \in \mathcal{RTN}$ can be proved.

(3) If $x \leq k = 4t$ $(x, k, t \in \mathbb{N})$ is Returnable, $x = k + 1$ will be Returnable. That is, if $x \in \mathcal{RTN}$, $(x \leq k = 4t, x, k, t \in \mathbb{N})$, $x = k+1 \in \mathcal{RTN}$ can be proved.

(4) If $x \leq k = 4t+2$ $(x, k, t \in \mathbb{N})$ is Returnable, $x = k+1$ will be Returnable. That is, if $x \in \mathcal{RTN}$, $(x \leq k = 4t + 2, x, k, t \in \mathbb{N})$, $x = k + 1 \in \mathcal{RTN}$ can be proved.

As Step (2) and Step (3) can be proved due to Proposition 2 and 3, respectively. In shorthand, the reduced version of the induction is as follows:

$$x \leq k = 4t + 2 \in \mathcal{RTN} \Rightarrow x = k + 1 \in \mathcal{RTN},$$

where $x, k, t \in \mathbb{N}$.

Remark 1. Reduced version of Induction for $3x + 1$ conjecture reduces the verification numbers by 75% (we only need to check $1/4$ natural numbers as before, so total checking time for Collatz conjecture is accelerated at least 75%). (See Appendix I program 1.)

Note that, Reduced Induction is different from the Induction that x is in different partitions of $\mathbb{N}$. To explain this, we hereby additionally give following loosen (or weaker version) conjectures for Collatz conjecture.

Conjecture 1. Collatz conjecture on $x \in [0]_2$. $\forall x = 2t, x, t \in \mathbb{N}$. After finite times of Collatz Transformation $x \Leftarrow CT(x)$, x will become 1.

Conjecture 2. Collatz conjecture on $x \in [1]_4$. $\forall x = 4t + 1, x \in \mathbb{N}, t \in \mathbb{Z}, t \geq 0$. After finite times of Collatz Transformation $x \Leftarrow CT(x)$, x will become 1.

Conjecture 3. Collatz conjecture on $x \in [3]_4$. $\forall x \in \mathbb{N}, x = 4t + 3, t \in \mathbb{Z}, t \geq 0$. After finite times of Collatz Transformation $x \Leftarrow CT(x)$, x will become 1.

Proposition 4. *If $3x + 1$ conjecture on $x \in [0]_2$, on $x \in [1]_4$ and on $x \in [3]_4$ are ALL True, Collatz conjecture is True.*

Proof. Straightforward. □

Above loosen conjectures provide an approach for the proof of Collatz conjecture. However, the proof for weaker version conjectures seems to be not easier than Collatz conjecture itself.

For example, the Induction for $3x + 1$ conjecture on $x \in [0]_2$ is as follows:

Induction (for Collatz conjecture on $x \in [0]_2$):

(1) $x = 2 \in \mathcal{RTN}$ (see Example 1(1)).

(2) If $x \leq k = 2t$ $(x, k, t \in \mathbb{N})$ is Returnable, $x = 2(t+1) = k+2$ will be Returnable. That is, if $x \in \mathcal{RTN}$, $(x \leq k = 2t, x, t \in \mathbb{N})$, $x = k+2 \in \mathcal{RTN}$ can be proved.

In shorthand, the induction is as follows:

$$x \leq k = 2t \in \mathcal{RTN} \Rightarrow x = k+2 \in \mathcal{RTN},$$

where $x, k \in \mathbb{N}$.

Next, we try to prove this Induction to explain why it seems not to be easier than original Induction.

Suppose $x \leq 2t \in \mathcal{RTN}$, where $t \in \mathbb{N}$. Next, try to prove $x = 2t+2 \in \mathcal{RTN}$. As $x = 2t+2 \in [0]_2$, $CT(x) = H(x) = (2t+2)/2 = t+1 \leq 2t$.

(1) If $t+1 \in [0]_2$, according to Induction condition $x \leq 2t \in \mathcal{RTN}$, we have $t+1 \in \mathcal{RTN}$. Thus, $x = 2t+2 \in \mathcal{RTN}$.

(2) If $t+1 \notin [0]_2$, $(3(t+1)+1)/2 = (3t+4)/2 = \frac{3}{2}t+2$.

(2.1) If $t \in [0]_4$, $\frac{3}{2}t+2 \in [0]_2$. Check when $\frac{3}{2}t+2 \leq 2t$, we have $t \geq 4$. Thus, $\frac{3}{2}t+2 \in [0]_2 \cap \frac{3}{2}t+2 \leq 2t$, when $t \geq 4, t \in [0]_4$. $\frac{3}{2}t+2 \in \mathcal{RTN}$. Thus, $x = 2t+2 \in \mathcal{RTN}$.

(2.2) If $t \in [2]_4$, we have $\frac{3}{2}t+2 \in [1]_2$. $3(\frac{3}{2}t+2)+1)/2 = (9t+14)/4$. It depends on $t \in [2]_8$ or $t \in [6]_8$ whether $(9t+14)/4 = 2t+3+(t+2)/4 \in [0]_2$ or not. Thus, to judge whether $(9t+14)/4 \in \mathcal{RTN}$ should be discussed further according to the partition of $t \in [2]_4$.

From above discussion, we can see that the proof for different partition of $x \in \mathbb{N}$ in weaker version conjectures is not easier than the proof of the Reduced Induction. Thus, we concentrate on Reduced Induction.

Definition 4. *Starting Number. It is the number to be checked whether it is Returnable.*

Definition 5. *Transformed Number. It is the number returned by Collatz Transformation during checking processes (dynamics) on a Starting Number.*

For example, in Example 1, starting numbers are $3, 5, 7$ in each item, and transformed numbers are y in all "$x \to y$" (on the right of "$\to$").

Moreover, if a computing algorithm or program is created to verify whether a given starting number (sn) is Returnable (i.e., $sn \in \mathcal{RTN}$), we only need to output the dynamics of its Collatz Transformations (i.e., $x \Leftarrow CT(x)$) until transformed number (tn) is less than the starting number (i.e., $tn < sn$). Of course, the checking sequence begins from a smaller starting number to a larger starting number. Obviously, this technique (until $tn < sn$ instead of $tn = 1$) further shortens the verification time for Collatz conjecture.

2.2 Code for Representing Dynamics

In this section, we define a new concept called code for representing dynamics during the verification of Collatz conjecture, which is a string consists of 1 representing "$3 * x + 1$" and 0 representing "$x/2$".

Notations.

(1) The Collatz transformation is denoted as "1" and "0". That is, $TPO(x) = 3*x+1$ is denoted as 1 and $H(x) = x/2$ is denoted as 0. For easily remembering and understanding, we may sometime call TPO as "Up" (because $TPO(x) = 3*x+1 > x$) and H as "Down" (because $H(x) = x/2 < x$).

(2) Dynamics. It consists of a serial of occurred Collatz transformations, and represents the process from a starting number to the *first* transformed number that is less than the starting number. For example, the original dynamics of 5 (recall Example 1 (2)) is TPO, H, H, H, or 1000. The "truncated" dynamics from starting number to the first transformed number (i.e., 4) is TPO, H, H, or 100. (We will prove that "truncated" dynamics is more primitive than original dynamics. Besides, our induction concentrates on "truncated" dynamics.)

(3) Code. It is a string consisting of 1 and 0 that represents the "truncated" dynamics of a starting number. For example, code for 5 is 100.

(4) Here for the convenience of presentation, we denote code for x as $CODE(x)$. $CODE : x \to y$, where $x \in \mathbb{N}$ is a starting number, and y is a bit string consisting of 1 and 0.

Note that, $CODE(x)$ is introduced as a notation mainly for representing code for x; We do not assume the existence of $CODE(x)$ for $\forall x \in \mathbb{N}$, which is exactly what Collatz conjecture wants to prove. $CODE(x)$ represents the dynamics of x that is already known or outputted by our computer program. We output codes for all $x \in [3, 99999999]$ (see Appendix and supplement information). We observe and analyze those outputted codes, discover and conjecture some inherent laws in codes. Such laws are proved as theorems and are held for all natural numbers.

Besides, Collatz transformations (i.e, $TPO(\cdot)$ and $H(\cdot)$) are both functions. Thus, given a starting number, transformed number after each Collatz transformation is unique. A serial of all transformed numbers is unique. The occurred transformation type (i.e., 1 or 0) is thus unique. Thus, code for a starting number is unique.

Definition 6. *Function $CODE(\cdot)$. $CODE : x \to y$ takes as input $x \in \mathbb{N}$ and outputs a bit string $y = \{0,1\}^n$, $n = |y|$, where $|\cdot|$ returns the length of a bit string. y represents occurred Collatz Transformations during the processes from a starting number x to the first transformed number that is less than x.*

Example 2. Dynamics for some $x = 4t+3, t \in \mathbb{N}$ are as follows:

(1) $107 \to 322 \to 161 \to 484 \to 242 \to 121 \to 364 \to 182 \to 91 < 107$. That is, the dynamics is "10100100", or $CODE(107) = 10100100$.

(2) $115 \to 346 \to 173 \to 520 \to 260 \to 130 \to 65 < 115$. In shorthand, the dynamics is "101000" or $CODE(115) = 101000$.

Remark 2. (1) If $n = |y|$ is finite for x, $CODE(x)$ exists; If $CODE(x)$ exists, n is finite.

(2) If $CODE(x)$ exists, $x \in \mathcal{RTN}$; If $x \in \mathcal{RTN}$, $CODE(x)$ exists.

(3) If Collatz conjecture is true, $\forall x \in \mathbb{N}$, $CODE(x)$ exists; If $\forall x \in \mathbb{N}$, Collatz conjecture is true.

(4) If the length of the code y is finite, the computer program for computing $CODE(\cdot)$ can be terminable; If the computer program for computing $CODE(\cdot)$ can be terminable, the length of the code y is finite. (The computer program, algorithm and outputted code are provided in appendix and supplement information.)

Indeed, without the knowledge of proposed induction and from a viewpoint of experimental mathematics, the proposal and analysis of code are also beneficial and necessary.

Remark 3. Indeed, code for all $x = 4t + 3 \in [3, 99999999]$ are outputted by our computer program. We verify Collatz conjecture from a smaller starting number to a larger starting number one by one. That is, if starting number is x and x can return to 1 after finite Collatz transformations, next starting number is $x+1$. Thus, in the verification of a larger starting number (i.e., $x+1$), if current transformed number is already less than starting number (i.e., $< x + 1$), the verification procedure can stop. The reason is that the transformed number can be looked as a verified starting number, and this verified starting number can return to 1. For example, in Example 1 (3), when transformed number is 5, the verification process stops. (The reason is that 5 has already been verified previously).

The computer program outputs all dynamics during the process from a starting number to a transformed number that is smaller than the starting number ("truncated" dynamics), instead of from a starting number to 1 ("un-truncated" dynamics or original dynamics). Note that, such treatment for recording such "truncated" dynamics has following advantages.

1. It will remove the redundance in "un-truncated" dynamics and help to discover inherent laws in codes. In fact, certain laws only exist in such "truncated" dynamics.
2. The "un-truncated" dynamics can be looked as the combination of finite "truncated" dynamics, thus "truncated" dynamics is more primitive element for exploration.
3. It will save the verification time largely and avoid the large amount of redundant verification.

E.g., Example 1 (4) ("un-truncated" dynamics) can be looked as "truncated" dynamics of 7 combines "truncated" dynamics of 5.

As a review of previous conclusions (e.g., Proposition 3), following example is given. It is straightforward to compute that the dynamics for $x = 4t+1, t \in \mathbb{N}$ is "100".

Example 3. The dynamics for $x = 4t+1, t \in \mathbb{N}$ is as follows (namely, $CODE(x) = 100$):

(1) $5 \to 16 \to 8 \to 4 < 5$;

(2) $101 \to 304 \to 152 \to 76 < 101$;

Proposition 5. *$CODE(\{x|x = 4t + 1, t \in \mathbb{N}\}) = 100$. That is, the dynamics for $x = 4t + 1 (t \in \mathbb{N})$ is "100" (namely, "TPO, H, H" or "Up, Down, Down").*

Proof. Straightforward. $x = 4t + 1$ is odd, $3x + 1 = 3(4t + 1) + 1 = 12t + 4 = 4(3t + 1) \in [0]_4$. Thus, double "Down" ("Down" is denoted as "0") occur intermediately after one "Up" (denoted as "1"). $3t + 1 < 4t + 1 = x$, thus the dynamics is terminated. The final dynamics is "100". That is, $CODE(x) = 100$, where $x = 4t + 1, t \in \mathbb{N}$. □

Above proposition includes all $x \in [1]_4, x \geq 5$ and does not include $x = 1$. It can be easily found that $CODE(1) = 100$, as $1 \to 4 \to 2 \to 1$. Therefore, together with Proposition 5, $CODE(\{x|x \in [1]_4\}) = 100$.

For the completeness, it is easy to check following fact.

Proposition 6. *$CODE(x) = 0$, where $x = 4t \cup x = 4t + 2, t \in \mathbb{N}$.*

Proof. Straightforward. It is due to Proposition 2. $x = 4t, 4t + 2$ is even, thus next transformation is H and $x/2 < x$. Thus, the dynamic is one "Down". □

Together with $CODE(2) = 0$, we have $CODE(\{x|x \in [0]_4 \cup [2]_4\}) = 0$.

In summary,

$$CODE(x) = \begin{cases} 100 & (x \in [1]_4) \\ 0 & (x \in [0]_4 \cup [2]_4) \end{cases} \tag{2}$$

If $x \in [0]_2 \cup [1]_4$, $x \in \mathcal{RTN}$. We thus only need to check the dynamics for x, where $x = 4t+3, t \in \mathbb{Z}$. Recall that in Example 1 (1), $CODE(3) = 101000$. That is, $t = 0, x = 4t + 3 = 3 \in \mathcal{RTN}$. Thus, $t \in \mathbb{N}$ (instead of $t \in \mathbb{Z}$) is considered in the following.

2.3 Properties of CODE

If all $CODE(x)$ is looked as a whole for the exploration of its general properties, we call it as CODE. That is, $CODE = \{code|\forall x \in \mathbb{N} \cap \mathcal{RTN}, code = CODE(x)\}$. We next analyze the properties of CODE, and further extend the properties of CODE to the dynamics for *all* nature numbers.

Theorem 1. *CODE must be prefix-free. That is, $\forall c \in CODE, c\|\{1, 0\}^* \notin CODE$.*

Proof. Straightforward. Suppose $CODE(x) = \{1, 0\}^n$. Due to the definition of $CODE(x)$, after the last transformation that is represented by the last bit of $CODE(x)$, the transformed number is less than the starting number x. If $CODE(x') = \{1, 0\}^n\|\{1, 0\}^m$ exists ($\|$ denotes concatenation), after the n-th transformation that is represented by the n-th bit of $CODE(x')$, the transformed number is **already** less than the starting number x'. Thus, $CODE(x')$ will be stopped at this bit, which means $CODE(x') = \{1, 0\}^n$. Therefore, CODE must be prefix-free. □

Proposition 7. *CODE must not be postfix-free.*

Proof. Intuitively, the transformed number of a starting number may equal another starting number. For example, the dynamics of 119 and 179 are as follows, thus, $CODE(119) = \underline{101000}$ is the postfix of $CODE(179) = 10\underline{101000}$. More specifically,

$119 \to 358 \to \mathbf{179} \to 538 \to 269 \to 808 \to 404 \to 202 \to 101,$

$\mathbf{179} \to 538 \to 269 \to 808 \to 404 \to 202 \to 101.$ □

Theorem 2. *If TPO occurs, H occurs intermediately after it. That is, "10" always consecutively (or together) occurs in any dynamics.*

Proof. If $x \in [1]_2$, $TPO(x) = 3 * x + 1 \in [0]_2$. Thus, the next Collatz transformation immediately after "TPO" must be "H". Therefore, H always occurs after a TPO transformation. That is, "10" always consecutively occurs. □

Note that, this conclusion holds for *any* dynamics without the concept of code.

Notation.

Due to Theorem 2, we introduce two notations as follows:

$I(x)$ is used to denote $H(TPO(x))$, where $TPO(x)$ is TPO transformation and $H(x)$ is H transformation. H always occurs after TPO, thus $H(TPO(x))$ can be written together (see Theorem 2), which is denoted by $I(x)$. That is, $I(x) = H(TPO(x))$.

Besides, we use $O(x)$ to denote $H(x)$ for easily remembering and better visualization (as $I(\cdot)$ looks like 1 and $O(\cdot)$ looks like 0).

For example, $CODE(x) = 100$, so dynamics sequences (i.e., transformation sequences) for x are TPO, H, and H. The transformation procedures are $TPO(x)$, $H(TPO(x))$, and $H(H(TPO(x)))$. It can also be simplified as $O(I(x)) = H(H(TPO(x)))$. Thus, "100" can be written as "IO". Also, $O(I(x))$ can be simply written as $IO(x)$. Besides, $100(x) = H(H(TPO(x))) = IO(x)$, where $100(\cdot)$ and $IO(\cdot)$ are a composite function.

Proposition 8. *$\forall x \in [3]_4, t = (x-3)/4 \in [0]_2$, the first five Collatz transformations are* 10100.

Proof. $x = 4t + 3 \in [1]_2$, $(3x+1)/2 = (12t+10)/2 = 6t+5 \in [1]_2$. $(3(6t+5)+1)/2 = 9t+8$. As $t \in [0]_2$, $9t+8 \in [0]_2$. Thus, the next transformation is "$x/2$". Thus, the first five Collatz transformations are "10100". □

Proposition 9. *$\forall x \in [3]_4, t = (x-3)/4 \in [1]_2 =$ 101010, the first six Collatz transformations are* 101010.

Proof. $x = 4t + 3 \in [1]_2$, $(3x+1)/2 = (12t+10)/2 = 6t+5 \in [1]_2$. $(3(6t+5)+1)/2 = 9t+8$. As $t \in [1]_2$, $9t+8 \in [1]_2$. Thus, the first six Collatz transformations are "101010". □

Proposition 10. *If $CODE(x \in [3]_4)$ exists, $CODE(x) = \{10\}^p \| ..., p \geq 2$.*

Proof. It can be obtained by Proposition 8 and Proposition 9.

Here put it in another way. Recall Eq. 2 $CODE(x \in [0]_2) = 0$, $CODE(x \in [1]_4) = \underline{10}0$, and CODE is prefix-free due to Theorem 1, thus "0" and "100" cannot be the head of $CODE(x \in [3]_4)$. Besides, "10" is always consecutively occurs in CODE due to Theorem 2. Therefore, if $CODE(x \in [3]_4)$ exists, it must be "$\underline{10100}\|...$" or "$\underline{101010}\|...$". Thus, $p \geq 2$. □

Sometimes, the current transformed number after some transformations is of interest. For starting number $x \in [3]_4$, observing following equations for current transformed number x (denoted as X_c) after consecutive $p \geq 2$ pairs of "10":

$$\begin{aligned} X_c &= (3...(3(3x+1)/2)+1)/2... + 1/2 \\ &= (\frac{3}{2})^p x + \frac{1}{2}((\frac{3}{2})^{p-1} + (\frac{3}{2})^{p-2} + ... + 1) \\ &= \frac{3}{2}(\frac{3}{2}(...\frac{3}{2}(\frac{3}{2}x + \frac{1}{2}) + \frac{1}{2}) + ... + \frac{1}{2}) + \frac{1}{2} \\ &= (\frac{3}{2})^p x + \frac{1}{2}(\frac{(\frac{3}{2})^p - 1}{\frac{3}{2} - 1}) \\ &= (\frac{3}{2})^p x + (\frac{3}{2})^p - 1 \\ &= (\frac{3}{2})^p (x+1) - 1 \end{aligned} \tag{3}$$

As $x = 4t+3, x, t \in \mathbb{N}$, above equation is also equivalent to following equation:

$$\begin{aligned} X_c = (\frac{3}{2})^p (x+1) - 1 &= (\frac{3}{2})^p (4t+3+1) - 1 \\ &= (\frac{3^p}{2^{p-2}})(t+1) - 1. \end{aligned} \tag{4}$$

As $X_c \in \mathbb{N}$, two cases will be occurred as follows:
Case I: $(t+1) \in [1]_2$, thus $p = 2$;
Case II: $(t+1) \in [0]_2$, i.e., $\frac{t+1}{2^{p-2}} \in \mathbb{N}$, $t \in [2^{p-2} - 1]_{2^{p-2}}, p > 2$.
Above analysis again confirms Proposition 10.

2.4 CODE Has a Unified Form

We next prove a formal theorem. This theorem shows that CODE has a unified form. That is, head of each code in CODE is $\{10\}^p\|0\|..., p \in \mathcal{Z}, p \geq 0$, which is called $FORMAT$. Here $p \in \mathcal{Z}$ instead of $p \in \mathbb{N}$ is used for including the case $p = 0$, in which $x \in [0]_4 \cup [2]_4$ is tackled.

Theorem 3. *(CODE Format Theorem.)* $CODE(x) \in FORMAT = \{10\}^p\|0\|...$, *where* $p \in \mathcal{Z}, p \geq 0$. *Besides,*

$$p=\begin{cases} 0 & & x\in[0]_2,([0]_2=[0]_4\cup[2]_4) \\ 1 & & x\in[1]_4 \\ 2 & & x=4t+3,t\in[0]_2, \\ \alpha+2 & t+1=2^\alpha*A,A\in[1]_2,\alpha\in\mathbb{N} & x=4t+3,t\in[1]_2. \end{cases} \tag{5}$$

Proof. If $x\in[0]_2$, $CODE(x)=0\in FORMAT$ and $p=0$.

If $x\in[1]_4$, $CODE(x)=100=\{10\}\|0\in FORMAT$ and $p=1$.

Next, we concentrate on $x\in[3]_4$. Let $x=4t+3,t\in\mathbb{N}$.

(1) Case I: $t\in[0]_2$.

As $x\in[3]_4\subset[1]_2$, $I(\cdot)$ is conducted consequently.

$X_c\Leftarrow I(x)=(3x+1)/2=(3(4t+3)+1)/2=(12t+10)/2=6t+5\in[1]_2$, thus transformation $I(\cdot)$ is conducted consequently.

$X_c\Leftarrow I(X_c)=(3(6t+5)+1)/2=(18t+16)/2=9t+8\in 9[0]_2+8=[0]_2$. Thus, $O(\cdot)$ is conducted consequently.

Therefore, $CODE(x)=I^2O\|...=\{10\}^2\|0\|...\in FORMAT$.

(2) Case II: $t\in[1]_2$.

As $x\in[3]_4\subset[1]_2$, $I(\cdot)$ is conducted consequently.

$X_c\Leftarrow I(x)=(3x+1)/2=(3(4t+3)+1)/2=(12t+10)/2=6t+5\in[1]_2$, thus $I(\cdot)$ is conducted consequently.

$X_c\Leftarrow I(X_c)=(3(6t+5)+1)/2=(18t+16)/2=9t+8\in[1]_2$. Thus, $I(\cdot)$ is conducted consequently.

$X_c\Leftarrow I(X_c)=(3(9t+8)+1)/2=(27t+25)/2$. It depends on t (more specifically, $t\in[1]_4$ or not) whether $(27t+25)/2$ is even or odd because of following observations:

$(27t+25)/2\in[0]_2$
$\Leftrightarrow(27t+25)/2=2K_1,K_1\in\mathbb{N}$
$\Leftrightarrow 27t+25=4K_1$
$\Leftrightarrow 27t=4K_2+3,K_2\in\mathbb{N}$
$\Leftrightarrow 3t=4K_3+3,K_3\in\mathbb{N}$
$\Leftrightarrow t=4K_4+1,K_4\in\mathbb{N}$
$\Leftrightarrow t\in[1]_4$.

Thus, if $t\in[1]_4$, $X_c=(27t+25)/2$ is even and $O(\cdot)$ will occur consequently. Thus, the transformed number after $O(\cdot)$ is $X_c=(27t+25)/4=(27[1]_4+25)/4=([3]_4+[1]_4)/4=[0]_4/4$.

Otherwise, if $t\in[3]_4$, X_c is odd and $I(\cdot)$ will occur consequently. Thus, the transformed number after $I(\cdot)$ is $X_c\Leftarrow I(X_c)=(3(27t+25)/2+1)/2=(81t+77)/4\in(81[3]_4+77)/4=([3]_4+[1]_4)/4=[0]_4/4$. The judgement on whether current $X_c\in[0]_2$ or not is undecidable (unless the range $t\in[1]_2=[1,3]_4=[1]_4\cup[3]_4$ is partitioned further).

For exploring more general result, we put it in another way. In general, due to Theorem 2, "10" occurs together. Suppose there p ($p>2,p\in\mathbb{N}$) pairs of "10" exist in the head of $CODE(x)$ (recall that here only Case II - $t\in[1]_2$ is considered). According to Eq. 3, current transformed number (after p times of $I(\cdot)$) X_c is as follows:

$$X_c = (3...(3(3x+1)/2+1)/2... + 1)/2$$
$$= (\frac{3}{2})^p(x+1) - 1 = (\frac{3}{2})^p(4t+3+1) - 1$$
$$= (\frac{3^p}{2^{p-2}})(t+1) - 1 \in \mathbb{N}.$$

Note that, it hereby implicitly includes a requirement (due to p times of consecutive $I(\cdot)$) as follows: all transformed numbers during the processes satisfy

$$X_c = (\frac{3^i}{2^{i-2}})(t+1) - 1 \in [1]_2,$$

where $2 \leq i \leq p-1, i \in \mathbb{N}$.

Besides, when $i = p$, $X_c = (\frac{3^i}{2^{i-2}})(t+1) - 1 \in [0]_2$, as only p consecutive $I(\cdot)$ occur. In other words, p can also be looked as the minimal value to let current transformed number $X_c \in [0]_2$ during such dynamics. Thus, we need to explore the requirement that for given t how to get p such that

$$(\frac{3^i}{2^{i-2}})(t+1) - 1 \in [1]_2, 2 \leq i \leq p-1 \cap (\frac{3^p}{2^{p-2}})(t+1) - 1 \in [0]_2.$$

We call this requirement as REQ.

Represent $t+1$ as $2^\alpha * A, A \in [1]_2, \alpha \in \mathbb{N}$. That is, $t+1 = 2^\alpha * A$. Obviously, this representation is unique. We thus need to prove that REQ is satisfied if and only if $p = \alpha + 2$. Note that, here p is indeed determined by α.

For $2 \leq i < p = \alpha + 2, i \in \mathbb{N}$, we have $\alpha + 2 - i > 0$ and

$$X_c = (\frac{3^i}{2^{i-2}})(t+1) - 1 = (\frac{3^i}{2^{i-2}}) * 2^\alpha * A - 1$$
$$= 3^i * 2^{\alpha-i+2} * A - 1 \in 3^i * [0]_2 * A - 1 = [1]_2.$$

When $i = p = \alpha + 2$, we have

$X_c = (\frac{3^p}{2^{p-2}})(t+1) - 1 = (\frac{3^p}{2^{p-2}}) * 2^\alpha * A - 1$
$= 3^p * 2^{\alpha-p+2} * A - 1 = 3^p * A - 1 \in 3^p * [1]_2 - 1 \in [0]_2$.

It is easy to see that $p = \alpha + 2$ is the minimal value to satisfy REQ. Thus, $p = \alpha + 2$ is the one and only one for REQ, as desired.

□

Corollary 1. *(t Determine p Theorem.) Given starting number $x = 4t + 3$, the number of consecutive "10" (denoted as p) is determined by t according to Eq. 6.*

$$p = \begin{cases} 2 & t \in [0]_2, \\ \alpha + 2 \quad \alpha = \log_2 \frac{t+1}{A} \in \mathbb{N}, A \in [1]_2 & t \in [1]_2. \end{cases} \tag{6}$$

Above theorem shows that p can be computed from t directly without conducting any concrete computation of $3 * x + 1$ and $x/2$ for counting times of "$\{10\}$".

For example, $CODE(7) = 10101001000 = \{10\}^3 0\{10\}^1 00 = IIIOIOO = I^3O\|IO^2$. $t = (7-3)/4 = 1 \in [1]_2$. $\alpha = \log_2(t+1)/A = \log_2(1+1)/1 = \log_2 2 = 1$. $p = \alpha + 2 = 1 + 2 = 3$.

Corollary 2.

$$CODE(x = 2^n - 1, n \in \mathbb{N}) = \{10\}^n\|0\|....$$

Proof. Suppose $CODE(x) = \{10\}^p\|0\|...$, due to Theorem 3.

(1) If $n = 1$, $x = 2-1 = 1 \in [1]_4$. Thus, $p = 1 = n$. Indeed, $CODE(x = 1) = \{10\}^1\|0$.

(2) If $n \in \mathbb{N}, n \geq 2$, thus $x = 2^n - 1 \in [3]_4$.

(2.1) If $n = 2$, $x = 2^2 - 1 = 3$, $(x-3)/4 = 0 \in [0]_2$, $p = 2$. Thus, $p = n$. Indeed, $CODE(x = 3) = 101000 = \{10\}^2\|0\|...$

(2.2) If $n > 2$, $t = (x-3)/4 = (2^n - 1 - 3)/4 = 2^{n-2} - 1 \in [1]_2$. $t + 1 = 2^{n-2}$. $A = 1$. $\alpha = \log_2(t+1)/A = \log_2 2^{n-2} = n - 2$. Thus, $p = \alpha + 2 = n - 2 + 2 = n$.

Therefore, $CODE(x = 2^n - 1, n \in \mathbb{N}) = \{10\}^n\|0\|....$ □

(Indeed, Eq. 6 can be used to design an automata that takes as input x and outputs $CODE(x)$. We leave it as future work and will discuss it in another paper.)

Next corollary shows that double down or "00" must exist in each code, which means that the times of "down" are always more than "up".

Corollary 3. *(Double Down Must Occur.) Code segment "00" must occur in* $CODE(\{x|x = 4t + 3, t \in \mathbb{N}\}$. *That is,* $...00... \in CODE$.

Proof. It is straightforward due to Format Theorem (Theorem 3).

For more details or put it another way, observing Eq. 3 and Eq. 4, we have $X_c = (\frac{3}{2})^p(x+1) - 1 = (\frac{3}{2})^p(4t+3+1) - 1 = (\frac{3^p}{2^{p-2}})(t+1) - 1 \in \mathbb{N}$.

If $t \in [0]_2, t + 1 \in [1]_2$, $p = 2$. Thus, $X_c = 3^2(t+1) - 1 = 9t + 8 \in [0]_2$. Thus, next transformation will be "0". Thus, "00" occurs.

If $t \in [1]_2, t + 1 \in [0]_2$. Suppose $t + 1 = 2^\alpha * A, \alpha \in \mathbb{N}, A \in \mathbb{N}, A \in [1]_2$. In this case, $p - 2 = \alpha$ to make X_c an integer, as p is determined by α, where $\alpha = \log_2(\frac{t+1}{A}), A \in [1]_2, A \in \mathbb{N}$. Therefore, $X_c = 3^{\alpha+2} * A - 1 \in [1]_2 * [1]_2 - 1 \in [0]_2$. Thus, "00" occurs. More specifically, $CODE(x) = \{10\}^{\alpha+2}\|0\|y$. It depends on $X_c \in [0]_4$ or not whether "y" is 0 or 1. □

Above corollary shows that we can divide $CODE(x)$ into different code segments that end with double or more '0's.

Corollary 4. $CODE(x)$ *satisfies*

$$CODE(x) = \begin{cases} O & 0 & x \in [0]_2, \\ IO & 100 & x \in [1]_4, \\ II\|... & 1010\|... & x \in [3]_4. \end{cases} \tag{7}$$

Proof. Straightforward. □

Next corollary gives more details on $CODE(\{x|x \in [3]_4\})$ that has a unified form as

$$\{10\}^{p_1 \geq 2} \| 0^{q_1 \geq 1} \| \{10\}^{p_2 \geq 1} \| 0^{q_2 \geq 1} \| ... \{10\}^{p_n \geq 1} \| 0^{q_n \geq 1}.$$

That is, each code consists of one or more segments, and each segment has a unified form as $\{10\}^p 0^q, p \geq 1, q \geq 1$.

Corollary 5. $CODE(x) = Segment_0 \| Segment^n, Segment_0 = I^{p_0} O^{q_0}, p_0 \geq 2, p_0 \in \mathbb{N}, q_0 \in \mathbb{N}, Segment \in \{Segment_i | Segment_i = I^{p_i} O^{q_i}, i, p_i, q_i \in \mathbb{N}\}$.

Proof. Straightforward. As current transformed number X_c after p consecutive pairs of "I" is even, X_c will become odd after one or more "O" transformations. When X_c becomes odd, next round of I transformations will occur. Above dynamics occur iteratively. Thus, each segment of codes has a unified form $I^{p_i} O^{q_i}$, where $p_i, q_i \in \mathbb{N}$.

It is worth to stress that why the first segment is listed solely in front of other segments, because the distinction between the first segment and other segments is that $p_0 \geq 2$, but $p_i \geq 1, i \in \mathbb{N}$. □

Obviously, in each code segments, more '0' than '1' exist. Thus, for all segments in codes, more '0' than '1'. Next corollary shows that D is more than U in each code (or during dynamics).

Corollary 6. *There exists more* 0 *than* 1 *in each code.*

Proof. Straightforward.

It is trivial when $x \in [0]_2 \cup [1]_4$, in which there exist one more 0 than 1.

Regarding $x \in [3]_4$, each segment in a code has a form $I^{p_i} O^{q_i}, p_i, q_i \in \mathbb{N}, i \geq 0, i \in \mathbb{N}$. In each segment, the number of '1' is p_i and the number of '0' is $p_i + q_i$. Thus, there exist more '0' than '1' in each segment. More specifically, the gap of more '0' than '1' is q_i. As all segments in a code have a unified form, the total number of '1' is more than the total number of '0'. More specifically, suppose $CODE(x) = I^{p_1} O^{q_1} \| I^{p_2} O^{q_2} \| ... \| I^{p_n} O^{q_n}, i, p_i, q_i \in \mathbb{N}$. Thus, $U = \Sigma_{i=1}^{i=n} p_i$, $D = \Sigma_{i=1}^{i=n} q_i + U$, and the gap between '0' and '1' is $D - U = \Sigma_{i=1}^{i=n} q_i$. □

CODE Format Theorem and its corollaries are all verified by our outputting CODE for $x \in [3, 99999999], x \in \mathbb{N}$, although it is not necessary. The source code and outputting code data will be provided as a supplement information, and also can be downloaded publicly.

Following proposition states the "truncated" dynamics is the building block of "un-truncated" dynamics. Thus, it is more primitive.

Proposition 11. *The "un-truncated" dynamics is the combination of "truncated" dynamics.*

Proof. Straightforward. The "un-truncated" dynamics means from a starting number (denoted as x) to 1. In this process, x must reach the first transformed number (denoted as a_1) that is less than x (i.e., $a_1 < x$). Thus, "truncated" dynamics (denoted as $CODE(x)$) is the first code segment of "un-truncated" dynamics. Next, start from a_1 or look a_1 as a new starting number, $CODE(a_1)$ will reach to a_2 where $a_2 < a_1$. Do it iteratively, we have $a_i < a_{i-1}$, $i = 2, ..., n$. Suppose $a_n = 2$, $CODE(a_n) = 0$ and transformed number is 1. Thus, "un-truncated" dynamics is $CODE(x)\|CODE(a_1)\|...\|CODE(a_n)$, which is the combination of "truncated" dynamics.

□

2.5 Numbers in Specific Residue Class Have Short Code

After observation of codes for $[3, 99999999]$ outputting by our computer program, we discover that certain specific numbers have code with short length. We formally prove those observations as theorems in the following.

Theorem 4. *$CODE(x) = 101000$, where $x \in [3]_4$ and $t = (x-3)/4 \in [0]_4$.*

Proof. $x = 4t + 3 \in [0]_4, 3x + 1 = 3(4t + 3) + 1 = 12t + 10 \in [0]_2$. As $t \in [0]_4$, $(3x+1)/2 = (12t+10)/2 = 6t+5 \in [1]_2$, $(3(6t+5)+1)/2 = 9t+8 \in [0]_4$. Thus, the next transformation should be "00". As $(9t + 8)/2/2 = 2.25t + 2 < 3t + 2 < 4t + 3 = x$, the code *ends* hereby with "101000". That is, $CODE(x) = 101000$. Certainly, it can also be written as $CODE(x) = IIOO$ or $CODE(x) = I^2O^2$ for convenience. □

Example 4. $115 \to 346 \to 173 \to 520 \to 260 \to 130 \to 65 < 115$; $CODE(115) = 101000, t = (115-3)/4 = 28 \in [0]_4$.

Theorem 5. *$CODE(x) = 10100100$, where $x \in [3]_4$ and $t = (x-3)/4 \in [2]_8$.*

Proof. $x = 4t + 3 \in [1]_2$, thus $3x + 1 \in [0]_2$. As $t \in [2]_8 \subset [0]_2$, $(3x + 1)/2 = (12t+10)/2 = 6t+5 \in [1]_2$. As $t \in [2]_8$, $(3(6t+5)+1)/2 = 9t+8 \in [2]_8 \subset [2]_4 = [0]_2 \cap \neg[0]_4$. Thus, the CODE segment "10100" occurs consequently. (It is easy to know that current transformed number is still larger than starting number due to $9t + 8 > 4t + 3$, thus further transformation will occur.)

As $t \in [2]_8$, $3\frac{9t+8}{2}+1 \in 3\frac{9*[2]_8+8}{2}+1 = 3*[2]_8/2+1 = 3*[1]_4+1 = [0]_4$. Thus the CODE segment "00" occurs consequently. (In the following, "*" in $a * [i]_m$ may be omitted and $a * [i]_m$ can be written as $a[i]_m$ for simplicity.)

$(3\frac{9t+8}{2} + 1)/2/2 = (13.5t + 13)/2/2 = (6.75t + 6.5)/2 = 3.375t + 3.25 = 4t + 3 + (0.25 - 0.625t) < 4t + 3 = x$, the code ends with "10100100". That is, $CODE(x) = 10100100$, where $x \in [3]_4$ and $t = (x-3)/4 \in [2]_8$. Certainly, it can also be written as $CODE(x) = IIOIO$ or $CODE(x) = I^2OIO$ for convenience.

□

Example 5. $11 \to 34 \to 17 \to 52 \to 26 \to 13 \to 40 \to 20 \to 10 < 11$; $CODE(11) = 10100100$, $(11-3)/4 = 2 \in [2]_8$.

Theorem 6. *$CODE(x) = 10101001000$, where $x \in [3]_4$ and $t = (x-3)/4 \in [1]_{32}$.*

Proof. $x = 4t + 3 \in [1]_2, t \in [1]_{32}$.
$I(x) = (3x+1)/2 = (12t+10)/2 = 6t + 5 \in [1]_2$.
$I^2(x) = (3(6t+5)+1)/2 = 9t + 8 \in 9[1]_{32} + 8 = [17]_{32} \subset [1]_2$.
$I^3(x) = (3(9t+8)+1)/2 \in (3[17]_{32}+1)/2 = [20]_{32}/2 = [10]_{16} \subset [0]_2$.

Therefore, CODE segment for above dynamics is "1010100".

As $[10]_{16}/2 = [5]_8 \subset [1]_2$,

$$I(O(I^3(x))) = (3[5]_8 + 1)/2 = [0]_8/2 = [0]_4.$$

Thus, double "0" will follow immediately. In summary, above dynamics to current transformed number is thus I^3OIOO.

Next, we check whether it is the final code (or dynamics is terminated at this transformed number) by checking whether current transformed number (denoted as X_c) is less than the staring number (i.e., x).

$X_c \Leftarrow I^3(x) = (3(9t+8)+1)/2 = (27t+25)/2 = 13.5t + 12.5$,
$X_c \Leftarrow O(X_c) = O(I^3(x)) = (13.5t + 12.5)/2 = 6.75t + 6.25$,
$X_c \Leftarrow I(X_c) = (I(O(I^3(x))) = (3(6.75t + 6.25) + 1)/2$
$= (20.25t + 18.75 + 1)/2 = (20.25t + 19.75)/2 = 10.125t + 9.875$,
$X_c \Leftarrow O(X_c) = O(I(O(I^3(x)))$
$= (10.125t + 9.875)/2 = 5.0625t + 4.9375 > 4t + 3 = x$,
$X_c \Leftarrow O(X_c) = O(O(I(O(I^3(x)))))$
$= (5.0625t + 4.9375)/2 = 2.53125t + 2.46875 < 3t + 3 < 4t + 3 = x$.

Therefore, dynamics ends with "10101001000" (dynamics is terminated). That is, $CODE(x) = 10101001000$.

Certainly, it can also be written as $CODE(x) = IIIOIOO$ or $CODE(x) = I^3OIO^2$ for convenience. □

Actually, we can use Eq. 3 for computing X_c after $I^3(x)$ to simplify above process in the proof. After $I^3(x)$,
$X_c \Leftarrow (\frac{3}{2})^3(x+1) - 1 = 1.5^3x + 1.5^3 - 1 = 3.375x + 2.375$,
$X_c \Leftarrow O(X_c) = 0.5(3.375x + 2.375) = 1.6875x + 1.1875$,
$X_c \Leftarrow I(X_c) = 1.5(1.6875x + 1.1875) + 0.5 = 2.53125x + 2.28125$,
$X_c \Leftarrow O(X_c) = 0.5(2.53125x + 2.28125) = 1.265625x + 1.140625$,
$X_c \Leftarrow O(X_c) = 0.5(1.265625x + 1.140625) = 0.6328125x + 0.5703125 = x + (0.5703125 - 0.3671875x) < x$, as $x \geq 2$ due to $x \in [3]_4$.

Example 6. $135 \to 406 \to 203 \to 610 \to 305 \to 916 \to 458 \to 229 \to 688 \to 344 \to 172 \to 86 < 135$; $CODE(135) = 10101001000$, $(x-3)/4 = (135-3)/4 = 33 \in [1]_{32}$.

Following two theorems originally stem from our conjectures on the base of our observations. Here we formally prove them as two theorems.

Theorem 7. *$CODE(x) = 10100101000$, where $x \in [3]_4$ and $t = (x-3)/4 \in [14]_{32}$.*

Proof. $x = 4t + 3 \in [1]_2, t \in [14]_{32}$.
$I(x) = (3x+1)/2 = (12t+10)/2 = 6t+5 \in [1]_2$, thus I will follow consequently.
$I^2(x) = (3(6t+5)+1)/2 = 9t+8 \in 9[14]_{32}+8 = [6]_{32} \subset [0]_2$, thus O will follow.
$O(I^2(x)) = [6]_{32}/2 = [3]_{16} \subset [1]_2$, thus I will follow.
$I(O(I^2(x))) = (3[3]_{16}+1)/2 = [10]_{16}/2 = [5]_8 \subset [1]_2$, thus I will follow.
$I^2(O(I^2(x))) = (3[5]_8+1)/2 = [0]_8/2 = [0]_4$, thus double "$O$" (or "0") will follow.

Next, X_c will be checked for whether $X_c < x$ after above transformations.
$X_c \Leftarrow I^2(x) = 9t + 8$,
$X_c \Leftarrow O(X_c) = O(I^2(x)) = (9t + 8)/2 = 4.5t + 4$,
$X_c \Leftarrow I(X_c) = (I(O(I^2(x))) = (3(4.5t+4)+1)/2 = (13.5t+13)/2 = 6.75t+6.5$,
$X_c \Leftarrow I(X_c) = I^2(O(I^2(x))) = (3(6.75t + 6.5) + 1)/2 = (20.25t + 20.5)/2 = 10.125t + 10.25$,
$X_c \Leftarrow O(X_c) = O(I^2(O(I^2(x))) = (10.125t + 10.25)/2 = 5.0625t + 5.125 > 4t + 3 = x$,
$X_c \Leftarrow O(X_c) = O^2(I^2(O(I^2(x))) = (5.0625t + 5.125)/2 = 2.53125t + 2.5625 < 3t + 3 < 4t + 3 = x$.
Therefore, the dynamics ends with "10100101000". That is, $CODE(x) = 10100101000$.
Of course, it can also be written as $IIOIIOO$, $I^2OI^2O^2$. □

Theorem 8. *$CODE(x) = 10101010000$, where $x \in [3]_4$ and $t = (x-3)/4 \in [3]_{32}$.*

Proof. $x = 4t + 3 \in [1]_2, t \in [3]_{32}$.
$I(x) = (3x + 1)/2 = (12t + 10)/2 = 6t + 5 \in [1]_2$,
$I^2(x) = (3(6t + 5) + 1)/2 = 9t + 8 \in 9[3]_{32} + 8 = [3]_{32} \subset [1]_2$.
$I^3(x) = (3[3]_{32} + 1)/2 = [10]_{32}/2 = [5]_{16} \subset [1]_2$.
$I^4(x) = (3[5]_{16} + 1)/2 = [0]_{16}/2 = [0]_8$, thus triple "0" will follow.

$X_c \Leftarrow I^2(x) = 9t + 8$,
$X_c \Leftarrow I(X_c) = I^3(x) = (3(9t + 8) + 1)/2 = (27t + 25)/2 = 13.5t + 12.5$,
$X_c \Leftarrow I(X_c) = I^4(x) = (3(13.5t+12.5)+1)/2 = (40.5t+38.5)/2 = 20.25t+19.25$,
$X_c \Leftarrow O(X_c) = O(I^4(x))) = (20.25t + 19.25)/2 = 10.125t + 9.625$,
$X_c \Leftarrow O(X_c) = O^2(I^4(x))) = (10.125t+9.625)/2 = 5.0625t+4.8125 > 4t+3 = x$,
$X_c \Leftarrow O(X_c) = O^3(I^4(x))) = 2.53125t + 2.40625 < 3t + 3 < 4t + 3 = x$.
Therefore, the dynamics ends with "10101010000". That is, $CODE(x) = 10101010000$.
Of course, it can also be written as $IIIIOOO$ or I^4O^3. □

In summary, similar to Eq. 2, aforementioned codes that have short length are given in Eq. 8 as follows:

$$CODE(\{x|x = 4t + 3, t \in \mathbb{N}\}) = \begin{cases} 101000 & IIOO & t \in [0]_4, \\ 10100100 & IIOIO & t \in [2]_8, \\ 10101000 & IIIOO & t \in [5]_8, \\ 10101001000 & IIIOIOO & t \in [1]_{32}, \\ 10101010000 & IIIIOOO & t \in [3]_{32}, \\ 10100101000 & IIOIIOO & t \in [14]_{32}. \end{cases} \quad (8)$$

Note that, we call above codes have short length, because $\|CODE(x)\|$ is short, where $\|\cdot\|$ is the length of $CODE(x)$. The length is measured by the total number of "I" or "O". For example, $CODE(\{x|x \in [3]_4, t = (x-3)/4 \in [0]_4\}) = \|IIOO\| = 4$, $CODE(\{x|x \in [3]_4, t = (x-3)/4 \in [2]_8\}) = \|IIOIO\| = 5$.

In summary, Eq. 2 and Eq. 8 are both presented together in Eq. 9 as follows. It again verifies that CODE is prefix-free (recall Theorem 1). It also verifies that CODE is unique, as all intersection sets for x or t are empty.

$$CODE(x) = \begin{cases} 0 & O & 1 & x \in [0]_2, \\ 100 & IO & 2 & x \in [1]_4, \\ 101000 & IIOO & 4 & x = 4t+3, t \in [0]_4, \\ 10100100 & IIOIO & 5 & x = 4t+3, t \in [2]_8, \\ 10101000 & IIIOO & 5 & x = 4t+3, t \in [5]_8, \\ 10101001000 & IIIOIOO & 7 & x = 4t+3, t \in [1]_{32}, \\ 10101010000 & IIIIOOO & 7 & x = 4t+3, t \in [3]_{32}, \\ 10100101000 & IIOIIOO & 7 & x = 4t+3, t \in [14]_{32}. \end{cases} \tag{9}$$

We discover that Eq. 9 list all $\{x|\|CODE(x)\| \leq 7\}$.

Proposition 12. $\{x|\|CODE(x)\| \leq 7\} = \{x|x \in [0]_2 \cup [1]_4\} \cup \{x|x \in [3]_4, (x-3)/4 \in [0]_4 \cup [2]_8 \cup [5]_8 \cup [1]_{32} \cup [3]_{32} \cup [14]_{32}.\}$

Proof. If $\|CODE(x)\| = 3$, $CODE(x) = IIO$ due to Theorem 2 and Theorem 1. However, $IIO(x) = (3(3x+1)/2+1)/2/2 = (3(1.5x+1.5)+1)/4 = 4.5/4x + 5.5/4 = 1.125x + 1.375 > x$. Thus, $\|CODE(x)\| \neq 3$.

If $\|CODE(x)\| = 4$, $CODE(x) = IIOO$ due to Theorem 3. That is, there exists only one type of code for $\|CODE(x)\| = 4$. Similarly, we can prove that there exists two types of code for $\|CODE(x)\| = 5$ and three types of code for $\|CODE(x)\| = 7$. Besides, $\|CODE(x)\| = 6$ is impossible, as $IIIOIO(x) > x$, $IIIIOO(x) > x$ and $IIOIIO(x) > x$. □

The simpler proof relies on a result about the relation between the number of 1 and the number of 0 in codes, which is extensively discussed in our another paper.

Corollary 7. $\{x|x \in [0]_2 \cup [1]_4, t = (x-3)/4 \in [2]_8 \cup [5]_8 \cup [1]_{32} \cup [3]_{32} \cup [14]_{32}\} \in \mathcal{RTN}$.

Note that, for the proof of Collatz conjecture, we need to prove that $\forall x \in \mathbb{N}, x \in \mathcal{RTN}$.

Corollary 8. *According to Eq. 8 only* $1 - (1/4 + 2/8 + 3/32) = 1 - 19/32 = 13/32 = 40.625\%$ *for* $x \in [3]_4$ *needs to be checked (for being returnable). That is, verification time is shorten by* 59.375%.

Corollary 9. *According to Eq. 8 and Eq. 2, only* $1/4 * 13/32 = 13/128 = 10.15625\%$ *for* $x \in \mathbb{N}$ *needs to be checked. That is, verification time is shorten about* 90%.

In general, we give a conjecture as follows:

Conjecture 4. $t \in [i]_m$ conjecture. $\forall x = 4t + 3, t \in \mathbb{N}$, $\exists m, i,\ m \in \mathbb{N}, m \geq 4, 0 \leq i \leq m - 1, i \in \mathbb{Z}$, such that $CODE(\{x|x = 4t + 3, t \in [i]_m\}$ exists.

Proposition 13. *If* $t \in [i]_m$ *conjecture is true,* $\forall x \in [3]_4$, $CODE(x) = CODE(\{x|x = 4i + 3\}) = CODE(\{x|x = 4(km + i) + 3, k \in \mathbb{N}\})$.

Proposition 14. *If* $t \in [i]_m$ *conjecture is true, Collatz conjecture is true.*

Proof. If $t \in [i]_m$ conjecture is true, $\forall x = 4t+3, t \in \mathbb{N}$ will lead to $(x-3)/4 = t \in [i]_m$, which results in the existence of $CODE(x)$. Together with $CODE(\{x|x \in [0]_2\}) = 0$, $CODE(\{x|x \in [1]_4\}) = 100$, we have $\forall x \in \mathbb{N}$, $CODE(x)$ exists. Thus, Collatz conjecture is true. □

In other words, if we can discover the link between the existence of $CODE(x)$ and $(x - 3)/4 = t \in [i]_m$, and if we can prove $\bigcup\{t|t \in [i]_m\} = \mathbb{N}$, the proof of Collatz conjecture will be accomplished.

3 Conclusion

In this paper, we propose a mathematical induction method whose proof leads to the proof of Collatz conjecture. We define a new concept called code to represent the dynamics, which consists of occurred Collatz transformations in the process from a starting number to the first transformed number that is smaller than the starting number. Some inherent laws only exist in above "truncated" dynamics. Above "truncated" dynamics is more primitive than "un-truncated" dynamics, as "un-truncated" dynamics can be looked as the combination of "truncated" dynamics.

The analysis of code properties can help prove the Collatz conjecture. We discover that code as a whole called CODE is prefix-free. We also discover that starting numbers that in certain residue class have the same code, such as $(x - 3)/4 \in [0]_4 \cup [2]_8 \cup [5]_8$ and $(x - 3)/4 \in [1]_{32} \cup [3]_{32} \cup [14]_{32}$.

Moreover, we prove that codes for certain residue classes have short length ($\|CODE(x)\| \leq 7$). $CODE(x \in [1]_4) = 100$, $CODE((x - 3)/4 \in [0]_4) = 101000$, $CODE((x - 3)/4 \in [2]_8) = 10100100$, $CODE((x - 3)/4 \in [5]_8) = 10101000$, $CODE((x-3)/4 \in [1]_{32}) = 10101001000$, $CODE((x-3)/4 \in [3]_{32}) = 10101010000$, $CODE((x - 3)/4 \in [14]_{32}) = 10100101000$. Especially, we also prove that only those residue classes have short code length no more than 7.

We prove that code as a whole have a unified form. Every code can be divided into code segments and each segment has a form $\{10\}^p 0^q$.

Our discovery can shorten the verification time for Collatz conjecture to 10%. That is, only 10% numbers are left and need to be verified. Our analysis also provides foundation for designing an automata for verifying Collatz conjecture.

We also give a conjecture on $x \in [3]_4$, whose proof can lead to the proof of Collatz conjecture. We point out the possible approach for the proof of Collatz conjecture on the base of $CODE(\{x|x \in [3]_4, (x-3)/4 = t \in [i]_m\})$.

Those discoveries on codes instantiate the proposed induction, describe the properties of dynamics of occurred Collatz transformations, and are helpful to the final proof of Collatz conjecture via proposed induction. The experiment results for $x \in [3, 99999999]$ or even larger also confirm our discoveries, although all proofs for above conclusions are formal and sufficient enough for their soundness.

Acknowledgement

The research was financially supported by the National Natural Science Foundation of China (61170217).

References

1. Wikipedia. Collatz conjecture. *https://en.wikipedia.org/wiki/Collatz_conjecture*, Retrieved Sept. 10, 2016.

Appendix: Algorithm for outputting code

The major algorithm is listed as follows. Input is a starting number x; Output is $code(x)$.

```
   Data: x
   Result: code = CODE(x)
 1 a ⇐ x;
 2 while (a >= x) do
 3 |   if (a is odd) then
 4 |   |   a ⇐ 3 * a + 1;
 5 |   |   code ⇐ code‖'1';
 6 |   |   continue;
 7 |   end
 8 |   if (a is even) then
 9 |   |   a ⇐ a/2;
10 |   |   code ⇐ code‖'0';
11 |   |   if (a < x) then
12 |   |   |   return code;
13 |   |   end
14 |   |   continue;
15 |   end
16 end
```

Algorithm 1: $CODE(x)$

All codes for $x \in [3, 99999999]$ are provided as supplement information (some of them have big size over 100MB will be provided by personal web site or other public achieves). Source codes in ANSI C are provided as supplement information.