Proof of the Collatz conjecture using the Div sequence

Masashi Furuta
https://github.com/righ1113/collatzProof_DivSeq/
https://righ1113.hatenablog.com/
https://twitter.com/righ1113/

Abstract

We define the "Div sequence" that sets up the number of times divided by 2 in the Collatz operation. Using this and the "infinite descent", we prove the Collatz conjecture.

.
.

Part1 all odd numbers of multiples of 3 are represented by the Complete Div sequence

---

Chapter01 Introduction

Collatz conjecture

Consider the following operation on an arbitrary positive integer:

- If the number is even, divide it by two.
- If the number is odd, triple it and add one.

Now form a sequence by performing this operation repeatedly, beginning with any positive integer, and taking the result at each step as the input at the next.
The Collatz conjecture is: This process will eventually reach the number 1, regardless of which positive integer is chosen initially.

Let's call (3x+1) for odd x and divide by 2 to be ***(one) Collatz operation*** .
Let's call the number of Collatz operations ***Collatz value*** .

Div sequence & Complete Div sequence

Definition1-1 Div sequence

Set up the number of times divided by 2 by Collatz operations.
We call this ***Div sequence*** .
For example, in the case of 9, Collatz sequence are

```
9,28,14,7,22,11,34,17,52,26,13,40,20,10,5,16,8,4,2,1
```

(When it reaches 1 it will stop there) to Div sequence

```
[2,1,1,2,3,4]
```

is.

Definition1-2 Complete Div sequence

We name the Div sequence whose initial value is a multiple of 3 as ***Complete Div sequence*** .

```
9[2,1,1,2,3,4] is Complete Div sequence.
```

```
7[1,1,2,3,4] is Div sequence.
```

We can check only odd numbers of multiples of 3

We do not need to check even numbers

By dividing every even number by 2, we reach either odd number. Therefore, we can check only "whether all odd numbers will reach 1 with Collatz operation".

We can check only odd numbers of multiples of 3

For number x which is not divisible by 3, think backwards by Collatz reverse operation.
The remainder obtained by dividing x by 9 is one of 1, 2, 4, 5, 7, and 8, this

```
1*2^6≡1
2*2^5≡1
4*2^4≡1
5*2^1≡1
7*2^2≡1
8*2^3≡1 (mod 9)
```

like, By multiplying 2 by an appropriate number of times, dividing by 9 makes it possible to make it even one more surplus.
If you subtract 1 from this and divide by 3 it will be an odd multiple that is a multiple of 3.

Tracing back one Collatz operation from x, it is an odd number of multiples of 3.
If an odd number of multiples of 3 arrives at 1, x which operated odd numbers of multiples of 3 once in Collatz operation 1.
Therefore,

Theorem1-1

```
We need to check only "Does odd numbers of multiples of 3 reach 1 with Collatz
Operation?".
```

Chapter02 Star transformation

Definition2-1 Star transformation

Star transformation is defined.
Let us consider a mapping from the Complete Div sequence of length n to the Complete Div sequence of length n+1.
First, the Complete Div sequence of length n is represented by length n+1 with o added to the first term.
Next, modulo of the Collatz value x divided by 9,

- 3 ... stick A[6,-4] or B[1,-2]
- 6 ... stick C[4,-4] or D[3,-2]
- 0 ... stick E[2,-4] or F[5,-2]

If the original initial term becomes negative, G[+6] is done in advance.

```
・example
117≡0(mod 9) 117[5, 1, 2, 3, 4]
At this time, can transformation
E[2, -4] -> 9[2, (5-4), 1, 2, 3, 4] and
F[5, -2] -> 309[5, (5-2), 1, 2, 3, 4].
```

case	StarT 1	StarT 2
x ≡ 3 mod 9	A[6,-4] y=4x/3-7	B[1,-2] y=x/6-1/2
x ≡ 6 mod 9	C[4,-4] y=x/3-2	D[3,-2] y=2x/3-1
x ≡ 0 mod 9	E[2,-4] y=x/12-3/4	F[5,-2] y=8x/3-3
always	G[+6] y=64x+21	

The function expresses the change of the Collatz value.

All odd numbers of multiples of 3 are represented by the Complete Div sequence

Let's see how Collatz value changes with each Star transformation.

StarT	x	comment 1	comment 2
3 mod 9			
A[6,-4]So exclude the first term of the Div sequence less than 4			

StarT	x	comment 1	comment 2
A[6,-4] y=4x/3-7	3+9t	When t is an odd number, exclude because x is an even number	
	3+18t	(3(3+18t)+1)/2=5+27t	Except when t is even number
	21+36t	(3(21+36t)+1)/4=16+27t	Except when t is odd number
	21+72t	(3(21+72t)+1)/8=8+27t	Except when t is odd number
	21+144t	(3(21+144t)+1)/16=4+27t	Except when t is odd number
A map 21+288t to 【21+384t】	21+288t	4(21+288t)/3-7 = 21+384t	

B[1,-2]So exclude the first term of the Div sequence less than 2			
B[1,-2] y=x/6-1/2	3+9t	When t is an odd number, exclude because x is an even number	
	3+18t	(3(3+18t)+1)/2=5+27t	Except when t is even number
	21+36t	(3(21+36t)+1)/4=16+27t	Except when t is odd number
B map 21+72t to 【3+12t】	21+72t	(21+72t)/6-1/2 = 3+12t	

6 mod 9			
C[4,-4]So exclude the first term of the Div sequence less than 4			
C[4,-4] y=x/3-2	6+9t	When t is an even number, exclude because x is an even number	

StarT	x	comment 1	comment 2
	15+18t	(3(15+18t)+1)/2=23+27t	Except when t is even number
	33+36t	(3(33+36t)+1)/4=25+27t	Except when t is even number
	69+72t	(3(69+72t)+1)/8=26+27t	Except when t is odd number
	69+144t	(3(69+144t)+1)/16=13+27t	Except when t is even number
C map 213+288t to 【69+96t】	213+288t	(213+288t)/3-2 = 69+96t	
D[3,-2]So exclude the first term of the Div sequence less than 2			
D[3,-2] y=2x/3-1	6+9t	When t is an even number, exclude because x is an even number	
	15+18t	(3(15+18t)+1)/2=23+27t	Except when t is even number
	33+36t	(3(33+36t)+1)/4=25+27t	Except when t is even number
D map 69+72t to 【45+48t】	69+72t	2(69+72t)/3-1 = 45+48t	
0 mod 9			
E[2,-4]So exclude the first term of the Div sequence less than 4			
E[2,-4] y=x/12-3/4	9t	When t is an even number, exclude because x is an even number	
	9+18t	(3(9+18t)+1)/2=14+27t	Except when t is odd number

StarT	x	comment 1	comment 2
	9+36t	(3(9+36t)+1)/4=7+27t	Except when t is even number
	45+72t	(3(45+72t)+1)/8=17+27t	Except when t is even number
	117+144t	(3(117+144t)+1)/16=22+27t	Except when t is odd number
E map 117+288t to 【9+24t】	117+288t	(117+288t)/12-3/4 = 9+24t	
F[5,-2]So exclude the first term of the Div sequence less than 2			
F[5,-2] y=8x/3-3	9t	When t is an even number, exclude because x is an even number	
	9+18t	(3(9+18t)+1)/2=14+27t	Except when t is odd number
	9+36t	(3(9+36t)+1)/4=7+27t	Except when t is even number
F map 45+72t to 【117+192t】	45+72t	8(45+72t)/3-3 = 117+192t	
G[+6] y=64x+21	3+6t		
G map 3+6t to 【213+384t】	3+6t	64(3+6t)+21=213+384t	

We can see that all transformations are mapped from 3+6t to 3+6t'.

composite

```
G【213+384t】┓━[21+192t]   ━┓━[21+96t]  ━┓━[21+48t]  ━┓━[21+24t]
A【21+384t】 ┛  F【117+192t】┛  C【69+96t】┛  D【45+48t】┛
━[21+24t]  ━┓━[9+12t]    ━┓━[3+6t]☆☆☆
  E【9+24t】┛     B【3+12t】┛
```

Therefore,

Theorem2-1

```
All odd numbers of multiples of 3 are represented by the Complete Div sequence.
```

If this Complete Div sequence is all finite terms, Collatz conjecture is also true.

Part2 allDivSeq at level 0 is all finite terms

Chapter03 Extended Complete Div sequence

inspection

After star transformation, it is prohibited that the element of the Div sequence becomes 0 or negative,
What will happen if we admit this?
I will try two more.

- 9[2,1,1,2,3,4]
  ↓F[5,-2] y=8x/3-3
  21[5,0,1,1,2,3,4]
- Calculation of confirmation
  We follow the Div sequence from the opposite.
  When 1<-1<-2<-3<-4 is done, Collaz value is 7.
  (72^0-1)/3=2
  (22^5-1)/3=21 matching.
  .
  .
- 15[1,1,1,5,4]
  ↓C[4,-4] y=x/3-2
  3[4,-3,1,1,5,4]
- Calculation of confirmation
  We follow the Div sequence from the opposite.
  When 1<-1<-5<-4 is done, Collaz value is 23.
  (23*2^-3-1)/3=5/8
  ((5/8)*2^4-1)/3=3 matching.

Both are out of the rules of Collatz,
Calculation of (3x+1)/2^p is done.

confirmation

It is not possible in all cases.
We will check about Star transformation.

Trans	TFunc	before	after	treatment
3 mod 9				

Trans	TFunc	before	after	treatment
A[6,-4]	y=4x/3-7	x=3+9t	y=3(4t-1)	Prohibit t=0
B[1,-2]	y=x/6-1/2	x=3+9t	y=3t/2	Prohibit t:odd
6 mod 9				
C[4,-4]	y=x/3-2	x=6+9t	y=3t	no probrem
D[3,-2]	y=2x/3-1	x=6+9t	y=3(2t+1)	no probrem
0 mod 9				
E[2,-4]	y=x/12-3/4	x=9t	y=(3/4)(t-1)	Prohibit t-1 is not a multiple of 4
F[5,-2]	y=8x/3-3	x=9t	y=3(8t-1)	Prohibit t=0

If the transformed Collatz value is prohibited from being negative or fractional,
This transformation appears in multiples of 3 from multiples of 3.

Definition3-1

Let's call the resulting split sequence ***Extended Complete Div sequence*** .

important point

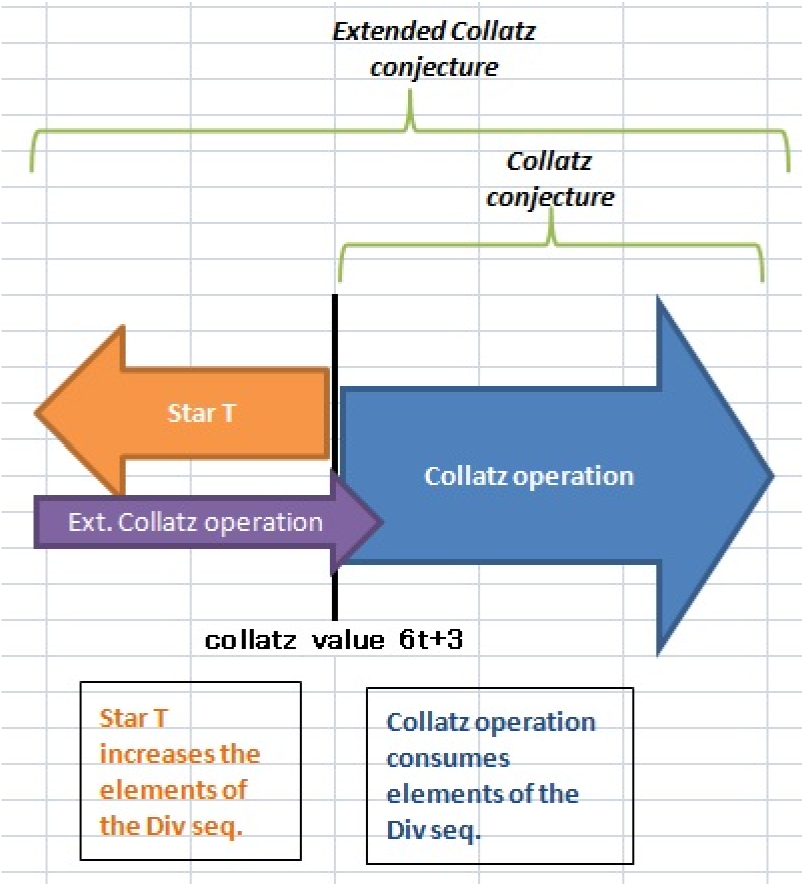
Multiple Extended Complete Div sequence correspond to one Collatz value.

Chapter04 Extended Collatz conjecture

Definition4-1 Extended Collatz conjecture

```
6t+3 (t≦0) is prepared. (If we do Collatz operation from here, it will become the Collatz conjecture)
Let α be the one subjected to Collatz operation once.
Perform 1 to 3 Star Transformation from 6t+3.
Extended Collatz operation therefrom. Switch to normal Collatz operation when returning to α.
We define this as the Extended Collatz conjecture.
※Extended Collatz operation
Apply (3x+1)/2^p to the Collatz value x. p is the first term of the Div seq (0 or negative can also be taken).
```

- image



- Collatz values may be 0 or fractions in the Extended Collatz operation area.

Theorem4-1

```
From a certain Collatz value (one that Collatz operation to α) to
・ Once we do the Star transformation, operate twice with the Extended Collatz
operation, it returns to α
・ Twice we do the Star transformation, operate 3 times with the Extended Collatz
operation, it returns to α
・ 3 times we do the Star transformation, operate 4 times with the Extended Collatz
operation, it returns to α
```

- example, one pattern
  x=9t+3 [3, *,...]
  Once we do Collatz operation: (3(9t+3)+1)/8 = **(27t+10)/8** .
  When we do the Star transformation to x with A[6, -4] y=(4/3)x -7
  12t-3 [6, -1, *,...] .
  In the Extended Collatz operation first time: (3(12t-3)+1)/2^6 = (9t-2)/2^4
  In the Extended Collatz operation second time: (3((9t-2)/2^4)+1)2
  = **(27t+10)/8** . Two match.
- all pattern
  (StarT: A, B, C, D, E, F) × (StarT: first, second, third) × (first term of Div Seq: 4, 3, 2, 1) is 1032 pattern

We confirmed it with Egison. See
https://github.com/righ1113/collatzProof_DivSeq/blob/master/program/Extension.egi .

Theorem4-2

```
Extended Collatz conjecture is true ⇒ Collatz conjecture is true
```

If you repeat the Extended Collatz operation from the Star transformation 1 to 3 times,
According to Theorem4-1, the same as α which transits from all 6t+3,
It is obtained without lack.
Therefore, if the Extended Collatz conjecture is true,
Regular sharing behind, the usual Collatz conjecture will also be true.

plan

In Chapter05-08, we prove the Extended Collatz conjecture.

Chapter05 allDivSeq

level

Let's call **divSeq x** as a function returning Complete Div sequence of normal
(all elements are positive) with x as the Collatz value.
Let this be Complete Div sequence of level 0.
Level 1 Complete Div sequence with Star transformation once is taken as level 1 Complete Div sequence.
Below, it is assumed that the level increases by 1 each time the Star transformation is performed.

Extended Complete Div sequence with level 1 or higher includes items whose elements are 0 or negative.
For proof, we use a Extended Complete Div sequence up to level 3.

allDivSeq

allDivSeq is a function that returns all Complete Div sequence below a specified level of a certain Collatz value.

The following is a simplified explanation.
The first argument is the Collatz value.
The second argument is the level.
We will explain with **allDivSeq 3 1** as an example.
We perform the Star transformation once and consider what Collatz value will be 3.

```
allDivSeq 3 1
  = B[1,-2] + allDivSeq 21 0
  , C[4,-4] + allDivSeq 15 0
```

```
  , D[3,-2] + allDivSeq 6 0
  , E[2,-4] + allDivSeq 45 0
```

We can express it recursively.
When the second argument becomes 0, its Collatz value shows,
We apply a positive Complete Div sequence (**divSeq**) for all terms.
Ignore even numbers.

```
allDivSeq 3 1
  = B[1,-2] + [6]
  , C[4,-4] + [1,1,1,5,4]
  , D[3,-2] + Nothing
  , E[2,-4] + [3,2,3,4]
```

Chapter06 infinite descent

In the proof, we use the following theorem, which transformed the infinite descent.

Theorem6-1

```
((n:Nat) -> P (S n) -> (m ** (LTE (S m) (S n), P m)))
  -> Not (P Z)
    -> Not (P n)
```

- **S** is a natural number plus one. **Z** is **0**.
- **LTE x y** is the meaning of **x≦y**.
- **(x : A ** P x)** is the meaning of **P(x)** is true, **x** of type **A** exists.

This theorem proved with proof assistant Isabelle.
Isabelle has a powerful automatic certification command called **sledgehammer**.

Chapter07 Level lowering function

in the case of False (infinite term does not exist)

False is easy.

```
-- ProofColDivSeqMain.idr
lvDown2 : (n, lv:Nat)
  -> any unLimited (allDivSeq (n+n+n) lv) = False
    -> any unLimited (allDivSeq (n+n+n) (pred lv)) = False -- function that pred is -1
```

- If lv = 0, pred 0 = 0, so you can simply return the argument.

- In the case of lv = (S lv), from the definition of allDivSeq,

```
  allDivSeq x (S lv) = allDivSeq x lv   ┐ ---------------------→ allDivSeq x lv
                     ++ allDivSeqA x lv  |                         β  There is no infinite term in allDivSeq whose
level has fallen by one
                     ++ allDivSeqB x lv  | α
                     ++ allDivSeqC x lv  | Because there is no infinite term in everything
                     ++ allDivSeqD x lv  |
                     ++ allDivSeqE x lv  |
                     ++ allDivSeqF x lv  |
                     ++ allDivSeqG x lv  ┘
```

We can lower the level.

.

.

In the case of True (infinite term exists)

True is a little annoying.

```
-- ProofColDivSeqLvDown.idr
  lvDown' : (n, lv:Nat) -> myAny (\t => not (limited t)) $ allDivSeq n lv = True
                        -> myAny (\t => not (limited t)) $ allDivSeq n (pred lv) = True
```

- If lv = 0, pred 0 = 0, so you can simply return the argument.
- In the case of lv = (S lv), from the definition of allDivSeq,

(A) when allDivSeq x lv has an infinite term

```
  allDivSeq x (S lv) = allDivSeq x lv   ★---------------------→ ★  allDivSeq x lv
                     ++ allDivSeqA x lv                   There is an infinite term in allDivSeq whose level has
dropped by one
                     ++ allDivSeqB x lv
                     ++ allDivSeqC x lv
                     ++ allDivSeqD x lv
                     ++ allDivSeqE x lv
                     ++ allDivSeqF x lv
                     ++ allDivSeqG x lv
```

(B) other (for example, if allDivSeqD has infinite term)

```
allDivSeq x (S lv) = allDivSeq x lv          allDivSeq x lv = allDivSeq x (pred lv)
                  ++ allDivSeqA x lv                         ++ allDivSeqA x (pred lv)
                  ++ allDivSeqB x lv                         ++ allDivSeqB x (pred lv)
                  ++ allDivSeqC x lv                        ++ allDivSeqC x (pred lv)
                  ++ allDivSeqD x lv  ★----------------→   ++ allDivSeqD x (pred lv)★?
                  ++ allDivSeqE x lv                         ++ allDivSeqE x (pred lv)
                  ++ allDivSeqF x lv                         ++ allDivSeqF x (pred lv)
                  ++ allDivSeqG x lv                         ++ allDivSeqG x (pred lv)
```

The definition of allDivSeqD is as follows.

```
allDivSeqD : Nat -> Nat -> List (Maybe (List Integer))
allDivSeqD x Z =
  if ((x+1) `myMod` 2) == 0 && (((x+1) `myMod` 2) `myMod` 2) == 1
    then [[3,-2] `dsp` (Just (divSeq ((x+1)*3 `myDiv` 2)))]
    else []
allDivSeqD x (S lv) =
  if ((x+1) `myMod` 2) == 0
    then map ([3,-2] `dsp`) $ allDivSeq ((x+1)*3 `myDiv` 2) (S lv)
    else []
```

In the case of Z, argument and return are the same from the nature of pred, so we can just return the argument as it is.
(S lv), the else clause is an empty list, but this is incompatible with being an infinite term. Just think about the then clause.
In the then clause, **map ([3, -2]`dsp`) $** is prefixed to it, but its presence or absence does not affect the infinity of the split sequence, so we will remove it.
Therefore,

```
allDivSeq ((x+1)*3 `myDiv` 2) (S lv)  ★------------→  allDivSeq ((x+1)*3 `myDiv` 2) lv  ★
                                        here
                                        use
                                        lvDown' !
So there is an infinite term in allDivSeqD x (pred lv) !
```

Use lvDown' while defining lvDown'.
Since the argument changes from lv to Z, (S lv), it is one step lower, so we can use lvDown' on the 1st row.
Agda and Idris are basic techniques.

Chapter08 allDivSeq at level 0 is all finite terms

base

Infinite descent.
It is proved that the level 2 Div sequence are all finite terms.

```
P : Nat -> Nat -> Type
P n lv = any unLimited $ allDivSeq (n+n+n) lv = True

postulate infiniteDescent :
  ((n:Nat) -> P (S n) 2 -> (m ** (LTE (S m) (S n), P m 2)))
    -> any unLimited $ allDivSeq Z 2 = False
      -> any unLimited $ allDivSeq (n+n+n) 2 = False
```

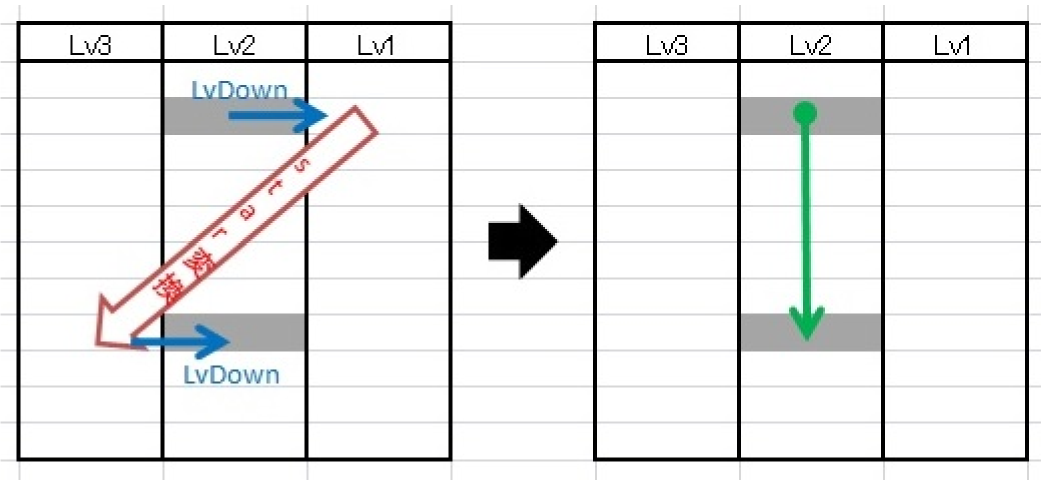
first sufficiency

If there is an infinite term Div sequence at Level 2's 1 or more Collatz value c, there is also an infinite term Div sequence for c' smaller than that.

Theorem8-1

```
  ((n:Nat) -> P (S n) 2 -> (m ** (LTE (S m) (S n), P m 2)))
```

Assuming Div sequence of an infinite term whose Collatz value is c, apply Star transformation that reduces the Collatz value.
(Even if Star transformation is applied, the infinity of Div sequence does not change)
Since the level changes at this time, level is unified to 2 using the level lowering function.



Proof by case.

case	starT 1	after 1	starT 2	after 2	small?
3 mod 9					

case	starT 1	after 1	starT 2	after 2	small?
18t+3	B[1,-2] y=x/6-1/2	3t			18t+3 > 3t
54t+12	A[6,-4] y=4x/3-7	72t+9	E[2,-4] y=x/12-3/4	6t	54t+12 > 6t
54t+30	A[6,-4] y=4x/3-7	72t+33	C[4,-4] y=x/3-2	24t+9	54t+30 > 24t+9
54t+48	A[6,-4] y=4x/3-7	72t+57	B[1,-2] y=x/6-1/2	12t+9	54t+48 > 12t+9
6 mod 9					
9t+6	C[4,-4] y=x/3-2	3t			9t+6 > 3t
0 mod 9					
36t+9	E[2,-4] y=x/12-3/4	3t			36t+9 > 3t
108t+27	F[5,-2] y=8x/3-3	288t+69	C[4,-4] y=x/3-2	96t+21	108t+27 > 96t+21
108t+63	F[5,-2] y=8x/3-3	288t+165	B[1,-2] y=x/6-1/2	48t+27	108t+63 > 48t+27
108t+99	F[5,-2] y=8x/3-3	288t+261	E[2,-4] y=x/12-3/4	24t+21	108t+99 > 24t+21
108t+18	F[5,-2] y=8x/3-3	288t+45	E[2,-4] y=x/12-3/4	24t+3	108t+18 > 24t+3
108t+54	F[5,-2] y=8x/3-3	288t+141	C[4,-4] y=x/3-2	96t+45	108t+54 > 96t+45
108t+90	F[5,-2] y=8x/3-3	288t+237	B[1,-2] y=x/6-1/2	48t+39	108t+90 > 48t+39
108t+36	F[5,-2] y=8x/3-3	288t+93	B[1,-2] y=x/6-1/2	48t+15	108t+36 > 48t+15
108t+72	F[5,-2] y=8x/3-3	288t+189	E[2,-4] y=x/12-3/4	24t+15	108t+72 > 24t+15
108t+108	F[5,-2] y=8x/3-3	288t+285	C[4,-4] y=x/3-2	96t+93	108t+108 > 96t+93

The first sufficiently satisfied with this.

second sufficiency

Level 2, Div sequence of Collatz value 0, are all finite terms.

Theorem8-2

```
any unLimited $ allDivSeq Z 2 = False
```

Since this can be actually calculated by a program, we confirmed it by running it.
Please refer to BaseLog0.txt .

checkmate

Since we could say **any unLimited $ allDivSeq (n+n+n) 2 = False** , after that we lowered this level to 0 and it was completed.

Theorem8-3

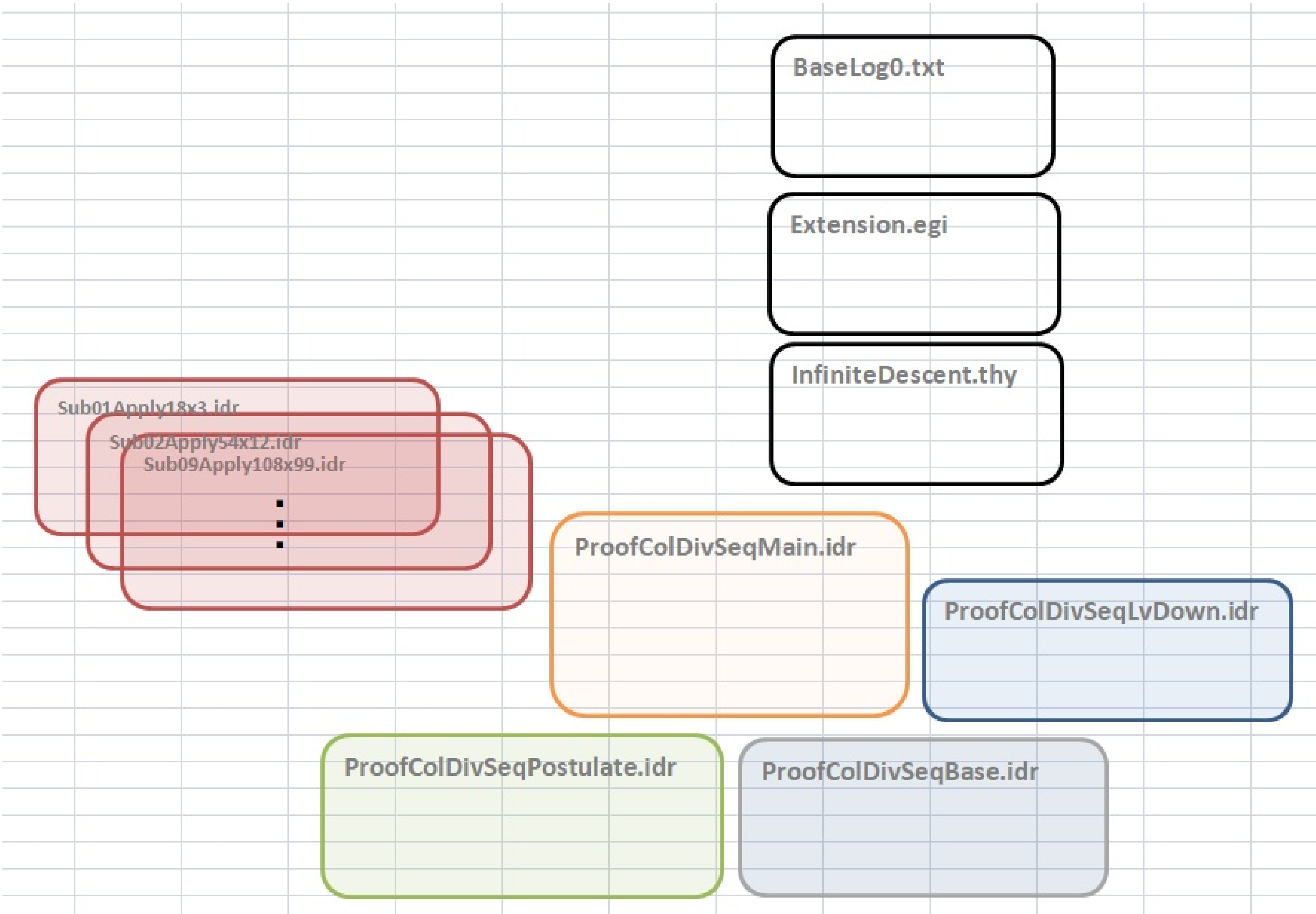
```
allDivSeqInfFalse' : any unLimited (allDivSeq (n+n+n) 2) = False
allDivSeqInfFalse' = infiniteDescent unifi base0

-- last theorem
allDivSeqInfFalse : (n:Nat)
  -> any unLimited (allDivSeq (n+n+n) 0) = False
allDivSeqInfFalse n = lvDown2 n 1 $ lvDown2 n 2 allDivSeqInfFalse'
```

Part3 formal verification in Idris

Chapter09 description of each file

image



other

BaseLog0.txt

- "Div sequence of level 2, Collatz value 0 is all finite term" execution result.

```
any unLimited $ allDivSeq Z 2 = False
```

Extension.egi

- Theorem4-1

```
From a certain Collatz value (one that Collatz operation to α) to
・ Once we do the Star transformation, operate twice with the Extended Collatz
operation, it returns to α
・ Twice we do the Star transformation, operate 3 times with the Extended Collatz
operation, it returns to α
・ 3 times we do the Star transformation, operate 4 times with the Extended Collatz
operation, it returns to α
```

proof. Use Egison.

InfiniteDescent.thy

- Proof of the infinite descent and proof of the theore using "any".
  Use Isabelle.

Idris

ProofColDivSeqBase.idr

- We are making a base part. allDivSeq etc.

ProofColDivSeqLvDown.idr

- Definition and proof of functions that Level lowering function.

ProofColDivSeqMain.idr

- This is the main proof. Using the infinite descent, we divide and prove 15 patterns.

ProofColDivSeqPostulate.idr

- We have 36 postural propositions.

Sub01...~Sub15....idr(s)

- The case of 15 patterns of the main function is divided by 1 file separately.
  For these reasons, these files are dirty for technical reasons.

Chapter10 proof of postulate proposition, by hand

from ProofColDivSeqBase

postulate infiniteDescent

```
postulate infiniteDescent :
  ((n:Nat) -> P (S n) 2 -> (m ** (LTE (S m) (S n), P m 2)))
    -> any unLimited $ allDivSeq Z 2 = False
      -> any unLimited $ allDivSeq (n+n+n) 2 = False
```

We proved it with InfiniteDescent.thy OK.

postulate base0

```
postulate base0 : any unLimited $ allDivSeq Z 2 = False
```

From BaseLog0.txt, it is guaranteed, so OK.

from ProofColDivSeqLvDown

postulate any1

```
postulate any1 : (pp:a->Bool) -> (xs, ys:List a)
  -> myAny pp (xs ++ ys) = myAny pp xs || myAny pp ys
```

We proved it with InfiniteDescent.thy OK.

postulate changeA

```
postulate changeA : (x, lv:Nat) -> (myAny (\t => not (limited t)) (allDivSeqA n
lv) = True)
  -> (myAny (\t => not (limited t)) (allDivSeq (divNatNZ ((x+7)*3) 4 SIsNotZ) lv)
= True)
```

Expand allDivSeqA n lv.
Even if you scratch the front, it will not affect the infinity of Div sequence OK.
The same is true for 6 below.

postulate changeA0

```
postulate changeA0 : (x:Nat) -> (myAny (\t => not (limited t)) (allDivSeqA n 0) =
True)
  -> (myAny (\t => not (limited t)) [Just (divSeq (divNatNZ ((x+7)*3) 4 SIsNotZ))]
= True)
```

Expand allDivSeqA n 0.
Even if you scratch the front, it will not affect the infinity of Div sequence OK.
The same is true for 6 below.

postulate unfold3

```
postulate unfold3 : (x, lv:Nat) -> (myAny (\t => not (limited t)) $ allDivSeq x lv
= True) =
  Either (myAny (\t => not (limited t)) $ allDivSeq x (pred lv) = True)
    (Either (myAny (\t => not (limited t)) $ allDivSeq (divNatNZ ((x+7)*3) 4
SIsNotZ) (pred lv) = True)
      (Either (myAny (\t => not (limited t)) $ allDivSeq (x*6+3) (pred lv) = True)
        (Either (myAny (\t => not (limited t)) $ allDivSeq (x*3+6) (pred lv) =
True)
          (Either (myAny (\t => not (limited t)) $ allDivSeq (divNatNZ ((x+1)*3) 2
SIsNotZ) (pred lv) = True)
            (Either (myAny (\t => not (limited t)) $ allDivSeq (x*12+9) (pred lv)
```

```
= True)
              (Either (myAny (\t => not (limited t)) $ allDivSeq (divNatNZ
((x+3)*3) 8 SIsNotZ) (pred lv) = True)
                      (myAny (\t => not (limited t)) $ allDivSeq (divNatNZ (x
`minus` 21) 64 SIsNotZ) (pred lv) = True)))))))
```

Expand allDivSeq x lv.
Even if you scratch the front, it will not affect the infinity of Div sequence OK.

postulate unfold0

```
postulate unfold0 : (x:Nat) -> (myAny (\t => not (limited t)) $ allDivSeq x 0 =
True) =
  Either (myAny (\t => not (limited t)) $ [Just (divSeq x)] = True)
    (Either (myAny (\t => not (limited t)) $ [Just (divSeq (divNatNZ ((x+7)*3) 4
SIsNotZ))] = True)
      (Either (myAny (\t => not (limited t)) $ [Just (divSeq (x*6+3))] = True)
        (Either (myAny (\t => not (limited t)) $ [Just (divSeq (x*3+6))] = True)
          (Either (myAny (\t => not (limited t)) $ [Just (divSeq (divNatNZ
((x+1)*3) 2 SIsNotZ))] = True)
            (Either (myAny (\t => not (limited t)) $ [Just (divSeq (x*12+9))] =
True)
              (Either (myAny (\t => not (limited t)) $ [Just (divSeq (divNatNZ
((x+3)*3) 8 SIsNotZ))] = True)
                      (myAny (\t => not (limited t)) $ [Just (divSeq (divNatNZ (x
`minus` 21) 64 SIsNotZ))] = True)))))))
```

Expand allDivSeq x 0.
Even if you scratch the front, it will not affect the infinity of Div sequence OK.

postulate lvDown

```
postulate lvDown : (n, lv:Nat) -> P n lv -> P n (pred lv)
```

We proved lvDown' on ProofColDivSeqLvDown.idr OK.

from sub0xxxxx

postulate b18x3To3x'

```
postulate b18x3To3x' :
  (k:Nat) -> P (S (plus (plus (plus k k) (plus k k)) (plus k k))) 1 -> P k 2
```

OK from the table of first sufficiency in Chapter08.
The same is true for 14 below.

from ProofColDivSeqMain

postulate aDSFalse

```
postulate aDSFalse : (x, lv:Nat)
  -> any unLimited (allDivSeq x lv
                ++ allDivSeqA x lv
                ++ allDivSeqB x lv
                ++ allDivSeqC x lv
                ++ allDivSeqD x lv
                ++ allDivSeqE x lv
                ++ allDivSeqF x lv
                ++ allDivSeqG x lv) = False
    -> any unLimited (allDivSeq x lv) = False
```

If any is False, all elements are False OK.
.
.

Acknowledgment

I thank everyone who read this paper.
In 5ch "コラッツ予想がとけたらいいな その2", I gave a meaningful opinion.
In particular, I am grateful to "前786 ◆5A/gU5yzeU" san.

.
.

Reference

[1] Lagarias, Jeffrey C., ed. (2010). The Ultimate Challenge: The 3x+1 Problem, American Mathematical Society, ISBN 978-0-8218-4940-8, Zbl 1253.11003.