Collatz algorithm & its relatives

Grażyna Mirkowska & Andrzej Salwicki

Instytut Informatyki UKSW &
Dombrova Research , Dąbrowa Leśna

October, 24, 2021

Abstract

An abridged, simplified proof of Collatz theorem is presented.

Part I - Observations

Collatz algorithm *Cl*

```
var n: Nat ;
        (* n is a natural number >0, *)
         (* Nat is the environment in which Cl is executed-*)
read(n);
```

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

Properties of Nat structure – reminder

$$Nat = \langle N, +, 0, 1; =, <, Parzyste \rangle$$

N is a set,

$+ : N \times N \to N$ functor $+$ denotes operation of addition ,

$0, 1$ distinguished elements of N

$=$ equality relation $<$ ordering relation

even parity relation

valid sentences (axioms)

$$\forall_n \; n + 0 = n \tag{1}$$

$$\forall_{n,m} \; n + (m + 1) = (n + m) + 1 \Rightarrow n = m \tag{2}$$

moreover , for every formula Φ

$$(\Phi(0) \land \forall_n \; [\Phi(n) \to \Phi(n+1)]) \to \forall_n \; \Phi(n) \tag{3}$$

Algorithm Cl repeated

Algoruthm *Cl* is idempotent, it may be executed twice, with no harm.

```
read (m);
n:=m;
     while n≠ 1 do
Cl:     if even(n) then n:=n÷ 2 else n:= 3n+1 fi
     od ;
------------------------------------------------------------
n:=m;
     while n≠ 1 do
        if even(n) then n:=n÷2 else n:= 3n+1 fi
     od
```

Let's count the number of divisions

```
read(m);
n:=m; z:=0
while n≠ 1 do
   if even(n) then n:=n÷2 ; z:=z+1else n:= 3n+1 fi
od
```

```
n:=m;
while n≠ 1 do
   if even(n) then n:=n÷2 else n:= 3n+1 fi
od
```

Let's count the number of multiplications

```
read(m);
n:=m; z:=0; x:=0;
while n≠ 1 do
   if even(n) then n:=n÷2 ; z:=z+1else n:= 3n+1; x:=x+1fi
od
```

```
n:=m;
while n≠ 1 do
   if even(n) then n:=n÷2 else n:= 3n+1 fi
od
```

Observations

Fact

$z > x$

and not entirely out of thin air[1]

Fact

$2^z > n \cdot 3^x$

[1] G. Mirkowska, A.Salwicki, *On Collatz theorem*, 2021
https://dabrowa-research.pl/images/c/c4/On-Collatz_thm-11–10-21.pdf

Let's look

```
read(m);
n:=m; z:=0;  x:=0;
while n≠ 1 do
   if even(n) then n:=n÷2 ;  z:=z+1else n:= 3n+1;  x:=x+1fi
od
```

```
n:=m;
while n≠ 1 do
   if even(n) then n:=n÷2;z:=z-1 else n:= 3n+1; x:=x-1 fi
od
```

Now, after execution of the above program $x = 0$ and $z = 0$.

Put y= $2^z - n \cdot 3^x$

```
read(m);
n:=m; z:=0;  x:=0;
while n≠ 1 do
   if even(n) then n:=n÷2 ;  z:=z+1else n:= 3n+1;  x:=x+1fi
od ;
```

---

n:=m; y:= $2^z - n \cdot 3^x$;

```
while n≠ 1 do
   if even(n) then n:=n/2; z:=z-1;y:=y/2
   n:=3n+1; y:=y-3^(x-1); x:=x-1; fi
od ;
```

Now $x = 0 \wedge z = 0 \wedge y = 0$ isn't it?

Invariant

The following formula holds in every step of execution of the second while statement $\boxed{2^z = n \cdot 3^x + y}$

```
read(m);
n:=m; z:=0; x:=0;
while n≠ 1 do
   if even(n) then n:=n÷2 ; z:=z+1else n:= 3n+1; x:=x+1fi
od
```

---

n:=m; y:=$2^z - n \cdot 3^x$;

---

while n≠ 1 do (* $\boxed{2^z = n \cdot 3^x + y}$ *)

if even(n) then n:=n/2; z:=z-1;y:=y/2 (* $\boxed{2^z = n \cdot 3^x + y}$ *)

else n:=3n+1; y:=y-3^{x-1}; x:=x-1; (* $\boxed{2^z = n \cdot 3^x + y}$ *) fi

od ; (* $\boxed{2^z = n \cdot 3^x + y}$ *)

Eliminate variable n in the second while statement

Remark an invariant: $n \mod 1 \equiv y \mod 1$

```
read(m); n:=m;
CI':  z:=0;  x:=0;
      while n≠ 1 do
         if even(n) then n:=n÷2 ;  z:=z+1else n:= 3n+1;  x:=x+1fi
      od ;
```

y:=$2^z - m \cdot 3^x$;

```
IC':  while 3^x + y ≠ 2^z do
         if even(y) then z:=z-1;y:=y/2
         else y:=y-3^(x-1); x:=x-1;
      od
```

What is the conclusion?

Part 2

Four lemmas and the theorem.

Case 1 - when execution of CI algorithm is finite

Let IC denote the following algorithm

IC:
```
while 3^x + y ≠ 2^z do
   if ¬ even(y) ∧(x = 0 ∨ y < 3^(x−1)) then Err:=true; exit fi;
   if even(y) then z:=z-1;y:=y/2
   else y:=y-3^(x−1); x:=x-1; fi
od ;
```

Fact

Every execution of algorithm IC is finite.

Lemma 1

If for a given natural number n execution of algorithm CI is finite, then there are three natural numbers x, y, z, such that $n \cdot 3^x + y = 2^z$ and moreover the execution of algorithm IC is error-free.

Case 2 – error-free execution of *IC* implies halt of CL

Lemma 2

Suppose that for some natural numbers n, x, y, z the equality $n \cdot 3^x + y = 2^z$ holds and the execution of algorithm IC is free of error Err,
tthen the execution of Collatz algorithm for natural number n is finite.

Let $\mathfrak{M}$ denote the following algebraic structure, a non-standard model of theory of addition.

$$\mathfrak{M} = \langle \mathbb{Z} \times \mathbb{Q}^+; +, \underbrace{(0;0)}_{0}, \underbrace{(1;0)}_{1}; =\rangle$$

The universe is the set of pairs $\langle k, w\rangle$ such, that $k \in \mathbb{Z}$ jis an integer, , and $w \in \mathbb{Q}^+$ is a positive, rational number. Note, , when $w = 0$ then $k \geq 0$.
Addition is defined as follow $\langle k, w\rangle + \langle k', w'\rangle = \langle k + k', w + w'\rangle$.
Element 1 (one) is $\langle 1, 0\rangle$, 0 (zero) is $\langle 0, 0\rangle$.
Any element e such, that $w \neq 0$ i.e. element $\langle k, w\rangle$ is *unreachable* , for the program

{ y:=(0,0); while e $\neq$ y do y :=y+(1,0) od}

will not terminate.

Lemma 3 - unreachable $\subset$ non-Collatz

Lemma 3

For every unreachable element of the structure $\mathfrak{M}$ execution of Collatz algorithm does not terminate.

Instead of proof. Consider the following example of computatuin of Collatz algorithm for $n_\varepsilon = (5; \frac{1}{2})$. (Remember, addition in structure $\mathfrak{M}$ is defined pairwise.)

$(5; \frac{1}{2}) \stackrel{\cdot 3+1}{\to} (16; \frac{3}{2}) \stackrel{/2}{\to} (8; \frac{3}{4}) \stackrel{/2}{\to} (4; \frac{3}{8}) \stackrel{/2}{\to} (2; \frac{3}{16}) \stackrel{/2}{\to} (1; \frac{3}{32}) \stackrel{\cdot 3+1}{\to} (4; \frac{9}{32}) \stackrel{/2}{\to} (2; \frac{9}{64}) \stackrel{/2}{\to}$
$(1; \frac{9}{128}) \stackrel{\cdot 3+1}{\to} (4; \frac{27}{128}) \stackrel{/2}{\to} (2; \frac{27}{256}) \stackrel{/2}{\to} (1; \frac{27}{512}) \stackrel{\cdot 3+1}{\to} \ldots$

Experiment with n= (5;0) and compare.

Case 4 – non-Collatz $\subset$ unreachable

Lemma 4

If for an element ε computation of Collatz algorithm is infinite then the struture in which the algorithm is executed contains unreachable element.

For a lengthy proof consult the paper

https://dabrowa-research.pl/images/c/c4/On-Collatz_thm-11–10-21.pdf

On following slides we offer some hints.

Algorithm IIC in search of error-free triple

For an element n we let $z = (\mu l)(2^l > n)$.

IIC:

```
read(n);
x, x_s := 0;  z_s := z;  y, y_s := 2^z − n; Err := false;
while 3^{x_s} + y_s ≠ 2^{z_s} do
  IC: while 3^x + y ≠ 2^z do
        if odd(y) ∧ (x = 0 ∨ y < 3^{x−1})
          then Err:=true; exit fi;
        if odd(y) then y := y − 3^{x−1}; x := x − 1
          else y := y/2; z := z − 1 fi;
      od;
  if Err then
     x, x_s := x_s + 1;  z, z_s := z_s + 2;  y, y_s := 2^{z_s} + 3 · y_s;  Err := false;
  else  exit fi;
od
```

Fact

An element n has an infinite Collatz computation if and only if, the algorithm IIC has an infinite computation.

algorithm B4 with FIFO queue of triples

For a given element n the algorithm B4 constructs its Collatz tree DC and returns the triple t which represents n and is error-free. But what if $n \notin DC$?.

```
      unit F4: class(m,x,y,z: Nat); end F4;
```

B4:

```
read(n);   x:=0; y:=0; z:=0;m:=1; p:=new Kolejka;
while n ≠ m  do
   p:=put(new F4(⟨2 ∗ m, x, 2 ∗ y, z + 1⟩),p);
   if m  mod 3 = 1 ∧ m ≠ 4 then
        if ((m − 1) ÷ 3)  mod 2 = 1 then
          p:=put(new F4((m − 1) ÷ 3, x + 1, y + 3^x, z),p);
        fi
   fi;
   t := first(p); m:=t.m;x:=t.x;y:=t.y;z:=t.z; p:=usun1z(p);
od
```

Fact

If for an element n the computation of Collatz algorithm is indinite, then a triple $\langle x, y, z\rangle$ exists such that the computation of IC algorithm is infinite and the elements x, y, z are unreachable.

Collatz theorem

Theorem

For any reachable (i.e. standard) natural number n, the execution of Collatz algorithm Cl terminates.

The end

Your comments are welcome.
mailto: salwicki[at]gmail[dot]com