SpecVer project (2023) 1–34
Dombrova Research

Collatz conjecture becomes theorem

Grażyna Mirkowska
UKSW University of Warsaw
Institute of Informatics

Andrzej Salwicki
Dombrova Research
Łomianki, POLAND
salwicki at mimuw.edu.pl

Abstract. The Collatz hypothesis is a theorem of the algorithmic theory of natural numbers. We prove the (algorithmic) formula that expresses the halting property of Collatz algorithm. The calculus of programs is used in the proof.
The observation that Collatz's theorem cannot be proved in any elementary number theory completes the main result. For the Collatz property can be falsified in non-standard models of such theories.

Key words: algorithm of Collatz, halting property, Collatz tree, calculus of programs,algorithmic theory of numbers.

MSC-class: 03D02 (Primary) 68Q02 (Secondary)
ACM-class: F.3.1; D.2.4

Contents

1. Introduction

The $3x + 1$ problem remained open for over 80 years. It has been noticed in 1937 by Lothar Collatz. The problem became quite popular due to its wording, for it is short and easy to comprehend.
Collatz remarked that for any given natural number $n > 0$, the sequence $\{n_i\}$ defined by the following recurrence

$$\left.\begin{aligned} &n_0 = n \\ &n_{i+1} = \begin{cases} n_i \div 2 & \text{when } n_i \text{ is even} \\ 3 \cdot n_i + 1 & \text{when } n_i \text{ is odd} \end{cases} \quad \text{for } \ i \geq 0 \end{aligned}\right\} \tag{rec1}$$

seem always reach the value 1.
He formulated the following conjecture

$$\text{for all } n \text{ exists } i \text{ such that } n_i = 1 \tag{Collatz conjecture}$$

One can give another formulation of the hypothesis of Collatz [1].
The number of papers devoted to the problem surpasses 200, c.f. [Lag10] . It is worthwhile to consult social media: wikipedia, youtube etc, there you can find some surprising ideas to prove the Collatz hypothesis as well as a technical analysis of the problem.
Computers are used and are still crunching numbers in the search of an eventual counterexample to the Collatz conjecture. The reports on progress appear each year. We claim that the counterexample approach is pointless, i.e. the computers can be turned off. Namely, we shall prove that the program that searches a counterexample will never stop. Our goal will be achieved if we prove that for each number n the computation of the following Cl algorithm is finite.

$$\left\{\begin{array}{l} \textbf{while } n \neq 1 \textbf{ do} \\ \quad \textbf{if } even(n) \textbf{ then } n := n \div 2 \textbf{ else } n := 3n + 1 \textbf{ fi} \\ \textbf{od} \end{array}\right\} \tag{Cl}$$

We need the following items

- a formula Θ_{Cl} such that it expresses the termination property of program Cl,
- a definition of relation $\mathcal{C}$ of logical consequence operation (provability) and
- a verifiable proof Π of the halting formula Θ_{Cl}.

Ah, we need also a specification of the domain in which the algorithm is to be executed, i.e. the axioms $\mathcal{A}x$ of the algebraic structure of natural numbers.

QUESTION 1. How to express the termination property of a program K as a formula Θ_K (i.e. a Boolean expression)?
Note, there is no a universal algorithm that builds the halting formula of a given program K as an appropriate first-order logical formula Θ_K. The existence of such algorithm would contradict the theorem on incompleteness of arithmetics,

[1] Let $f(n,0) \stackrel{df}{=} n$, and $f(n,i+1) \stackrel{df}{=} \begin{cases} f(n,i)/2 & \text{if } f(n,i) \text{ is even} \\ 3 \cdot f(n,i)+1 & \text{if } f(n,i) \text{ is odd} \end{cases}$. Now, conjecture reads $\forall_n \exists_i f(n,i) = 1$.

cf. Kurt Gödel . According to Gödel, the property *to be a natural number* is not expressible by any set of first-order formulas. The reader may wish to note, that halting property of the algoritm

$$q := 0; \textbf{while } q \neq n \textbf{ do } q := q + 1 \textbf{ od}$$

is valid in a data structure$\mathfrak{A}$ iff n is a standard (i.e. reachable) natural number. Therefore the halting property allow to define the set of natural numbers. In this situation it seems natural to pass from first-order language to another more expressive language. There are three candidates:

1° a second-order language admitting variables and quantifiers over sets,

2° the language of infinte disjunctions and conjunctions $\mathcal{L}_{\omega_1\omega}$ and

3° language of algorithmic logic.

Problem with second order logic is in lack of adequate definition of consequence operation. True, we can limit our considerations to the case of finite sets (*aka*, weak second order logic). Still we do not know a complete set of axioms and inference rules for the weak second-order logic. Applying second-order logic to program analysis results in a heavy overhead. Because, first you have to translate the semantic property of the program into a property of a certain sequence or set, prove this property and make a backward translation. The question of whether this approach is acceptable to software engineers seems to be appropriate.
The language of infinite disjunctions and conjunctions is not an acceptable tool for software engeeners for the programs are of finite length.
We shall use the language and the consequence operation offered by algorithmic logic i.e. calculus of programs. We enlarge the set of well formed expressions: beside terms and formulas of first order language we accept algorithms and we modify the definition of logical formulas. The simplest algorithmic formulas are of the form:

$$\{algorithm\}\,(formula).$$

As an example of an algorithmic formula consider the expression

$$\{q := 0; \textbf{while } q \neq n \textbf{ do } q := q + 1 \textbf{ od}\}\,(n = q) \tag{1}$$

The latter formula is valid iff every element n can be reached from 0 by adding 1.
Now our goal is to deduce the following statement

$$\forall_{n \in N} \left\{ \begin{array}{l} \textbf{while } n \neq 1 \textbf{ do} \\ \quad \textbf{if } even(n) \textbf{ then } n := \frac{n}{2} \\ \quad \textbf{else } n := 3n + 1 \textbf{ fi} \\ \textbf{od} \end{array} \right\} (n = 1) \tag{Θ}$$

from the axioms of algorithmic theory of natural numbers $\mathcal{ATN}$, c.f. subsection 6.4. For the formula (Θ) expresses the termination property of program Cl.

QUESTION 2. How to prove such algorithmic formula?
Note, all structures that assure the validity of axioms of the $\mathcal{ATN}$ theory are isomorphic (this is the categoricity *meta*-property of the theory $\mathcal{ATN}$). Therefore, the termination formula, can be either proved (with the inference rules and axioms of calculus of programs $\mathcal{AL}$, or validated in this unique model of axioms of $\mathcal{ATN}$.

Let us make a simple observation. The computation of Collatz algorithm if succesful goes through intermediate values. The following diagram (Dg) illustrates a computation where all odd numbers were exposed as stacked fractions.

$$n \to \frac{n}{2^{k_0}} \to \frac{3*(\frac{n}{2^{k_0}})+1}{2^{k_1}} \to \frac{3*\frac{3*\frac{n}{2^{k_0}}+1}{2^{k_1}}+1}{2^{k_2}} \cdots \to \frac{3\cdot\frac{3\cdot\frac{3\cdot\frac{3\cdot\frac{n}{2^{k_0}}+1}{2^{k_1}}+1}{2^{k_2}}+1}{(\ldots)}+1}{2^{k_{x-1}}}+1}{2^{k_x}} = 1 \tag{Dg}$$

where $k_0 = expo(n,2), k_1 = expo(3 \cdot \frac{n}{2^{k_0}} + 1, 2), k_2 = expo(3 * \frac{(3 \cdot \frac{n}{2^{k_0}} + 1)}{2^{k_1}} + 1, 2), \dots$ [2]. In our earlier paper [MS21] we studied the halting formula of the Collatz algorithm. We remarked that the computation of Collatz algorithm is finite iff there exist three natural numbers x, y, z such that:
a) the equation $n \cdot 3^x + y = 2^z$ is satisfied and
b) the computation of another algorithm IC, cf. page 14, is finite, the algorithm computes on triples$\langle x, y, z\rangle$.
It is worthwhile to mention that the subsequent triples are decreasing during computation.
The proof we wrote in [MS21] is overly complicated.

Here we show that the 4-argument relation

$$\{n, x, y, z\} : \{IC\}(true).$$

is elementary recursive, since it may be expressed by an arithmetic expression with operator $\sum$.
The present paper shows arguments simpler and easier to follow.

2. Collatz tree

Definition 2.1. *Collatz tree $\mathcal{DC}$* is a subset $D \subset N$ of the set N of natural numbers and the function f defined on the set $D \setminus \{0, 1\}$.

$$\mathcal{DC} = \langle D, f \rangle$$

where $D \subset N, 1 \in D, f \colon D \setminus \{0, 1\} \to D$.
Function f is determined as follows

$$f(n) = \begin{cases} n \div 2 & \text{when } n \mod 2 = 0 \\ 3n + 1 & \text{when } n \mod 2 = 1 \wedge n \neq 1 \end{cases}$$

, the set D is the least set containing the number 1 and closed with respect to the function f,

$$D = \{ n \in N : \exists_{i \in N}\, f^i(n) = 1 \} .$$

As it is easy to see, this definition is highly entangled and the decision whether the set D contains every natural number is equivalent to the Collatz problem.

Conjecture 2.1. The Collatz tree contains all the reachable natural numbers.

[2]**Note**, the function $expo$ returns the largest exponent of 2 in the prime factorization of number x .
$expo(x, 2) \overset{df}{=} \{l := 0; y := x;$ **while** $even(y)$ **do** $l := l + 1;\ y := y \div 2$ **od**$\}(\text{result} = l)$ i.e. $l = expo(x, 2)$. We introduce the notation $expo$ instead exp for our friends programmers who interpret the notation exp as exponentiation operation.

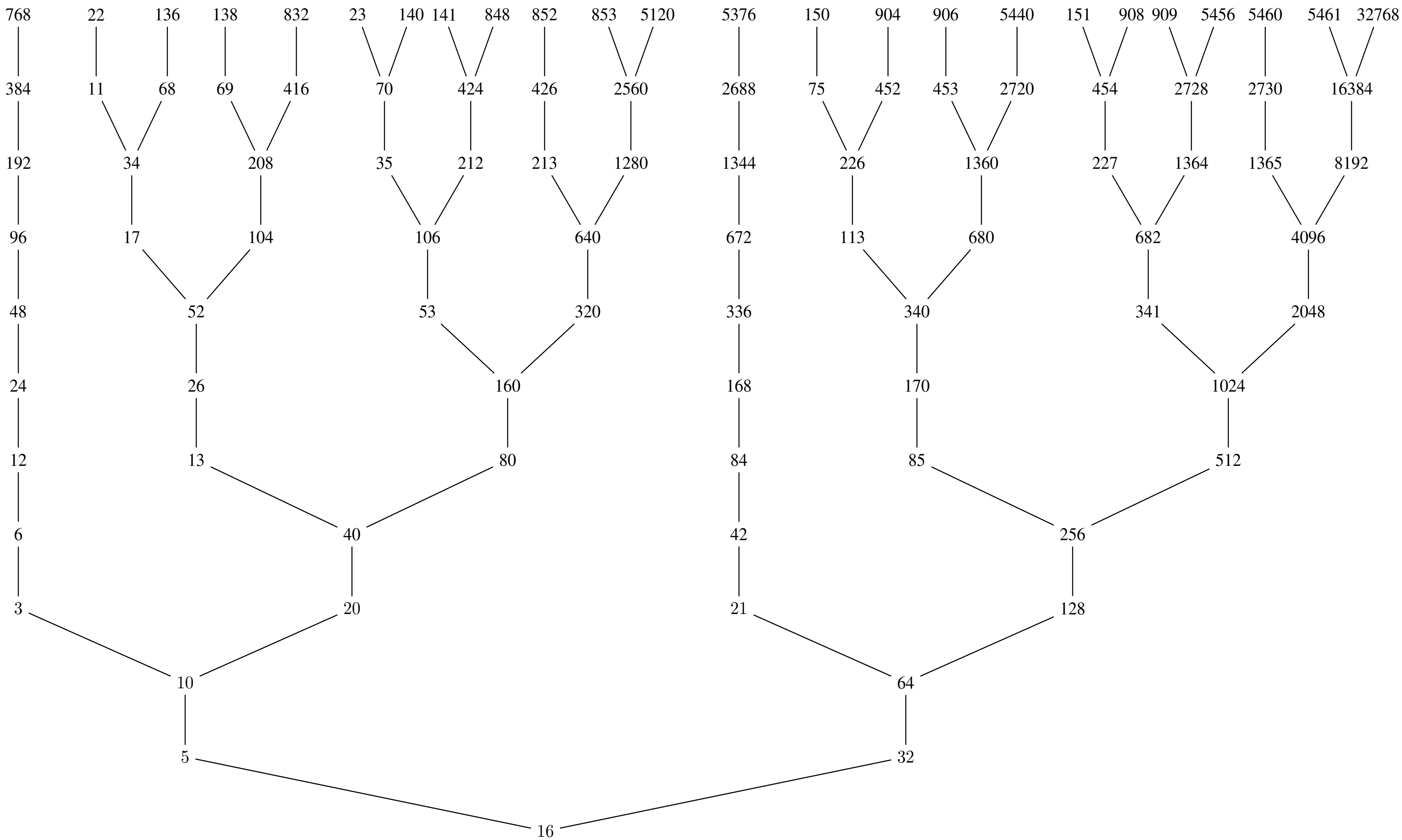

Figure 1. A fragment of Collatz tree, levels 4-15. It does not include levels 0-3, they consist of elements 1 — 2 — 4 — 8 — .

3. Four algorithms, relatives of Cl algorithm

In this section we present an algorithm Gr equivalent to the algorithm Cl and three algorithms $Gr1, Gr2, Gr3$ that are successive extensions of the Gr algorithm.

Lemma 3.1. The following algorithm Gr is equivalent to Collatz algorithm Cl.

Gr:
```
while even(n) do n:= n ÷ 2 od ;
while n ≠1 do
  n:= 3*n+1;
  while even(n) do n:= n ÷ 2 od
od ;
```

Proof:
The equivalence of the algorithms Cl and Gr is intuitive. Compare the recurrence of Collatz (rec1) and the following recurrence (rec2) that is calculated by the algorithm Gr.

$$\left.\begin{array}{rcl} k_0 = expo(n,2) & \wedge & m_0 = \frac{n}{2^{k_0}} \\ k_{i+1} = expo(3m_i+1,2) & \wedge & m_{i+1} = \frac{3m_i+1}{2^{k_{i+1}}} \quad \text{for } i \geq 0 \end{array}\right\} \tag{rec2}$$

One can say the algorithm Gr is obtained by the elimination of **if** instruction from the Cl algorithm. However, construction of a formal proof is a non-obvious task. A sketch of a proof is given in subsection 6.5 We are encouraging the reader to fill the details. □

Next, we present the algorithm $Gr1$, an extension of algorithm Gr.

$Gr1$:
```
var n,l,i :integer ; k,m :arrayof integer;
Γ1:   i := 0; l := 0;
      while even(n) do n:= n ÷ 2; l := l + 1 od ; k_i := l; m_i :=n:
while n ≠1 do
Δ1:   { m_i=n }  n:= 3*n+1;  l := 0;
      while even(n) do n:= n ÷2; l := l + 1od ; k_{i+1} := l; m_{i+1} :=n;
      {m_{i+1} = (3·m_i+1)/2^{K_i} ∧ k_{i+1} = exp(3 * m_i + 1, 2)}   i := i + 1
od
```

Lemma 3.2. Algorithm $Gr1$ has the following properties:
(i) Algorithms Gr and $Gr1$ are equivalent with respect to the halting property.
(ii) The sequences $\{m_i\}$ and $\{k_i\}$ calculated by the algorithm $Gr1$ satisfy the recurrence rec2.

Proof:
Both statements are very intuitive. Algorithm $Gr1$ is an extension of algorithm Gr. The inserted instructions do not interfere with the halting property of algorithm $Gr1$. Second part of the lemma follows easily from the remark that $k_0 = exp(\mathrm{n}, 2)$ and $m_0 = \frac{n}{2^{k_0}}$ and that for all $i > 0$ we have $k_{i+1} = exp(3 * m_i + 1, 2)$ and $m_{i+1} = \frac{3 \cdot m_i + 1}{2^{k_{i+1}}}$. □

Each odd number m in Collatz tree, $m \in D$, initializes a new branch. Let us give a color number $x + 1$ to each new branch emanating from a branch with color number x. Note, for every natural number$p$ the set of branches of the color p is infinite. Let W_x denote the set of natural numbers that obtained the color x.
Besides the levels of Collatz tree, one can distinguish the structure of strata in the tree.

First, we define a congruence relation $\sim$ between numbers: given two odd numbers a and b the relation $a \sim b$ holds iff $b = 4a + 1$, We take symmetric and transitive closure of the relation $\sim$.
Further, if $a \sim b$ and two odd numbers c, d are such $a = 3c + 1$ and $b = 3d + 1$ then $c \sim d$.

Definition 3.1. Definition of strata. Stratum W_0 consists of powers of two.

$$W_0 \stackrel{df}{=} \{n \in N \colon n = 2^{expo(n,2)}\} \tag{W0}$$

Any number $n \notin W_0$ belongs to some set $W_x, x \neq 0$.

$$W_{x+1} \stackrel{df}{=} \left\{n \in N \colon \exists_{m \in W_x} \left(3 \cdot \frac{n}{2^{expo(n,2)}} + 1 = m \right)\right\} \tag{Wx+1}$$

A couple of observations will be used in the sequel.

Remark 3.1. Among properties of sets W_i we find

1. Each set W_i is infinite. If it contains a number n then the number $2n \in W_i$.
2. The set W_0 contains one odd number 1. All other sets contain infinitely many odd numbers. For if $a = 2j + 1$ and $a \in W_i$ then $4a + 1 \in W_i$.
3. Define the function $f \colon N \to N$ as follow: $f(n) = \begin{cases} n \div 2 & \text{when } n \text{ is even} \\ 3n + 1 & \text{when } n \text{ is odd } n \neq 1 \\ undefined & \text{when } n = 1 \end{cases}$
4. Let $x > 0$ be a natural number. Let the sequence $\{o_j\}_{i \in N}$ contain all odd numbers that are in the set W_x. Let $S_{o_j} = \left\{2^i \cdot o_j\right\}$ be the set. Every set W_x may be partitioned as follow $W_x = \bigcup_{j=0}^{\infty} S_{o_j}$. If $j \neq j'$ then $S_{o_j} \cap S_{o_{j'}} = \emptyset$
5. For every $i, j \in Nat$ if $i \neq j$ then $W_i \cap W_j = \emptyset$.
6. For every $n, j \in Nat$ if $n \in W_j \wedge j > 1 \wedge n \mod 2 = 1$ then $3n + 1 \in W_{j-1}$.
7. The sequence of sets $\left\{\bigcup_{i=0}^{j} W_i\right\}_{j \in N}$ is monotone, increasing.

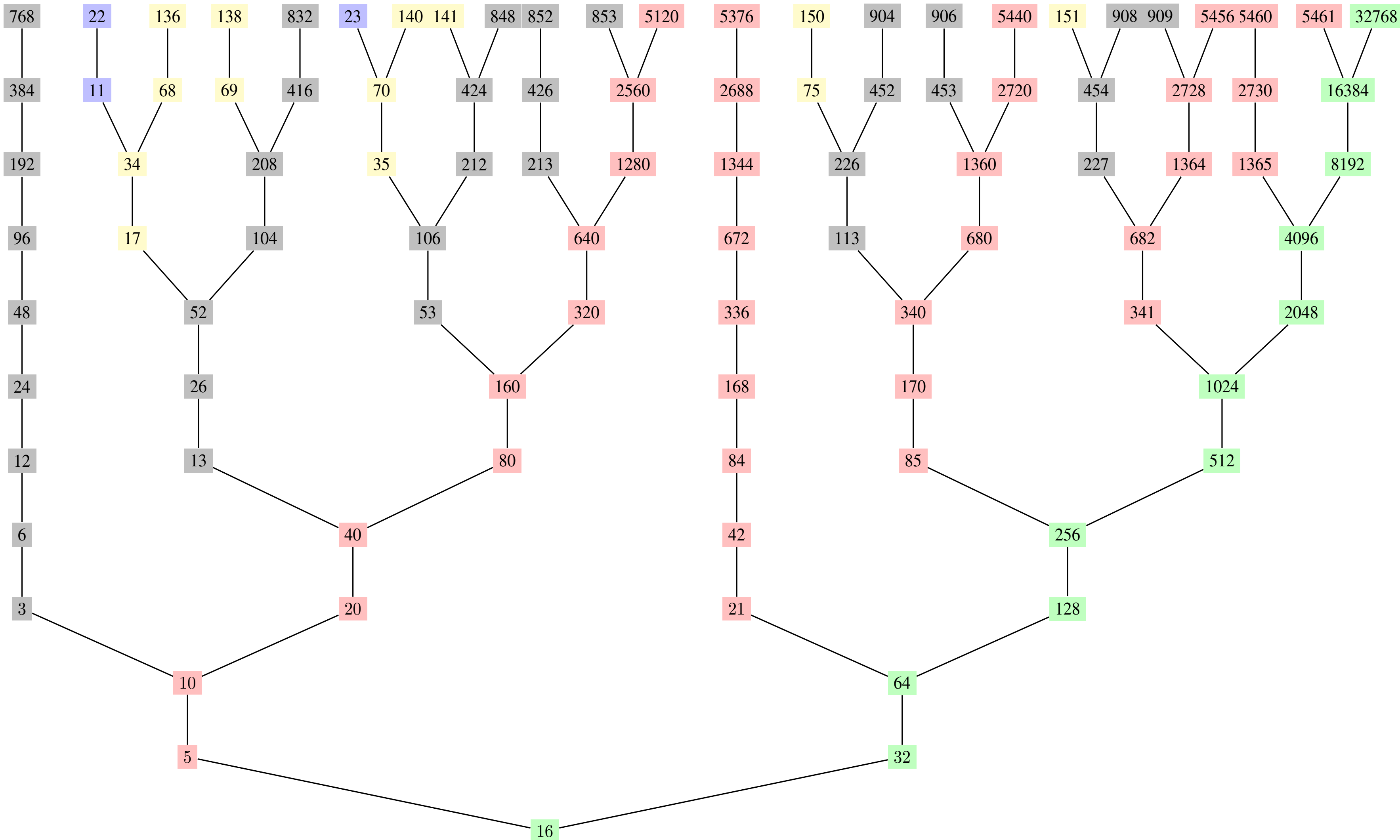

Figure 2. Strata $W_0 - -W_4$ of Collatz tree

Let s be a variable not occurring in algorithm $Gr1$. The following lemma states the partial correctnes of the algorithm $Gr1$ w.r.t. precondition $s = n$ and postcondition $s \in W_i$.

Lemma 3.3. Algorithm $Gr1$ computes the number i of storey W_i of number n,

$$\{Gr1\}(true) \implies \big((s = n) \implies \{Gr1\}(s \in W_i)\big)$$

Next, we present another algorithm $Gr2$ and a lemma.

```
     var n,l,i,x,y,z :integer ; k,m :arrayof integer;
           i := 0; l := 0;
     Γ2:   while even(n) do n:= n ÷ 2; l := l + 1 od ;
           z, k_i := l; m_i:=n; y := 0;
Gr2: while m_i ≠1 do
           n:= 3*n+1; i := i + 1; l := 0 ;
     Δ2:   while even(n) do n:= n ÷ 2; l := l + 1 od ;
           k_i := l; m_i:=n; z := z + k_i; y := 3 * y + 2^z; x := i
     od
```

Lemma 3.4. Algorithm $Gr2$ has the following properties:
(*i*) Both algorithms $Gr1$ and $Gr2$ are equivalent with respect to the halting property.
(*ii*) Formula φ : $\boxed{n \cdot 3^i + y = m_i \cdot 2^z}$ is an invariant of the program $Gr2$ i.e. the formulas (2) and (3)

$$\{\Gamma_2\}\,(n \cdot 3^i + y = m_i \cdot 2^z) \tag{2}$$

$$(n \cdot 3^i + y = m_i \cdot 2^z) \implies \{\Delta_2\}(n \cdot 3^i + y = m_i \cdot 2^z) \tag{3}$$

are theorems of the algorithmic theory of numbers $\mathcal{ATN}$.

Proof:
Proofs of formulas (2), (3) are easy, it suffices to apply the axiom of assignment instruction Ax_{18}, □

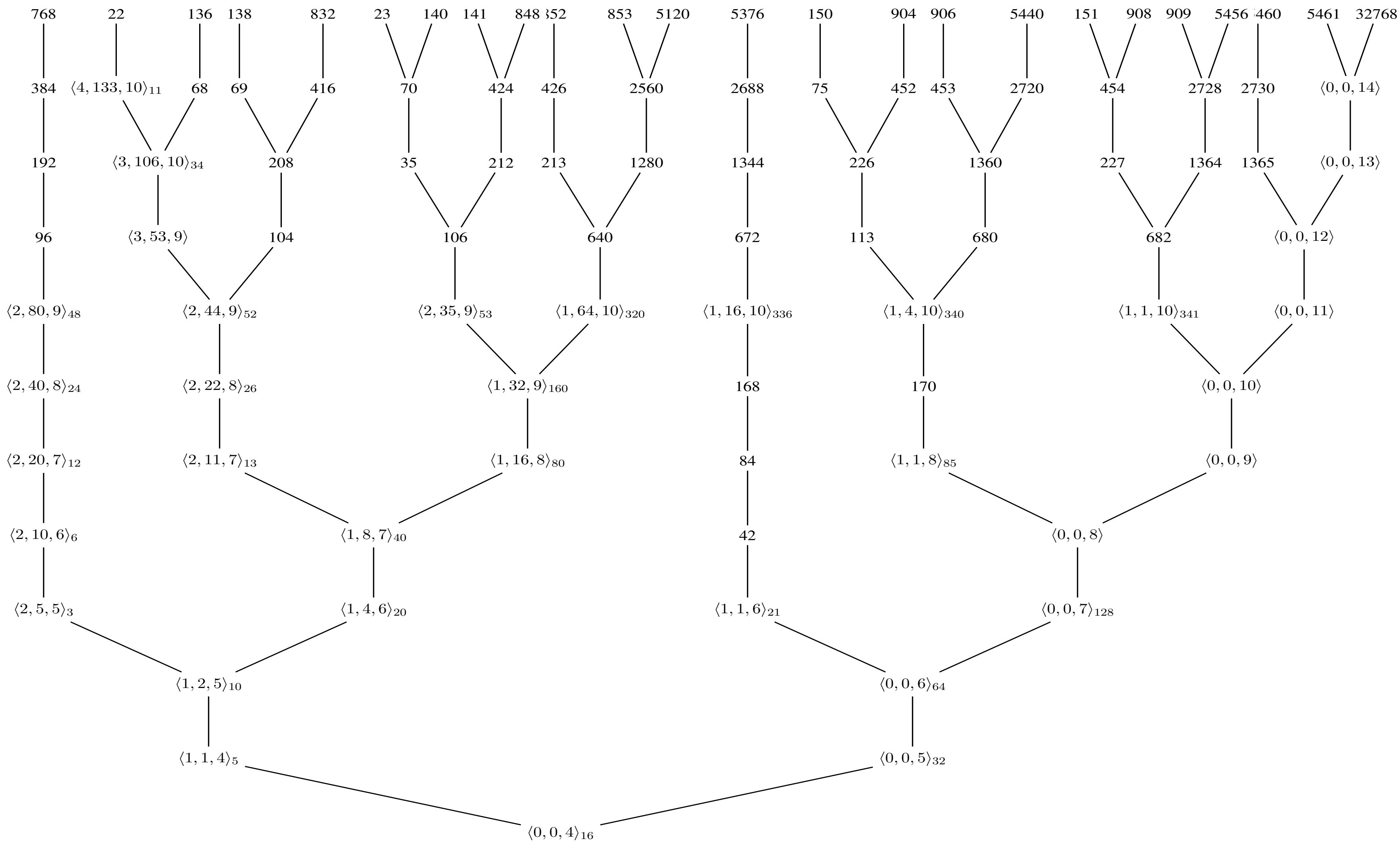

Figure 3. Tree of triples (levels 4 – 15)

Subsequent algorithm $Gr3$ exposes the history of the calculations of x, y, z.

$$
Gr3: \boxed{\begin{array}{l}
\textbf{varn}, i, aux, mn : integer;\ k, m, X, Y, Z : \textbf{arrayof} integer; \\
\Gamma_3 : \boxed{\begin{array}{l} i := 0;\ k_i := exp(\mathrm{n}, 2);\ m_i := \frac{\mathrm{n},}{2^{k_i}};\ mn := m_i; \\ Z_i := k_0;\ Y_i, X_i := 0; \end{array}} \\
\textbf{while}\ n \cdot 3^i + Y_i \neq 2^{Z_i}\ \textbf{do} \\
\Delta_3 : \boxed{\begin{array}{l} aux := 3 * m_i + 1; \qquad i := i + 1; \\ k_i := exp(aux, 2);\ m_i := aux/2^{k_i};\ mn := m_i; \\ Y_i := 3Y_{i-1} + 2^{Z_{i-1}};\ Z_i := Z_{i-1} + k_i;\ X_i := i \end{array}} \\
\textbf{od}
\end{array}}
$$

See some properties of the algorithm $Gr3$.

Lemma 3.5. Both algorithms $Gr2$ and $Gr3$ are equivalent with respect to the halting property. For every element n after each i-th iteration of algoritm $Gr3$, the following formulas are satisfied

$$
\begin{array}{ll}
\varphi :\ n \cdot 3^i + Y_i = m_i \cdot 2^{Z_i} & X_i = i \\
Z_i = \sum\limits_{j=0}^{i} k_j & Y_i = \sum\limits_{j=0}^{i-1} \left(3^{i-1-j} \cdot 2^{Z_j} \right)
\end{array}
$$

where the sequences $\{m_i\}$and $\{k_i\}$ are determined by the recurrence (rec2).
in other words, the following formula is valid in the structure $\mathfrak{N}$

$$\mathfrak{N} \models \Gamma_3 \bigcap \{\textbf{if}\ m_i \neq 1\ \textbf{then}\ \Delta_3\ \textbf{fi}\}\, \varphi$$

Remark 3.2. Hence, for every element n algorithm $Gr3$ calculates an increasing, monotone sequence of triples $\langle i\,(= X_i), Y_i, Z_i \rangle$.

Remark 3.3. We can say informally that the algorithm $Gr3$ performs as follow

$i := 0;$
while $n \notin W_i$ **do** $i := i + 1$ **od**

Note, $\{Gr_3\}\,(n \in W_i)$

3.1. Hotel Collatz

Hotel contains rooms of any natural number. Let $n = 2^i \cdot (2j + 1)$. It means that the room number n is located in tower number j on the floor number i . Each tower is equipped with an elevator (shown as a green line). Moreover, each tower is connected to another by a staircaise that connects numbers $k = 2j + 1$ and $3k + 1$. This is shown as a red arrow $\overrightarrow{\langle k, 3k + 1 \rangle}$.

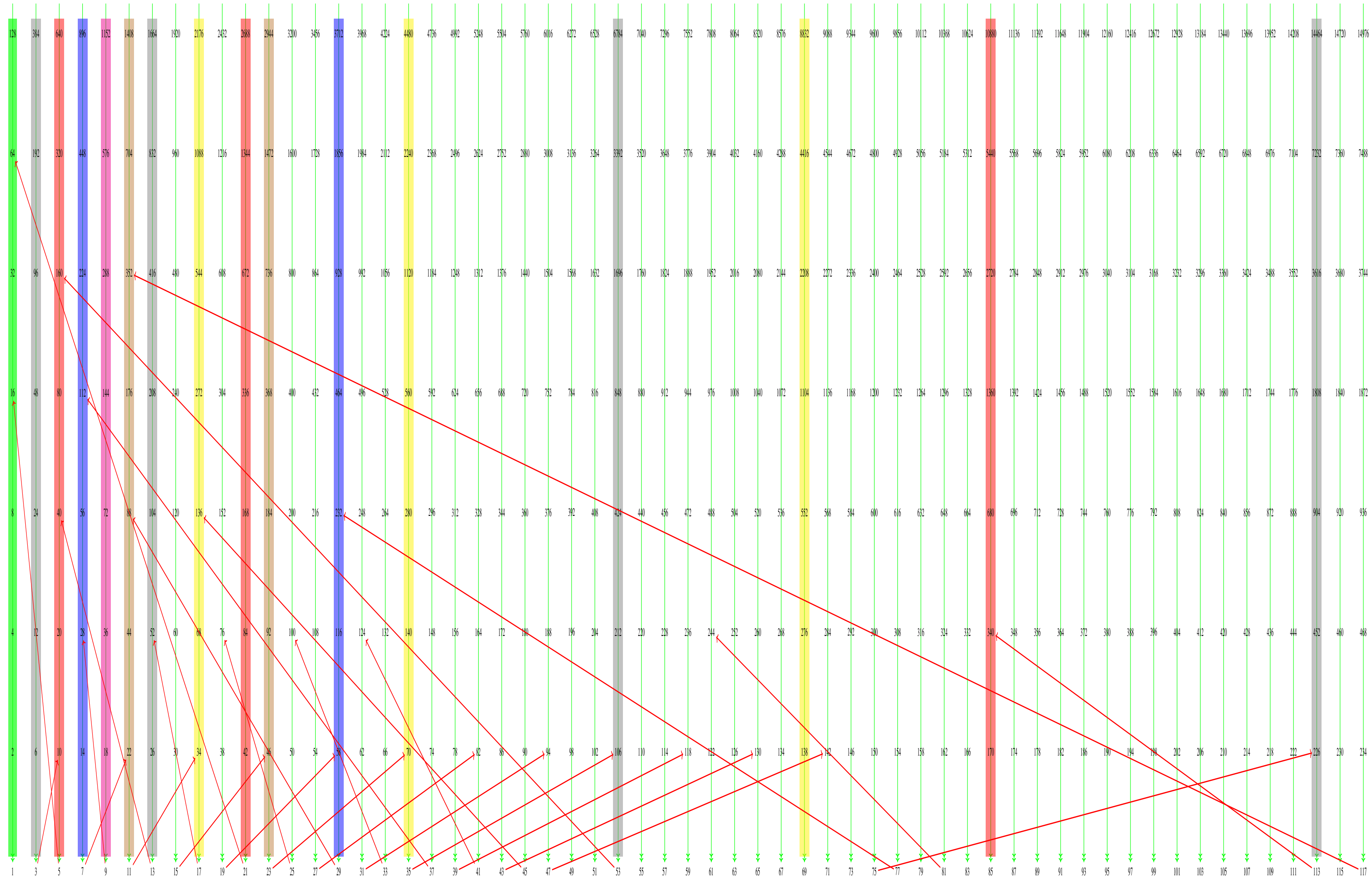

Figure 4. Hotel Collatz

Definition 3.2. (Hotel Collatz)
The graph $HC = \langle V, E \rangle$ is defined as follows

$V = N$ i.e. the set of vertices is the set of standard, reachable, natural numbers
$E = \{\overrightarrow{\langle k, p \rangle} : \exists_p\, k = p + p\} \cup \{\overrightarrow{\langle k, 3k+1 \rangle} : \exists_p k = p + p + 1\}$ are edges of the graph

Note. Don't forget, our drawing is only a small fragment of the infinite HC structure. The picture shows a small part of red arrows. We drew only those red arrows that fit entirely on a page.

Conjecture 3.1. The hotel Collatz is an infinite, connected, acyclic graph, i.e. it is a tree. Number 1 is the root of the tree.

Making use of the definition 3.1 one can formulate the following

Conjecture 3.2. The set $\overline{W} \stackrel{df}{=} \bigcup_{x \in N} W_x$ is a partition of the set N of nodes of Hotel Collatz.

4. On finite and infinite computations of Collatz algorithm

QUESTION Can some computation contain both reachable and unreachable elements?
No. The subset of reachable elements is closed with respect of division by 2 and multiplication by 3. The same observation applies to the set of unreachable elements.
We know, cf. subsection 6.1 that computations of nonreachable elements are infinite.

4.1. Finite computations

Let $\mathfrak{M} = \langle M; 0, 1, +, = \rangle$ be any algebraic structure that is a model of elementary theory of addition of natural numbers, c.f. subsection 6.2.
Denotation. Let $\theta(x, y)$ be a formula. The expression $(\mu x)\theta(x, y)$ denotes the least element $x(y)$ such that the value of the formula is truth.
EXAMPLE. The formula $(\mu x)(x + x = y \vee x + x + 1 = y)$ defines the operation $\div$, i.e. $x = y \div 2$.

The following lemma gathers the facts established earlier.

Lemma 4.1. Let n be an arbitrary element of the structure $\mathfrak{M}$. The following conditions are equivalent

(*i*) The sequence $n_0 = n$ and $n_{i+1} = \begin{cases} n_i \div 2 & \text{when } n_i \mod 2 = 0 \\ 3n_i + 1 & \text{when } n_i \mod 2 = 1 \end{cases}$ determined by the recurrence (rec1) contains an element $n_j = 1$

(*ii*) The computation of the algorithm Cl is finite.

(*iii*) The sequence $m_0 = \frac{n}{2^{k_0}}$ and $m_{i+1} = \frac{3m_i+1}{2^{k_i}}$ determined by the recurrence (rec2) stabilizes, i.e. there exist l such that $m_k = 1$ for all $k > l$

(*iv*) The computation of the algorithm Gr is finite.

(*v*) The computation of the algorithm $Gr1$ is finite and the subsequent values of the variables M_i and K_i satisfy the recurrence (rec2) .

(*vi*) The computation of the algorithm $Gr2$ is finite and the subsequent values of the variables m_i and k_i satisfy the recurrence (rec2). The formula $n \cdot 3^x + y = m_i \cdot 2^z$ holds after each iteration of **while** instruction, i.e. it is the invariant of the program Δ_2. The final valuation of variables x, y, z and n satisfies the equation $n \cdot 3^x + y = 2^z$.

(vii) The computation of the algorithm $Gr3$ is finite.
The subsequent values of the variables m_i and k_i satisfy the recurrence (rec2) .
The subsequent values of the variables X_i, Y_i, Z_i form a monotone, increasing sequence of triples.
The formula $n \cdot 3^{X_i} + Y_i = m_i \cdot 2^{Z_i}$ is satisfied after each i-th iteration of the program $Gr3$, i.e. the value of the following exression $\{\Gamma_3;\ \Delta_3^i\}(X_i + Z_i)$ is the total number of operations excuted. The value of the variable Y_i encodes the history of the computation till the i-th iteration of Δ_3

Suppose that for a given element n the computationof algorithm $Gr2$ is finite.
Let $\bar{x} = (\mu x)\left(n \cdot 3^x + \left[\sum_{j=0}^{x-1} \left(3^{x-1-j} \cdot 2^{\sum_{l=0}^{j} k_l}\right) \right] = 2^{\sum_{j=0}^{x} k_j} \right)$. Put $\bar{y} = \sum_{j=0}^{\bar{x}-1} \left(3^{\bar{x}-1-j} \cdot 2^{\sum_{l=0}^{j} k_l}\right)$ and $\bar{z} = \sum_{j=0}^{\bar{x}} k_j$.
We present the algorithm IC', which is a slightly modified version of the algorithm IC devised in [MS21] .

$$IC' : \left\{ \begin{array}{l} \textbf{var}\ \mathrm{x}, \mathrm{y}, \mathrm{z}, k : integer, Err : Boolean; \\ Err := false; \\ \textbf{while}\ \mathrm{x}+\mathrm{y}+\mathrm{z} \neq 0\ \ \textbf{do} \\ \quad Tr : \boxed{\begin{array}{l} \textbf{if}\ \big(odd(\mathrm{y}) \wedge ((\mathrm{x}=0)\ \vee\ (\mathrm{y} < 3^{\mathrm{x}-1}))\big) \\ \quad \textbf{then}\ Err := true;\ \textbf{exit} \\ \textbf{fi}; \\ \ \mathrm{x} := \mathrm{x}-1;\ \mathrm{y} := \mathrm{y} - 3^{\mathrm{x}};\ k := expo(\mathrm{y}, 2); \\ \mathrm{y} := \dfrac{\mathrm{y}}{2^k};\ \mathrm{z} := \mathrm{z} - k; \end{array}} \\ \textbf{od} \end{array} \right\} \qquad \text{(IC')}$$

We observe the following fact

Lemma 4.2. For every element n

$$(n = b) \wedge \{Gr2\}\Big(\big((x = \bar{x} \wedge y = \bar{y} \wedge z = \bar{z}) \wedge (b \cdot 3^{\bar{x}} + \bar{y} = 2^{\bar{z}})\big) \Rightarrow \{IC'\}(x = y = z = 0) \Big)$$

and

$$(x = \bar{x} \wedge y = \bar{y} \wedge z = \bar{z}) \wedge (b \cdot 3^{\bar{x}} + \bar{y} = 2^{\bar{z}})) \wedge \\ \Big(\{IC'\}(x = y = z = 0) \implies (n = b) \implies \{Gr2\}(x = \bar{x} \wedge y = \bar{y} \wedge z = \bar{z}) \Big)$$

The contents of this lemma are best explained by the commutativity of the diagram below.

$$\begin{tikzcd} \boxed{\left(\frac{n}{b}\right)\left(\frac{x\ y\ z}{0\ 0\ 0}\right)} \arrow[r, bend left, "Gr2(n)"] & \boxed{\left(\left(\frac{x\ y\ z}{\bar{x}\ \bar{y}\ \bar{z}}\right)\left(\frac{n}{1}\right)\right.} \wedge (b \cdot 3^{\bar{x}} + \bar{y} = 2^{\bar{z}})) \arrow[l, bend left, "IC(x\,y\,z)"] \end{tikzcd}$$

Proof:
The proof makes use of two facts:

1) $\quad even(n) \equiv even(\sum_{j=0}^{x-1} \left(3^{x-1-j} \cdot 2^{\sum_{l=0}^{j} k_l}\right))$

2) $\sum_{j=0}^{x-1} \left(3^{x-1-j} \cdot 2^{\sum_{l=0}^{j} k_l}\right) = 2^{k_0} \cdot (3^{x-1} + 2^{k_1} \cdot (3^{x-2} + 2^{k_2} \cdot (\cdots + 2^{k_x} \cdot 3^0)))$

One can prove this lemma by induction w.r.t. number of encountered odd numbers.

The thesis of the lemma is very intuitive. Look at the Collatz hotel Fig. 4. The lemma states that for every room number n the two conditions are equivalent (1) there is a path from room number n to the exit, which is located near room number 1, (2) there is a path from entry to the hotel near room number 1 to the room number n. It is clear that such a path must be complete, no jumps are allowed. □

Lemma 4.3. For every element n the following conditions are equivalent

(i) computation of Collatz algorithm Cl is finite,

(ii) there exists the LEAST element x such that the following equality holds

$$n \cdot 3^x + \left(\sum_{j=0}^{x-1} 3^{x-1-j} \cdot 2^{\sum_{l=0}^{j} k_l}\right) = 2^{\sum_{j=0}^{x} k_j}. \tag{Mx}$$

where the sequence $\{k_i\}$ is determined by the element n in accordance to the recurrence (rec2).

Proof:
The implication (i) ⇒ (ii) follows from lemma 4.1(vii).
Consider the inverse implication (ii) ⇒ (i). If (ii) holds, then the computation of program Gr_3 reaches 1 after x-th iteration of internal instruction Δ_3. The value of m_x is then 1. The computation performs x multiplications and $z = \sum_{j=0}^{x} k_j$ divisions.
The value of $y = \left(\sum_{j=0}^{x-1} (3^{x-1-j} \cdot 2^{\sum_{l=0}^{j} k_l})\right)$ codes the history of the computation. □

Which means: x_0 is the number of multiplication by 3, $z = \sum_{j=0}^{x} k_j$ is the total number of divisions by 2 and for every $0 \le j \le x-1$ the number k_j is the number of divisions by 2 excuted in between the j-th and $j+1$-th execution of multiplication by 3.

The algorithm Cl executes $x + z$ iterations. The lemma 4.3 gives the halting formula i.e. a satisfactory and necessary condition for the computation of the Collatz program to be finite.
We shall summarize the considerations on finite computations in the following commutative diagram.

4.2. Infinite computations

Do infinite computations exist?
There are two answers *yes* and *no*.

Yes	Imagine your computer system is (*maliciously*) handled by a hacker. This can be done by preparing its hardware or software (e.g. someone modifies the class Int in Java). To hide the damage from the user, the hacker may come with a correct proof that all axioms of natural numbers (e.g. of Presburger's system) are valid. Yet, for many n execution of the Collatz algorithm will not terminate. See subsections 4.4 and 6.1.
No	If argument of Collatz procedure is a *standard* (i.e. reachable) natural number then the computation is finite.

Our aim is to prove the lemma

Lemma 4.4. (noCycle)
There is no reachable, natural number n such that Collatz computation for n is a cycle.

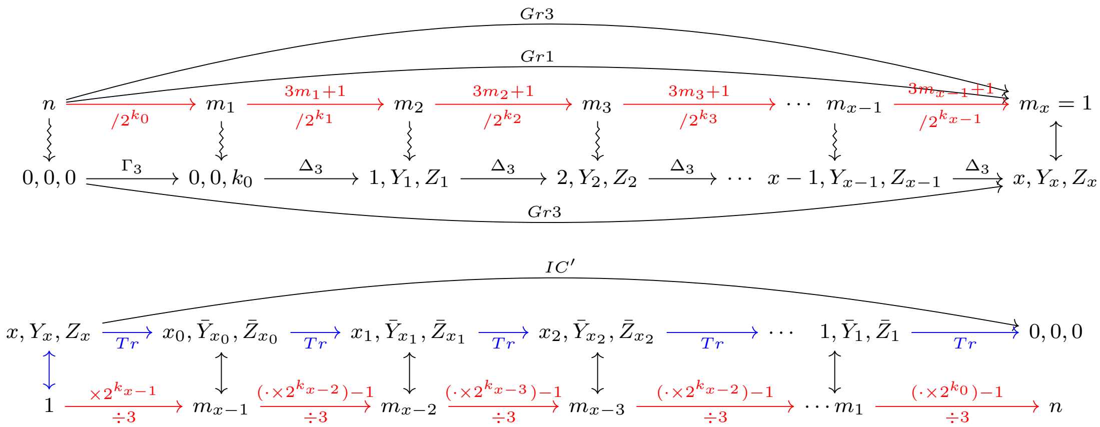

Figure 5. CASE OF FINITE COMPUTATION ILLUSTRATED
Upper row, (with red arrows) represents computation of $Gr1$,
elements k_i and m_i are calculated in accordance with the recurrence (rec2)
rows 1 and 2 show computation of $Gr3$, the subsequent triples are $X_{i+1} = i+1, Y_{i+1} = 3Y_i + 2^Z_i, Z_{i+1} = Z_i + k_i$
third row (blue arrows) shows computation of algorithm IC on triples, $\bar{Y}_x = Y_x$ and $\bar{Z}_x = Z_x$ and for $i = x, \ldots, 1$ we have $\bar{Z}_{i-1} = \bar{Z}_i - k_i$ and $\bar{Y}_{i-1} = (\bar{Y}_i/2^{k_i}) - 3^{i-1}$

Proof:
Suppose that there exists a cycle and certain reachable number n is in this cycle.
Let q be the length of the cycle. Remark, in the following formulas (4) – (7) the precedent of the implication holds and implies the inequality in its successor.
Note. Whenever the operator $-$ of subtraction appears the first argument is bigger than the second one. Similarly, whenever the operator $\div$ appears the dividend is bigger than divisor. In the sequel we make sure that $a > b$ before writing $a \div b$.

$$n \cdot 3^0 + 0 = n \cdot 2^0 \wedge n > 1 \tag{4}$$

$$n \cdot 3^q + Y_q = n \cdot 2^{Z_q} \implies n > (2^{Z_q} - Y_q) \div 3^q \tag{5}$$

$$n \cdot 3^{2q} + Y_{2q} = n \cdot 2^{Z_{2q}} \implies n > (2^{Z_{2q}} - Y_{2q}) \div 3^{2q} \tag{6}$$

$$\cdots$$

$$n \cdot 3^{r \cdot q} + Y_{r \cdot q} = n \cdot 2^{Z_{r \cdot q}} \implies n > (2^{Z_{r \cdot q}} - Y_{r \cdot q}) \div 3^{r \cdot q} \qquad \text{where } r \in N \tag{7}$$

$$\cdots$$

From the equality $n \cdot 3^q + Y_q = n \cdot 2^{Z_q}$ we infer inequality $3^q < 2^{Z_q}$.
From inequality $3^q < 2^{Z_q}$ and equality $3^q + (n-1) \cdot 3^q + Y_q = 2^{Z_q} + (n-1) \cdot 2^{Z_q}$ we infer inequality

$$3^q + Y_q < 2^{Z_q} \tag{8}$$

Since the computation is a cycle we have $Z_{2q} = Z_q + Z_q$ and $Y_{2q} = 3^q \cdot Y_q + 2^{Z_q} \cdot Y_q$.
Similarly, $Z_{(r+1)q} = Z_{rq} + Z_q$ and $Y_{(r+1)q} = 3^q \cdot Y_{rq} + 2^{Z_{rq}} \cdot Y_q$.
Next, we are going to prove that the sequence

$$1, (2^{Z_q} - Y_q) \div 3^q, (2^{Z_{2q}} - Y_{2q}) \div 3^{2q}, \cdots, (2^{Z_{r \cdot q}} - Y_{r \cdot q}) \div 3^{r \cdot q}, (2^{Z_{(r+1) \cdot q}} - Y_{(r+1) \cdot q}) \div 3^{(r+1) \cdot q}, \cdots \tag{9}$$

is infinite, increasing sequence of reachable, natural numbers.
The proof is by induction with respect to r.
(**Base**). From inequality $3^q + Y_q < 2^{Z_q}$ we obtain $3^q < 2^{Z_q} - Y_q$ and

$$1 < (2^{Z_q} - Y_q) \div 3^q.$$

(**Induction step**). Without loss of the generality we can consider the step from $r = 1$ to $r = 2$.
Once again we start with the valid inequality (8).

$$3^q + Y_q < 2^{Z_q}.$$

Our next inequality is valid too

$$3^q \cdot 2^{Z_q} + 2^{Z_q} \cdot Y_q < 2^{Z_{2q}}.$$

Making use of remarks made earlier (just below the inequality (8)) we get

$$(2^{Z_q} - Y_q) \cdot 3^q < 2^{Z_{2q}} - Y_{2q}$$

which is equivalent to the inequality

$$(2^{Z_q} - Y_q) \div 3^q < (2^{Z_{2q}} - Y_{2q}) \div 3^{2q}$$

Hence, the induction step is proved. (The reader may wish to verify our reasoning for the step from r-th to $r+1$-th iteration of the cycle.)
The sequence (9) is increasing, infinite. The number n is bigger than any element of this sequence. Therefore, the number n is bigger than any reachable number. This contradicts the assumption that number n is reachable. □

Now, we are going to prove the following

Lemma 4.5. (no Divergent computations)
If for certain number n computation of Collatz algorithm is infinite then n is not a reachable, natural number.

We remark that the semantical property: *for a given element n the computation of Collatz algorithm may be continued at will* , i.e. it is an infinite computation, is expressed by the following formula

$$\{\Gamma_3\} \bigcap \{\Delta_3\} (mn \neq 1) \qquad (\Pi)$$

where denotations Γ_3 and Δ_3 and mn were introduced on page 11, see program Gr_3.

We verify this assertion in a few steps

1° Define an infinite sequence of formulas $\{\vartheta_i\}_{i=0}^{\infty}$ as follows

$$\vartheta_i \stackrel{df}{=} \{\Gamma_3\} \{\Delta_3\}^i (m_i = 1)$$

2° Let V be the set of all variables appearing in program Gr_3. We put $v_0(n) = n_0$, the values of the remaining variables are not important.
For $i = 1, 2, \dots$ we put $v_i = (\{\Gamma_3\} \{\Delta_3\}^i)_{\mathfrak{N}}(v_0)$

3° By the definition of semantics of general iteration quantifier we have the following equation

$$\left(\{\Gamma_3\} \bigcap \{\Delta_3\} (mn \neq 1)\right)_{\mathfrak{N}} (v) = \mathbf{g.l.b.} \{(\vartheta_i)_{\mathfrak{N}} (v_0)\}_{i \in N} = \mathbf{g.l.b.} \{(m_i \neq 1)_{\mathfrak{N}} (v_i)\}_{i \in N}$$

For it says: after every iteration of subprogram $\{\Gamma_3; \Delta_3^i\}$ of the program Gr_3 the value of the variable mn is not equal 1.
Now, he following formula

$$\{i := 0;\} \bigcap \{i := i + 1\} (n \cdot 3^i + \sum_{j=0}^{i-1} 3^{i-1-j} \cdot 2^{\sum_{l=0}^{j} k_l} \neq 2^{\sum_{l=0}^{i} k_l}) \qquad (\Xi)$$

where the values of k_l are defined by recurrence rec2 see page 6 does express the semantical property: *for every reachable, natural number i the inequality $(n\cdot 3^i+\sum_{j=0}^{i-1} 3^{i-1-j}\cdot 2^{Z_j} \neq 2^{Z_i})$ holds.* By transposition, it is is equivalent to following statement: *there is no reachable number i such that the equality $(n \cdot 3^i + \sum_{j=0}^{i-1} 3^{i-1-j} \cdot 2^{Z_j} = 2^{Z_i})$ holds.* In other words

$$l.u.b.(n \cdot 3^i + \sum_{j=0}^{i-1} 3^{i-1-j} \cdot 2^{Z_j} = 2^{Z_i})_{\mathfrak{N}}(v) = truth \qquad \text{where } i = 1, 2, \ldots,$$

Hence our goal is to prove the implication $(\Pi \implies \Xi)$.
We begin with simple observation that every formula of the following set S_a is a tautology

$$S_a \stackrel{df}{=} \{\ (\{\Gamma_3\}\,\{\Delta_3\}^r\,(m_i \neq 1) \implies \{\Gamma_3\}\,\{\Delta_3\}^r\,(m_i \neq 1))\ \}_{r\in N}$$

here $\underline{r} = \underbrace{1+1+1+\cdots+1}_{r\ times}$ is any reachable, natural number.
For every natural number r the following equivalence is a theorem of algorithmic theory of numbers $\mathcal{ATN}$

$$\mathcal{ATN} \vdash \big(\ \{\Gamma_3\}\,\{\Delta_3\}^r\,(m_i = 1) \Leftrightarrow \{\Gamma_3\}\,\{\Delta_3\}^r\,(n \cdot 3^i + \sum_{j=0}^{i-1} 3^{i-1-j} \cdot 2^{Z_j} = 2^{Z_i})\ \big)$$

see lemma 4.1(*vii*), page 13. Hence the following set S_b consists of theorems of theory $\mathcal{ATN}$

$$S_b \stackrel{df}{=} \left\{\ (\{\Gamma_3\}\,\{\Delta_3\}^r\,(m_i \neq 1) \Rightarrow \{\Gamma_3\}\,\{\Delta_3\}^r\,(n \cdot 3^i + \sum_{j=0}^{i-1} 3^{i-1-j} \cdot 2^{Z_j} \neq 2^{Z_i}))\ \right\}_{r\in N}$$

Look at programs $\{\Gamma_3\}$ and $\{\Delta_3\}$, on page 11. Note that first program contains instruction $i := 0$ and the second program contains the instruction $i := i+1$, other instructions can be dropped. Hence the set S_c of formulas

$$S_c \stackrel{df}{=} \left\{\ \{\Gamma_3\}\,\{\Delta_3\}^r\,(n_i \neq 1) \Rightarrow \{i := 0\}\,\{i := i+1\}^r\,(n \cdot 3^i + \sum_{j=0}^{i-1} 3^{i-1-j} \cdot 2^{Z_j} \neq 2^{Z_i})\right\}_{r\in N}$$

contains only theorems of theory $\mathcal{ATN}$. Now by induction with respect to r we can prove that every formula of the set S_d is a theorem of theory $\mathcal{ATN}$. I.e. we introduce the universal iteration quantifier $\bigcap$ in the precedent of every implication of the set S_c of theorems of theory $\mathcal{ATN}$.

$$S_d \stackrel{df}{=} \left\{\{\Gamma_3\}\bigcap\{\Delta_3\}\,(m_i \neq 1) \Rightarrow \{i := 0\}\,\{i := i+1\}^r\,(n \cdot 3^i + \sum_{j=0}^{i-1} 3^{i-1-j} \cdot 2^{Z_j} \neq 2^{Z_i})\right\}_{r\in N}$$

In the proof we used axiom Ax_{23} of calculus of programs, de Morgan's laws (case of infinite operations in a Boolean algebra) and propositional calculus, c.f. page 30.
Now, we are ready to use the inference rule R_5, see page 30 and to obtain the desired theorem of algorithmic theory $\mathcal{ATN}$

$$\underbrace{\{\Gamma_3\}\bigcap\{\Delta_3\}\,(m_i \neq 1)}_{\Pi} \Rightarrow \overbrace{\{i := 0\}\bigcap\{i := i+1\}\,(n \cdot 3^i + \sum_{j=0}^{i-1} 3^{i-1-j} \cdot 2^{Z_j} \neq 2^{Z_i})}^{\Xi} \tag{10}$$

which reads:
if for a given number n the computation of Collatz algorithm is infinite then the number n is an unreachable element of non-standard model $\mathfrak{M}$ of Presburger arithmetic.
Why it is so?

- the formula Ξ is eqivalent to

$$\neg\,\{i := 0\}\bigcup\{i := i+1\}\,(n \cdot 3^i + \sum_{j=0}^{i-1} 3^{i-1-j} \cdot 2^{Z_j} = 2^{Z_i})$$

- If the formula Ξ holds then the value of i such that the equation $n \cdot 3^i + \sum_{j=0}^{i-1} 3^{i-1-j} \cdot 2^{Z_j} = 2^{Z_i}$ is satisfied, must be an unreachable number.
- Note, if the equation $n \cdot 3^i + \sum_{j=0}^{i-1} 3^{i-1-j} \cdot 2^{Z_j} = 2^{Z_i}$ holds then the elements n and i are either both reachable or both unreachable numbers,c.f. remark 6.1 on page 24.

This ends the proof of the lemma 4.5.

4.3. Collatz theorem

Till now we proved that Collatz conjecture is <u>valid</u> in the structure $\mathfrak{N}$ of standard (reachable) natural numbers as it is witnessed by the following lemma.

Lemma 4.6. Let n be any standard element of the structure $\mathfrak{N}$. The computation of Collatz algorithm Cl that begins with n is finite.

Proof:
The proof follows immediately from the lemmas 4.3 and 4.5. □

Corollary 4.1. Three conjectures 2.1, 3.1 and 3.2 formulated above are valid statements.

We are ready to prove the main result of this paper.

Theorem 4.1. The formula Θ_{CL} is a theorem of the algorithmic theory of natural numbers $\mathcal{ATN} \vdash \Theta_{CL}$.

$$\underbrace{\{n:=1\}\bigcap\{n:=n+1\}}_{\forall_{n\in N,\, n\geq 1}} \left\{ \begin{array}{l} \textbf{while } n\neq 1 \textbf{ do} \\ \quad \textbf{if } odd(n) \textbf{ then} \\ \quad\quad n:=3*n+1 \\ \quad \textbf{else} \\ \quad\quad n:=n\div 2 \\ \quad \textbf{fi} \\ \textbf{od} \end{array} \right\} (n=1) \tag{Θ_{CL}}$$

Proof:
The following formula Ψ_{CL} is <u>valid</u> in the algebraic structure $\mathfrak{N}$ i.e. in the standard model of theory $\mathcal{ATN}$.

$$\bigcup \left\{ \begin{array}{l} \textbf{if } n\neq 1 \textbf{ then} \\ \quad \textbf{if } odd(n) \textbf{ then} \\ \quad\quad n:=3*n+1 \\ \quad \textbf{else} \\ \quad\quad n:=n\div 2 \\ \quad \textbf{fi} \\ \textbf{fi} \end{array} \right\} (n=1) \tag{Ψ_{CL}}$$

For we have established two facts

- if the argument n of the Collatz program is an unreachable element then the computation is infinite, c.f. remark (6.1 on page 24),
- if for a certain argument n the computation of the Collatz program is infinite then the element is an unreachable number c.f. (lemma 4.6).

Now, we ought to precede the formula Ψ_{Cl} by a general quantifier $\forall_{n\in N}$. However, the membership predicate $\in$ does not belong to the language of any theory considered so far.
Fortunately, the expression $\boxed{\forall_{n\in N, n\geq 1}\, \alpha(n)}$ can be replaced by

$$\underbrace{\{n':=1\}\bigcap\{n':=n'+1\}\,\big(n=n'\wedge\alpha(n)\big)}_{\forall_{n\in N,\, n\geq 1}}$$

For the correctness of this replacement you can consult [Sal23], thm 6.1, page 15.
Hence, we know that the following formula

$$\underbrace{\{n' := 1\} \bigcap \{n' := n' + 1\}}_{\forall_{n \in N,\, n \geq 1}} \Big((n = n') \wedge \bigcup \left\{ \begin{array}{l} \textbf{if } n \neq 1 \textbf{ then} \\ \quad \textbf{if } odd(n) \textbf{ then} \\ \quad\quad n := 3 * n + 1 \\ \quad \textbf{else} \\ \quad\quad n := n \div 2 \\ \quad \textbf{fi} \\ \textbf{fi} \end{array} \right\} (n = 1) \Big) \qquad (\Theta'_{CL})$$

is valid in the structure $\mathfrak{N}$ of standard, reachable natural numbers . Note, the formulas (Θ'_{CL}) and (Θ_{CL}) are equivalent, they express the halting property of Collatz algorithm.

Now, we shall use two meta-mathematical facts, c.f. [MS87] pages 94 and 155.

Completenes thm For every consistent algorithmic theory $\mathcal{T}$, for any formula α, the formula is a theorem of $\mathcal{T}$ iff the formula α is valid in every model of theory $\mathcal{T}$.

Categoricity thm Every two models of the theory $\mathcal{ATN}$ are isomorphic.

From these observations we obtain the desired conclusion

$$\mathcal{ATN} \vdash \underbrace{\{n := 1\} \bigcap \{n := n + 1\}}_{\forall_{n \in N,\, n \geq 1}} \left\{ \begin{array}{l} \textbf{while } n \neq 1 \textbf{ do} \\ \quad \textbf{if } odd(n) \textbf{ then} \\ \quad\quad n := 3 * n + 1 \\ \quad \textbf{else} \\ \quad\quad n := n \div 2 \\ \quad \textbf{fi} \\ \textbf{od} \end{array} \right\} (n = 1) \qquad (\Theta_{CL})$$

□

Someone may ask: why this awkward prefix in the formula Θ_{CL}?
Note the difference between phrases $\forall_n$ and $\forall_{n \in N}$.

4.4. A counterexample

We argue, that the formulation of the Collatz problem requires more precision. For there are several algebraic structures that can be viewed as structure of natural numbers of addition. Some of them admit infinite computations of Collatz algorithm.
We recall less known fact: arithmetic (i.e. first-order theory of natural numbers) has standard (*Archimedean*) model $\mathfrak{N}$ as well as another *non-Archimedean* model $\mathfrak{M}$[3]. The latter structure allows for the existence of infinitely great elements.
Goedel's incompleteness theorem shows that there is no elementary theory T of natural numbers, such that every model is isomorphic to the standard model.
Two things are missing from the commonly accepted texts: 1) What do we mean by proof? 2) what properties of natural numbers can be used in the proof? We recall an algebraic structure $\mathfrak{M}$ that models [Grz71] all axioms of elementary theory of addition of natural numbers, yet it admits unreachable elements [Tar34]. It means that the model contains element ε, such that the computation of Collatz algorithm that starts with ε is infinite.

[3]A. Tarski [Tar34] confirms that S. Jaśkowski observed (in 1929) that the subset of complex numbers $M \stackrel{df}{=} \big\{a + bi \in \mathbb{C} \colon \big(a \in \mathbb{Z} \wedge b \in \mathbb{Q} \wedge (b \geq 0 \wedge (b = 0 \implies a \geq 0)\big)\big\}$ satisfies all axioms of Presburger arithmetic.

Example of a finite execution

$$\langle 13,0\rangle \overset{\times 3+1}{\to} \langle 40,0\rangle \overset{\div 2}{\to} \langle 20,0\rangle \overset{\div 2}{\to} \langle 10,0\rangle \overset{\div 2}{\to}, \langle 5,0\rangle \overset{\times 3+1}{\to} \langle 16,0\rangle \overset{\div 2}{\to} \langle 8,0\rangle \overset{\div 2}{\to} \langle 4,0\rangle \overset{\div 2}{\to} \langle 2,0\rangle \overset{\div 2}{\to} \langle 1,0\rangle$$

Example of an infinite execution

$$\langle 8,\frac{1}{2}\rangle \overset{\div 2}{\to} \langle 4,\frac{1}{4}\rangle \overset{\div 2}{\to} \langle 2,\frac{1}{8}\rangle \overset{\div 2}{\to} \langle 1,\frac{1}{16}\rangle \overset{\times 3+1}{\to} \langle 4,\frac{3}{16}\rangle \overset{\div 2}{\to} \langle 2,\frac{3}{32}\rangle \overset{\div 2}{\to} \langle 1,\frac{3}{64}\rangle, \overset{\times 3+1}{\to} \langle 4,\frac{9}{64}\rangle \overset{\div 2}{\to} \langle 2,\frac{9}{128}\rangle \overset{\div 2}{\to} \cdots$$

As you can guess, the data structure contains pairs $\langle k,w\rangle$ where k is an integer and w is a non-negative, rational number. The addition operation is defined componentwise. An element $\langle k,w\rangle$ is *even* if k is even, otherwise is *odd*.. A pair $\langle k,w\rangle$ divided by 2 returns $\langle k \div 2, w \div 2\rangle$.
The reader may prefer to think of complex numbers instead of pairs, e.g. $(2+\frac{9}{128}i)$ may replace the pair $\langle 2,\frac{9}{128}\rangle$.
The following observation seems to be of importance:.

Remark 4.1. *There exists an infinite computation* **c** *of Collatz algorithm in the structure* $\mathfrak{M}$ *, such that the computation* **c** *does not contain a cycle, and the sequence of pairs is not diverging into still growing pairs. The latter means, that there exist two numbers* $l_1 \in \mathcal{Z}$ *and* $l_2 \in \mathcal{Q}$*, such that for every step* $\langle k,v\rangle$ *of computation* **c***, the inequalities hold* $k < l_1 \wedge v < l_2$.

More details can be found in subsection 6.1.

5. Final remarks

Our message does not limit itself to the proof of Collatz theorem.
We show that the algorithmic language of program calculus is indispensable for expressing the semantic properties of programs. Halting property of program, axiomatic specification of data structure of natural numbers can not be expressed by (sets) of first-order formulas.
We show the potential of calculus of programs as a tool for

- specification of semantical properties of software and
- verification of software against some specifications.

We hope the reader will forgive us for a moment of insistence (is it a propaganda?).
Calculus of programs $\mathcal{AL}$ is a handy tool. For there are some good reasons to use the calculus of programs

(*i*) The language of calculus $\mathcal{AL}$ contains algorithms (programs) and *algorithmic formulas* besides terms and first-order formulas.

(*ii*) Any semantical property of an algorithm can be *expressed* by an appropriate algorithmic formula. Be it termination, correctness or other properties.

(*iii*) Algorithmic formulas enable to create complete, categorical *specifications* of data structures in the form of algorithmic theories.

(*iv*) Calculus of programs $\mathcal{AL}$ offers a *complete* set of tools for proving theorems of algorithmic theories.

For over 50 years we are studying the program calculus and use it in specification and verification of software. Some examples can be found in [MS87], [Sal23] and other publications.
In spite of appearance of ω-rules[4] one can verify proofs in an automatic way.

[4] I.e. the inference rules with enumerably many premises.

Historical remarks

Pàl Erdős said on Collatz conjecture: *"Mathematics may not be ready for such problems."*
We disagree. In our opinion a consortium of Alfred Tarski, Kurt Goedel and Stephen C. Kleene was able to solve the Collatz conjecture in 1937. (Tarski was advisor of M. Presburger and S. Jaśkowski.)

- Mojżesz Presburger has proved the completeness and decidability of arithmetic of addition of natural numbers in 1929.
- In the same year Stanisław Jaśkowski found a non-standard model of Presburger theory (see a note of A. Tarski of 1934).
- Kurt Gödel (1931) published his theorem on incompleteness of Peano's theory. His result is of logic, not an arithmetic fact.
- Thoralf Skolem (in 1934) wrote a paper on the non-characterization of the series of numbers by means of a finite or countably infinite number of statements with exclusively individual variables [Sko34]
- Stephen C. Kleene has shown (in 1936) that any recurrence that defines a computable function can be replaced by the operation of effective minimum (nowadays one can say every recursive function in the integers, is programmable by means of **while** instruction).
- Summing up, it seems that P. Erdős overlooked the computability theory, his colleagues - professors Rozsa Peter and Laszlo Kalmar (specialists in the theory of recursive functions) were able to point it out to him.

Andrzej Mostowski had a hope that many arithmetic theorems independent of the Peano axioms should be found. Collatz theorem is an example. The theorem on termination of Euclid's algorithm is another example of a theorem which is valid and unprovable in Peano theory. The law of Archimedes is yet another example. Note, both theorems need to be stated as algorithmic formulas, there is no first-order formula that expresess the termination property of Euclid's4.2 algorithm or law of Archimedes.

Further research

Perhaps you noted that two parts of our proof are formal (subsection 6.6) or near formal (subsection 4.2) proofs. We hope that a complete, easy to verify proof will be done in future.

Acknowlegments

Andrzej Szałas has shown to us the lacunes in our proofs. Hans Langmaack and Wiktor Dańko sent some comments. Antek Ciaputa helped in calculations and drawing of Hotel Collatz.

6. Suplements

For the reader's convenience, in this section we have included some definitions, some useful theorems, and samples of proofs in algorithmic natural number theory.

6.1. A structure with counterexamples

where Collatz computations may be of infinite length

Here we present some facts that are less known to the IT community.
These facts may seem strange. The reader may doubt the importance of those facts. Yet, it is worth considering, non-standard data structures do exist, and this fact has ramifications. Strange as they seem, still it is worthwhile to be aware of their existence.
Now, we will expose the algebraic structure $\mathfrak{J}$, which is a model of the theory Ar, i.e. all axioms of theory Ar are true in the structure $\mathfrak{J}$. First we will describe this structure as mathematicians do, then we will write a class (i.e. a program module) implementing this structure.

Mathematical description of Jaśkowski's structure

$\mathfrak{J}$ is an algebraic structure

$$\mathfrak{J} = \langle M; \underline{\mathbf{0}}, \underline{\mathbf{1}}, \oplus; =\rangle \tag{NonStandard}$$

such that M is a set of complex numbers $k + \imath w$, i.e. of pairs $\langle k, w\rangle$, where element $k \in \mathbb{Z}$ is an integer, and element $w \in \mathbb{Q}^+$ is a rational, non-negative number $w \geq 0$ and the following requirements are satisfied:

(i) for each element $k + \imath w$ if $w = 0$ then $k \geq 0$,

(ii) $\underline{\mathbf{0}} \stackrel{df}{=} \langle 0 + \imath 0\rangle$,

(iii) $\underline{\mathbf{1}} \stackrel{df}{=} \langle 1. + \imath 0\rangle$,

(iv) the operation $\oplus$ of addition is determined as usual

$$(k + \imath w) \oplus (k' + \imath w') \stackrel{df}{=} (k + k') + \imath(w + w').$$

(v) the predicate = denotes as usual identity relation.

Lemma 6.1. The algebraic structure $\mathfrak{J}$ is a model of first-order arithmetic of addition of natural numbers $\mathcal{T}$.

The reader may check that every axiom of the $\mathcal{T}$ theory (see definition6.2, p.26), is a sentence true in the structure $\mathfrak{J}$, cf. next subsection 6.2.

The substructure $\mathfrak{N} \subset \mathfrak{J}$ composed of only those elements for which $w = 0$ is also a model of the theory $\mathcal{T}$.
It is easy to remark that elements of the form $\langle k, 0\rangle$ may be identified with natural numbers k, $k \in N$. Have a look at table 1

The elements of the structure $\mathfrak{N}$ are called *reachable*, for they enjoy the following algorithmic property

$$\forall_{n\in N}\, \{y := \mathbf{0}; \mathbf{while}\ y \neq n\ \mathbf{do}\ y := y + \mathbf{1}\ \mathbf{od}\}(y = n)$$

The structure $\mathfrak{J}$ is not a model of the $\mathcal{ATN}$, algorithmic theory of natural numbers, cf . subsection 6.4. Elements of the structure $\langle k, w\rangle$. such as $w \neq \mathbf{0}$ are *unreachable*. i.e. for each element $x_0 = \langle k, w\rangle$ such that $w \neq 0$ the following condition holds

$$\neg\{y := \mathbf{0}; \mathbf{while}\ y \neq x_0\ \mathbf{do}\ y := y + \mathbf{1}\ \mathbf{od}\}(y = x_0)$$

The subset $\mathfrak{N} \subset \mathfrak{J}$ composed of only those elements for which $w = 0$ is a model of the theory $\mathcal{ATN}$ c.f. subsection 6.4. The elements of the structure $\mathfrak{N}$ are called *reachable*. A very important theorem of the foundations of mathematics is

Lemma 6.2. The structures $\mathfrak{N}$ and $\mathfrak{J}$ are not isomorphic.

For the proof see [Grz71], p. 256. As we will see in a moment, this fact is also important for IT specialists.

An attempt to visualize structure $\mathfrak{M}$ is presented in the form of table 1. The universe of the structure $\mathfrak{J}$ decomposes onto two disjoint subsets (one green and one red). Every element of the form $\langle k, 0\rangle$ (in this case $k > 0$) represents the natural number k. Such elements are called *reachable* ones. Note,

Definition 6.1. An element n is a standard natural number (i.e. is *reachable*) iff the program of adding ones to initial zero terminates

$$n \in N \overset{df}{\Leftrightarrow} \{q := \mathbf{0};\ \mathbf{while}\ q \neq n\ \mathbf{do}\ q := q + \mathbf{1}\ \mathbf{od}\}(q = n)$$

or, equivalently

$$n \in N \overset{df}{\Leftrightarrow} \{q := \mathbf{0}\} \bigcup \{\mathbf{if}\ n \neq q\ \mathbf{then}\ q := q + \mathbf{1}\ \mathbf{fi}\}(q = n)$$

Table 1. Model $\mathfrak{J}$ of Presburger arithmetic consists of complex numbers $a + \imath\, b$ where $b \in Q^+$ and $a \in Z$, additional condition: $b = 0 \Rightarrow a \geq 0$. Definition of order $n > m \overset{df}{\equiv} \exists_{u \neq 0}\, m + u = n$. Invention of S. Jaśkowski (1929).

STANDARD (*reachable*) elements	*Unreachable* (INFINITE) elements							
					$\cdots$			
	$-\infty \cdots$	$-11 + \imath 2$	$-10 + \imath 2$	$\cdots$	$0 + \imath 2$	$1 + \imath 2$	$2 + \imath 2$	$\cdots \infty$
					$\cdots$			
	$-\infty \cdots$	$-11 + \imath \frac{53}{47}$	$-10 + \imath \frac{53}{47}$	$\cdots$	$0 + \imath \frac{53}{47}$	$1 + \imath \frac{53}{47}$	$2 + \imath \frac{53}{47}$	$\cdots \infty$
					$\cdots$			
	$-\infty \cdots$	$-11 + \imath \frac{28}{49}$	$-10 + \imath \frac{28}{49}$	$\cdots$	$0 + \imath \frac{28}{49}$	$1 + \imath \frac{28}{49}$	$2 + \imath \frac{28}{49}$	$\cdots \infty$
					$\cdots$			
	$-\infty \cdots$	$-11 + \imath \frac{3}{47}$	$-10 + \imath \frac{3}{47}$	$\cdots$	$0 + \imath \frac{3}{47}$	$1 + \imath \frac{3}{47}$	$2 + \imath \frac{3}{47}$	$\cdots \infty$
					$\cdots$			
$0 \quad 1 \quad 2 \quad \cdots \quad 101 \quad \cdots \quad \infty$								

Note that the subset that consists of all non-reachable elements is well separated from the subset of reachable elements. Namely, every reachable natural number is less that any unreachable one. Moreover, there is no least element in the set of unreachable elements. I.e. the principle of minimum does not hold in the structure $\mathfrak{M}$.
Moreover, for every element n its computation contains either only standard, reachable numbers or is composed of only unreachable elements. This remark will be of use in our proof.

Remark 6.1. For every element n the whole Collatz computation is either in green or in reed quadrant of the table 1.

Elements of the structure $\mathfrak{M}$ are ordered as usual

$$\forall_{x,y}\, x < y \overset{df}{=} \exists_{z \neq \mathbf{0}}\, x + z = y.$$

Therefore, each reachable element is smaller than every unreachable element.
The order defined in this way is the lexical order. (Given two elements p and q, the element lying higher is bigger, if both are of the same height then the element lying on the right is bigger.)
The order type is $\omega + (\omega^* + \omega) \cdot \eta$

Remark 6.2. The subset of unreachable elements (red ones on the table 1) does not obey the principle of minimum.

Definition in programming language

Perhaps you have already noticed that the $\mathfrak{M}$ is a computable structure. The following is a class that implements the structure $\mathfrak{M}$. The implementation uses the integer type, we do not introduce rational numbers explicitly.

```
unit StrukturaM: class;
  unit Elm: class(k,li,mia: integer);
  begin
    if mia=0 then raise Error fi;
    if li * mia <0 then raise Error fi;
    if li=0 and k<0 then raise Error fi;
  end Elm;
  add: function(x,y:Elm): Elm;
  begin
    result := new Elm(x.k+y.k, x.li*y.mia+x.mia*y.li, x.mia*y.mia )
  end add;
  unit one : function:Elm;  begin   result:= new Elm(1,0,2)  end one;
  unit zero : function:Elm;  begin   result:= new Elm(0,0,2)  end zero;
  unit eq: function(x,y:Elm): Boolean;
  begin
    result := (x.k=y.k) and (x.li*y.mia=x.mia*y.li )
  end eq;
end StrukturaM
```

The following lemma expresses the correctness of the implementation with respect to the axioms of Presburger arithmetic $\mathcal{AP}$ (c.f. subsection 6.2) treated as a specification of a class (i.e. a module of program).

Lemma 6.3. The structure $\mathfrak{E} = \langle E, add, zero, one, eq\rangle$ composed of the set $E = \{o \text{ object} : o\ inElm\}$ of objects of class Elm with the *add* operation is a model of the $\mathcal{AP}$ theory,

$$\mathfrak{E} \models \mathcal{AP}$$

Infinite Collatz algorithm computation

How to execute the Collatz algorithm in StructuraM? It's easy.

```
pref StrukturaM block
  var n: Elm;
  unit odd: function(x:Elm): Boolean; ... result:=(x.k mod 2)=1 ... end odd;
  unit div2: function(x:Elm): Elm; ...
  unit 3xp1: function(n: Elm): Elm; . . . result:=add(n,add(n,add(n,one))); . . . end 3xp1;
begin
  n:= new Elm(8,1,2);
      ┌──────────────────────────────┐
      │ while  not eq(n,one)  do     │
      │   if  odd(n) then            │
  Cl: │     n:=3xp1(n)  else n:= div2(n) │   (* a version of algorithm Cl that uses class Elm *)
      │   fi                         │
      │ od                           │
      └──────────────────────────────┘
end block;
```

Below we present the computation of Collatz algorithm for $n = \langle 8, \frac{1}{2}\rangle$.

$$\langle 8, \frac{1}{2}\rangle,\ \langle 4, \frac{1}{4}\rangle, \langle 2, \frac{1}{8}\rangle,\ \langle 1, \frac{1}{16}\rangle,\ \langle 4, \frac{3}{16}\rangle,\ \langle 2, \frac{3}{32}\rangle,\ \langle 1, \frac{3}{64}\rangle,\ \langle 4, \frac{9}{64}\rangle,\ \langle 2, \frac{9}{128}\rangle, \cdots$$

Note, the computation of algorithm Gr for the same argument, looks simpler

$$\langle 8, \frac{1}{2}\rangle,\ \langle 4, \frac{1}{4}\rangle, \langle 2, \frac{1}{8}\rangle,\ \langle 1, \frac{1}{16}\rangle,\ \langle 1, \frac{3}{64}\rangle,\ \langle 1, \frac{9}{256}\rangle, \cdots$$

None of the elements of the above sequence is a standard natural number. Each of them is unreachable. It is worth looking at an example of another calculation. Will something change when we assign n a different object? e.g. n: =

new Elm (19,2,10)?

$$\langle 19, \frac{10}{2}\rangle, \langle 58, \frac{30}{2}\rangle, \langle 29, \frac{30}{4}\rangle, \langle 88, \frac{90}{4}\rangle, \langle 44, \frac{90}{8}\rangle, \langle 22, \frac{90}{16}\rangle, \langle 11, \frac{90}{32}\rangle, \langle 34, \frac{270}{32}\rangle, \langle 17, \frac{270}{64}\rangle,$$
$$\langle 52, \frac{810}{64}\rangle, \langle 26, \frac{405}{64}\rangle, \langle 13, \frac{405}{128}\rangle, \langle 40, \frac{1215}{128}\rangle, \langle 20, \frac{1215}{256}\rangle, \langle 10, \frac{1215}{256}\rangle, \langle 5, \frac{1215}{512}\rangle, \langle 16, \frac{3645}{512}\rangle, \langle 8, \frac{3645}{1024}\rangle,$$
$$\langle 4, \frac{3645}{2048}\rangle, \langle 2, \frac{3645}{4096}\rangle, \langle 1, \frac{3645}{8192}\rangle, \langle 4, \frac{3*3645}{8192}\rangle, \langle 2, \frac{3645*3}{2*8192}\rangle, \langle 1, \frac{3*3645}{4*8192}\rangle, \langle 4, \frac{9*3645}{4*8192}\rangle, \cdots$$

And one more computation.

$$\langle 19,0\rangle, \langle 58,0\rangle, \langle 29,0\rangle, \langle 88,0\rangle, \langle 44,0\rangle, \langle 22,0\rangle, \langle 11,0\rangle, \langle 34,0\rangle, \langle 17,0\rangle, \langle 52,0\rangle, \langle 26,0\rangle,$$
$$\langle 13,0\rangle, \langle 40,0\rangle, \langle 20,0\rangle, \langle 10,0\rangle, \langle 5,0\rangle, \langle 16,0\rangle, \langle 8,0\rangle, \langle 4,0\rangle, \langle 2,0\rangle, \langle 1,0\rangle.$$

Corollary 6.1. The structure $\mathfrak{M}$, which we have described in two different ways, is the model of the $\mathcal{AP}$ theory with the non-obvious presence of unreachable elements in it.

Corollary 6.2. The halting property of the Collatz algorithm cannot be proved from the axioms of the $\mathcal{T}$ theory, nor from the axioms of $\mathcal{AP}$ theory.

6.2. Presburger's arithmetic

Presburger's arithmetic is another name of elementary theory of natural numbers with addition.
We shall consider the following theory , cf. [Pre29],[Grz71] p. 239 and following ones.

Definition 6.2. Theory $\mathcal{T} = \langle \mathcal{L}, \mathcal{C}, Ax\rangle$ is the system of three elements:

$\mathcal{L}$ is a language of first-order. The alphabet of this language consist of: the set V of variables, symbols of operations: $0, S, +$, symbol of equality relation $=$, symbols of logical functors and quantifiers, auxiliary symbols as brackets ...
The set of well formed expressions is the union of te set T of terms and the set of formulas F.
The set T is the least set of expressions that contains the set V and constants 0 and 1 and closed with respect to the rules: if two expressions τ_1 and τ_2 are terms, then the expression $(\tau_1 + \tau_2)$ is a term too.
The set F of formulas is the least set of expressions that contains the equalities (i.e. the expressions of the form $(\tau_1 = \tau_2)$) and closed with respect to the following formation rules: if expressions α and β are formulas, then the aexpression of the form

$$(\alpha \vee \beta),\ (\alpha \wedge \beta),\ (\alpha \implies \beta),\ \neg\alpha$$

are also formulas, moreover, the expressions of the form

$$\forall_x\, \alpha,\ \exists_x\, \alpha$$

where x is a variable and α is a formula, are formulas too.

$\mathcal{C}$ is the operation of consquence determined by axioms of first-order logic and the inference rules of the logic,

Ax is the set of formulas listed below.

$$\forall_x\, x + 1 \neq 0 \tag{a}$$
$$\forall_x \forall_y\, x + 1 = y + 1 \implies x = y \tag{b}$$
$$\forall_x\, x + 0 = x \tag{c}$$
$$\forall_{x,y}\, (y + 1) + x = (y + x) + 1 \tag{d}$$
$$\Phi(0) \wedge \forall_x [\Phi(x) \implies \Phi(x+1)] \implies \forall_x \Phi(x) \tag{I}$$

The expression $\Phi(x)$ may be replaced by any formula. The result is an axiom of theory This is the induction scheme. We augment the set of axioms adding four axioms that define a coiple of useful notions.

$$even(x) \stackrel{df}{\equiv} \exists_y\, x = y + y \tag{e}$$

$$odd(x) \stackrel{df}{\equiv} \exists_y\, x = y + y + 1 \tag{o}$$

$$x\, div\, 2 = y \equiv (x = y + y \vee x = y + y + 1) \tag{D2}$$

$$3x \stackrel{df}{=} x + x + x \tag{3x}$$

The theory $\mathcal{T}'$ obtained in this way is a conservative extension of theory $\mathcal{T}$.

Below we present another theory $\mathcal{AP}$ c.f. [Pre29], we shall use two facts: 1) theory $\mathcal{AP}$ is complete and hence is decidable, 2) both theories are elementarily equivalent.

Definition 6.3. Theory $\mathcal{AP} = \langle \mathcal{L}, \mathcal{C}, AxP \rangle$ is a system of three elements :

$\mathcal{L}$ is a language of first-order. The alphabet of this language contains the set V of variables, symbols of functors : $0, +$, symbol of equality predicate $=$.
The set of well formed-expressions is the union of set of terms T and set of formulas F. The set of terms T is the least set of expressions that contains the set of variables V and the expression 0 and closed with respect to the following two rules: 1) if two expressions τ_1 and τ_2 are terms, then the expression $(\tau_1 + \tau_2)$ is also a term, 2) if the expression τ is a term, then the expression $S(\tau)$ is also a term.

$\mathcal{C}$ is the consequence operation determined by the axioms of predicate calculus and inference rules of first-order logic

AxP The set of axioms of the $\mathcal{AP}$ theory is listed below.

$$\forall_x\, x + 1 \neq 0 \tag{A}$$

$$\forall_x\, x \neq 0 \implies \exists_y x = y + 1 \tag{B}$$

$$\forall_{x,y}\, x + y = y + x \tag{C}$$

$$\forall_{x,y,z}\, x + (y + z) = (x + y) + z \tag{D}$$

$$\forall_{x,y,z}\, x + z = y + z \implies x = y \tag{E}$$

$$\forall_x\, x + 0 = x \tag{F}$$

$$\forall_{x,z}\, \exists_y\, (x = y + z \vee z = y + x) \tag{G}$$

$$\forall_x\, \exists_y\, (x = y + y \vee x = y + y + 1) \tag{H2}$$

$$\forall_x\, \exists_y\, (x = y + y + y \vee x = y + y + y + 1 \vee x = y + y + y + 1 + 1) \tag{H3}$$

. . . .

$$\forall_x\, \exists_y \left(\begin{array}{l} x = \underbrace{y + y + \cdots + y}_{k} \vee \\ \quad x = \underbrace{y + y + \cdots + y}_{k} + 1 \vee \\ \quad\quad x = \underbrace{y + y + \cdots + y}_{k} + \underbrace{1 + 1}_{2} \vee \\ \quad\quad\quad \ldots \\ \quad\quad\quad x = \underbrace{y + y + \cdots + y}_{k} + \underbrace{1 + 1 + \cdots + 1}_{k-2} \vee \\ \quad\quad\quad\quad x = \underbrace{y + y + \cdots + y}_{k} + \underbrace{1 + 1 + \cdots + 1}_{k-1} \end{array} \right) \tag{Hk}$$

...

The axioms $H2$ -Hk ... may be given a shorter form. Let us introduce numerals, ie. the constants representing term of the form

$\underline{\mathbf{2}} \stackrel{df}{=} 1+1$

$\underline{\mathbf{3}} \stackrel{df}{=} 1+1+1$

...

$\underline{\mathbf{k}} \stackrel{df}{=} \underbrace{1+1+\ldots 1}_{k \text{ times}}$

...

Now, the axioms take form

$$\forall_x \; x \bmod \underline{\mathbf{2}} = \underline{\mathbf{0}} \vee x \bmod \underline{\mathbf{2}} = \underline{\mathbf{1}} \tag{H2'}$$
$$\forall_x \; x \bmod \underline{\mathbf{3}} = \underline{\mathbf{0}} \vee x \bmod \underline{\mathbf{3}} = \underline{\mathbf{1}} \vee x \bmod \underline{\mathbf{3}} = \underline{\mathbf{2}} \tag{H3'}$$
...
$$\forall_x \bigvee_{j=0}^{k-1} x \bmod \underline{\mathbf{k}} = \underline{\mathbf{j}} \tag{Hk'}$$

Let us recall a couple of useful theorems

F1. Theory $\mathcal{T}$ is elementarily equivalent to the theory $\mathcal{AP}$.[Pre29] [Sta84]

F2. Theory $\mathcal{AP}$ is decidable. [Pre29].

F3. The computational complexity of theory $\mathcal{AP}$, is double exponential $O(2^{2^n})$ this result belongs to Fisher and Rabin, see [MF74].

F4. Theories $\mathcal{T}$ and $\mathcal{AP}$ have non-standard model, see section 6.1, p. 22.

Now, we shall prove a couple of useful theorems of theory $\mathcal{T}$.

First, we shall show that the sentence $\forall_n \exists_{x,y,z} \, n \cdot 3^x + y = 2^z$ is a theorem of the theory $\mathcal{T}$ of addition. Operations of multiplication and power are inaccessible in the theory $\mathcal{T}$. However, we do not need them.
We enrich the theory $\mathcal{T}$ adding two functions $P2(\cdot)$ and $P3(\cdot,\cdot)$. defined in this way

Definition 6.4. Two functions are defined $P2$ (of oneargument) and $P3$ (of two-arguments).

$$\begin{array}{l|l} P2(0) \stackrel{df}{=} 1 & P3(y,0) \stackrel{df}{=} y \\ P2(x+1) \stackrel{df}{=} P2(x) + P2(x) & P3(y,x+1) \stackrel{df}{=} P3(y,x) + P3(y,x) + P3(y,x) \end{array}$$

Lemma 6.4. The definitions given above are correct, i.e. the following sentences aretheorems of the theory with two definitions

$$\mathcal{T} \vdash \forall_x \exists_y \, P2(x) = y \text{ and}$$

$$\mathcal{T} \vdash \forall_{x,y,z} P2(x) = y \wedge P2(x) = z \implies y = z.$$

Similarly, the sentences $\forall_{y,x}\exists_z\, P3(y,x) = z$ and $\forall_{y,x,z,u} P3(y,x) = z \wedge P3(y,x) = u \implies z = u$ are theorems of theory $\mathcal{T}$.

An easy proof goes by induction with respect to the value of variable x.

In the proof of the lemma 6.5 , below, we shall use the definition of the order relation

$$a < b \overset{df}{=} \exists_{c \neq 0}\, a + c = b.$$

Making use of the definition of function $P2$ and $P3$ we shall write the formula $P3(n,x)+y = P2(z)$ as it exppresses the same content as expression $n \cdot 3^x + y = 2^z$.

Lemma 6.5. The following sentence is a theorem of the theory $\mathcal{T}$ enriched by the definitions of $P2$ and $P3$ functions.

$$\forall_n \exists_{x,y,z} P3(n,x) + y = P2(z)$$

Proof:
We begin proving by induction that $\mathcal{T} \vdash \forall_n\, n < 2^n$. It is easy to see that $\mathcal{T} \vdash 0 < P2(0)$. We shall prove that $\mathcal{T} \vdash \forall_n (n < P2(n) \implies (n+1 < P2(n+1))$. Inequality $n+1 < P2(n+1)$ follows from the two following inequalities $\mathcal{T} \vdash n < P2(n)$ and $\mathcal{T} \vdash 1 < P2(n)$. Hence the formula $n+1 < P2(n) + P2(n))$ is a theorem of theory $\mathcal{T}$. By definition $P2(n) + P2(n) = P2(n+1)$.
In the similar manner, we can prove the formula $\mathcal{T} \vdash \forall_n \forall_x\, P3(n,x) < P2(n+x+x)$
As a consequence we have $\mathcal{T} \vdash \forall_n \exists_{x,y,z}\, P3(n,x) + y = P2(z)$. □

Lemma 6.6. Let $\mathfrak{M}$ be any model of Presburger arithmetic. If there exists a triple $\langle x, y, z\rangle$ of reachable elements such that it satisfies the equation $P3(n,x) + y = P2(z)$ i.e. $n \cdot 3^x + y = 2^z$ then the element n is reachable.

Proof:
If the following formulas are valid in the structure $\mathfrak{M}$
$\{q := 0;\ \mathbf{while}\ q \neq x\ \mathbf{do}\ q := q+1\ \mathbf{od}\}(x = q)$,
$\{q := 0;\ \mathbf{while}\ q \neq y\ \mathbf{do}\ q := q+1\ \mathbf{od}\}(y = q)$,
$\{q := 0;\ \mathbf{while}\ q \neq z\ \mathbf{do}\ q := q+1\ \mathbf{od}\}(z = q)$
and the following equation is valid too $P3(n,x) + y = P2(z)$ then it is easy to verify that the formula $\{t := 0;\ \mathbf{while}\ n \neq t\ \mathbf{do}\ t := t+1\ \mathbf{od}\}(t = n)$ is valid too.

Nr	Reason
1	a1=P2(z) is reachable
2	y+a2 =a1, a2 is reachable and a2=2^z-y
3	a3=P3(1,x) is reachable , a3=3^x
4	$\left\{\begin{array}{l} q := 1; a5 := a3; \\ \mathbf{while}\ a5 \neq a2\ \mathbf{do} \\ \quad q := q+1; \\ \quad a5 := a5 + a3 \\ \mathbf{od} \end{array}\right\}(q * a3 = a2)$ hence q=n

□

6.3. An introduction to the calculus of programs $\mathcal{AL}$

For the convenience of the reader we cite the axioms and inference rules of calculus of programs i.e. algorithmic logic $\mathcal{AL}$.
Note. *Every axiom of algorihmic logic is a tautology.*
Every inference rule of $\mathcal{AL}$ is sound. [MS87]

Axioms

axioms of propositional calculus

Ax_1 $((\alpha \Rightarrow \beta) \Rightarrow ((\beta \Rightarrow \delta) \Rightarrow (\alpha \Rightarrow \delta)))$

Ax_2 $(\alpha \Rightarrow (\alpha \vee \beta))$

Ax_3 $(\beta \Rightarrow (\alpha \vee \beta))$

Ax_4 $((\alpha \Rightarrow \delta) \Rightarrow ((\beta \Rightarrow \delta) \Rightarrow ((\alpha \vee \beta) \Rightarrow \delta)))$

Ax_5 $((\alpha \wedge \beta) \Rightarrow \alpha)$

Ax_6 $((\alpha \wedge \beta) \Rightarrow \beta)$

Ax_7 $((\delta \Rightarrow \alpha) \Rightarrow ((\delta \Rightarrow \beta) \Rightarrow (\delta \Rightarrow (\alpha \wedge \beta))))$

Ax_8 $((\alpha \Rightarrow (\beta \Rightarrow \delta)) \Leftrightarrow ((\alpha \wedge \beta) \Rightarrow \delta))$

Ax_9 $((\alpha \wedge \neg\alpha) \Rightarrow \beta)$

Ax_{10} $((\alpha \Rightarrow (\alpha \wedge \neg\alpha)) \Rightarrow \neg\alpha)$

Ax_{11} $(\alpha \vee \neg\alpha)$

axioms of predicate calculus

Ax_{12} $((\forall x)\alpha(x) \Rightarrow \alpha(x/\tau)))$
where term τ is of the same type as the variable x

Ax_{13} $(\forall x)\alpha(x) \Leftrightarrow \neg(\exists x)\neg\alpha(x)$

axioms of calculus of programs

Ax_{14} $K((\exists x)\alpha(x)) \Leftrightarrow (\exists y)(K\alpha(x/y))$ for $y \notin V(K)$

Ax_{15} $K(\alpha \vee \beta) \Leftrightarrow ((K\alpha) \vee (K\beta))$

Ax_{16} $K(\alpha \wedge \beta) \Leftrightarrow ((K\alpha) \wedge (K\beta))$

Ax_{17} $K(\neg\alpha) \Rightarrow \neg(K\alpha)$

Ax_{18} $((x := \tau)\gamma \Leftrightarrow (\gamma(x/\tau) \wedge (x := \tau)true)) \wedge ((q := \gamma\prime)\gamma \Leftrightarrow \gamma(q/\gamma\prime))$

Ax_{19} **begin** $K; M$ **end** $\alpha \Leftrightarrow K(M\alpha)$

Ax_{20} **if** γ **then** K **else** M **fi** $\alpha \Leftrightarrow ((\neg\gamma \wedge M\alpha) \vee (\gamma \wedge K\alpha))$

Ax_{21} **while** γ **do** K **od** $\alpha \Leftrightarrow ((\neg\gamma \wedge \alpha) \vee (\gamma \wedge K($**while** γ **do** K **od**$(\neg\gamma \wedge \alpha))))$

Ax_{22} $\bigcap K\alpha \Leftrightarrow (\alpha \wedge (K \bigcap K\alpha))$

Ax_{23} $\bigcup K\alpha \equiv (\alpha \vee (K \bigcup K\alpha))$

Inference rules

propositional calculus

R_1 $\dfrac{\alpha, (\alpha \Rightarrow \beta)}{\beta}$ (also known as modus ponens)

predicate calculus

R_6 $\dfrac{(\alpha(x) \Rightarrow \beta)}{((\exists x)\alpha(x) \Rightarrow \beta)}$

R_7 $\dfrac{(\beta \Rightarrow \alpha(x))}{(\beta \Rightarrow (\forall x)\alpha(x))}$

calculus of programs AL

R_2 $\dfrac{(\alpha \Rightarrow \beta)}{(K\alpha \Rightarrow K\beta)}$

R_3 $\dfrac{\{s(\textbf{if } \gamma \textbf{ then } K \textbf{ fi})^i(\neg\gamma \wedge \alpha) \Rightarrow \beta\}_{i \in N}}{(s(\textbf{while } \gamma \textbf{ do } K \textbf{ od } \alpha) \Rightarrow \beta)}$

R_4 $\dfrac{\{(K^i\alpha \Rightarrow \beta)\}_{i \in N}}{(\bigcup K\alpha \Rightarrow \beta)}$

R_5 $\dfrac{\{(\alpha \Rightarrow K^i\beta)\}_{i \in N}}{(\alpha \Rightarrow \bigcap K\beta)}$

In rules R_6 and R_7, it is assumed that x is a variable which is not free in β, i.e. $x \notin FV(\beta)$. The rules are known as the rule for introducing an existential quantifier into the antecedent of an implication and the rule for introducing a universal quantifier into the successor of an implication. The rules R_4 and R_5 are algorithmic counterparts of rules R_6 and R_7. They are of a different character, however, since their sets of premises are infinite. The rule R_3 for introducing a **while** into the antecedent of an implication of a similar nature. These three rules are called ω-rules. The rule R_1 is known as *modus ponens*, or the *cut*-rule. In all the above schemes of axioms and inference rules, α, β, δ are arbitrary formulas, γ and γ' are arbitrary open formulas, τ is an arbitrary term, s is a finite sequence of assignment instructions, and K and M are arbitrary programs.

Theorem 6.1. (***theorem on completeness of the calculus*** $\mathcal{AL}$)
Let $\mathcal{T} = \langle \mathcal{L}, \mathcal{C}, \mathcal{A}x \rangle$ be a consistent algorithmic theory, let $\alpha \in \mathcal{L}$ be a formula. The following conditions are equivalent

(*i*) Formula α is a theorem of the theory T, $\alpha \in \mathcal{C}(\mathcal{A}x)$,

(*ii*) Formula α is valid in every model of the theory T, $\mathcal{A}x \models \alpha$.

The proof may be found in [MS87].

6.4. An introduction to the algorithmic theory of numbers $\mathcal{ATN}$

The language of algorithmic theory of natural numbers $\mathcal{ATN}$ is very simple. Its alphabet contains one constant 0 *zero* , one one-argument functor s and predicate = of equality. We shall write $x+1$ instead of $s(x)$. Axioms of $\mathcal{ATN}$ were presented in the book [MS87]

$$\begin{array}{lll} A_1) & \forall x \{q := 0; \textbf{ while } q \neq x \textbf{ do } q := s(q) \textbf{ od}\}(q = x) & (R) \\ A_2) & \forall x\, s(x) \neq 0 & (N) \\ A_3) & \forall x \forall y\, s(x) = s(y) \implies x = y & (J) \end{array}$$

We can add another two-argument functor + and its definition

$$A_4) \quad \forall x\, \forall y \left\{ \begin{array}{l} q := 0; w := x; \\ \textbf{while } q \neq y \textbf{ do} \\ \quad q := s(q)\, ; w := s(w) \\ textbfod \end{array} \right\} (x + y = w) \qquad (D)$$

The termination property of the program in A_4 is a theorem of $\mathcal{ATN}$ theory as well as the formulas $x + 0 = x$ and $x + s(y) = s(x + y)$.

A sample (11 – 15) of Theorems of $\mathcal{ATN}$

$$\mathcal{ATN} \vdash \exists_x\, \alpha(x) \Leftrightarrow \{x := 0\} \bigcup \{x := x + 1\} \alpha(x) \tag{11}$$

$$\mathcal{ATN} \vdash \forall_x\, \alpha(x) \Leftrightarrow \{x := 0\} \bigcap \{x := x + 1\} \alpha(x) \tag{12}$$

Law of Archimedes

$$\mathcal{ATN} \vdash 0 < x < y \implies \{a := x;\ \textbf{while } a < y \textbf{ do } a := a + x \textbf{ od}\}(a \geq y) \tag{13}$$

Scheme of induction

$$\mathcal{ATN} \vdash \Big(\alpha(x/0) \wedge \forall_x \big(\alpha(x) \Rightarrow \alpha(x/s(x))\big)\Big) \implies \forall_x \alpha(x) \tag{14}$$

Correctness of Euclid's algorithm

$$\mathcal{ATN} \vdash \begin{pmatrix} n_0 > 0 \wedge \\ m_0 > 0 \end{pmatrix} \implies \left\{ \begin{array}{l} n := n_0;\ m := m_0; \\ \mathbf{while}\, n \neq m\, \mathbf{do} \\ \quad \mathbf{if}\ n > m\ \mathbf{then}\ n := n \dot{-} m \\ \qquad \mathbf{else}\ m := m \dot{-} n \\ \quad \mathbf{fi} \\ \mathbf{od} \end{array} \right\} (n = gcd(n_0, m_0)) \tag{15}$$

The theory $\mathcal{ATN}$ enjoys an important property of categoricity.

Theorem 6.2. (*meta*-theorem on categoricity of $\mathcal{ATN}$ theory)
Every model $\mathfrak{A}$ of the algorithmic theory of natural numbers is isomorphic to the structure $\mathfrak{N}$, c.f. subsection 6.1.

6.5. Proof of lemma 3.1

Let P and P' be two programs. Let α be any formula. The semantic property *programs P and P' are equivalent with respect to the postcondition α* is expressed by the formula of the form $(\{P\}\,\alpha \Leftrightarrow \{P'\}\,\alpha)$.
We shall use the following tautology of calculus of programs $\mathcal{AL}$.

$$\vdash \left(\overbrace{\left\{ \begin{array}{l} \mathbf{while}\ \gamma\ \mathbf{do} \\ \quad \mathbf{if}\ \delta\ \mathbf{then}\ K\ \mathbf{else}\ M\ \mathbf{fi} \\ \mathbf{od}; \end{array} \right\}}^{P:} \alpha \Leftrightarrow \overbrace{\left\{ \begin{array}{l} \mathbf{while}\ \gamma\ \mathbf{do} \\ \quad \mathbf{while}\ \gamma \wedge \delta\ \mathbf{do}\ K\ \mathbf{od}; \\ \quad \mathbf{while}\ \gamma \wedge \neg\delta\ \mathbf{do}\ M\ \mathbf{od} \\ \mathbf{od} \end{array} \right\}}^{P':} \alpha \right) \tag{16}$$

We apply the axioms Ax20 and Ax21

$$\vdash \left(\left\{ \begin{array}{l} \mathbf{while}\ \gamma\ \mathbf{do} \\ \quad \mathbf{if}\ \delta\ \mathbf{then}\ K\ \mathbf{else}\ M\ \mathbf{fi} \\ \mathbf{od}; \end{array} \right\} \alpha \Leftrightarrow \left\{ \begin{array}{l} \mathbf{if}\ \gamma\ \mathbf{then} \\ \quad \mathbf{while}\ \gamma \wedge \delta\ \mathbf{do}\ K\ \mathbf{od}; \\ \quad \mathbf{while}\ \gamma \wedge \neg\delta\ \mathbf{do}\ M\ \mathbf{od}\ ; \\ \quad \mathbf{while}\ \gamma\ \mathbf{do} \\ \qquad \mathbf{while}\ \gamma \wedge \delta\ \mathbf{do}\ K\ \mathbf{od}; \\ \qquad \mathbf{while}\ \gamma \wedge \neg\delta\ \mathbf{do}\ M\ \mathbf{od} \\ \quad \mathbf{od} \\ \mathbf{fi} \end{array} \right\} \alpha \right) \tag{17}$$

We can omit the instruction **if** (why?) . We swap internal instructions **while** inside the instruction **while**.

$$\vdash \left(\left\{ \begin{array}{l} \mathbf{while}\ \gamma\ \mathbf{do} \\ \quad \mathbf{if}\ \delta\ \mathbf{then}\ K\ \mathbf{else}\ M\ \mathbf{fi} \\ \mathbf{od}; \end{array} \right\} \alpha \Leftrightarrow \left\{ \begin{array}{l} \mathbf{while}\ \gamma \wedge \delta\ \mathbf{do}\ K\ \mathbf{od}\ ; \\ \mathbf{while}\ \gamma \wedge \neg\delta\ \mathbf{do}\ M\ \mathbf{od} \\ \mathbf{while}\ \gamma\ \mathbf{do} \\ \quad \mathbf{while}\ \gamma \wedge \neg\delta\ \mathbf{do}\ M\ \mathbf{od}; \\ \quad \mathbf{while}\ \gamma \wedge \delta\ \mathbf{do}\ K\ \mathbf{od} \\ \mathbf{od} \end{array} \right\} \alpha \right) \tag{18}$$

We can safely skip the second instruction while.

$$\vdash \left(\left\{ \begin{array}{l} \mathbf{while}\ \gamma\ \mathbf{do} \\ \quad \mathbf{if}\ \delta\ \mathbf{then}\ K\ \mathbf{else}\ M\ \mathbf{fi} \\ \mathbf{od}; \end{array} \right\} \alpha \Leftrightarrow \left\{ \begin{array}{l} \mathbf{while}\ \gamma \wedge \delta\ \mathbf{do}\ K\ \mathbf{od}\ ; \\ \mathbf{while}\ \gamma\ \mathbf{do} \\ \quad \mathbf{while}\ \gamma \wedge \neg\delta\ \mathbf{do}\ M\ \mathbf{od}; \\ \quad \mathbf{while}\ \gamma \wedge \delta\ \mathbf{do}\ K\ \mathbf{od} \\ \mathbf{od} \end{array} \right\} \alpha \right) \tag{19}$$

Now, we put $(\gamma \stackrel{df}{=} n \neq 1)$, $(\delta \stackrel{df}{=} even(n))$, $(K \stackrel{df}{=} \{n := n \div 2\})$ and $(M \stackrel{df}{=} \{n := 3n+1\})$. We make use of two theorems $(even(n) \implies n \neq 1)$ and $(M\delta)$ of theory $\mathcal{AP}$ c.f. subsection 6.2.

$$\mathcal{T} \vdash \left(\left\{ \begin{array}{l} \mathbf{while}\ n \neq 1\ \mathbf{do} \\ \quad \boxed{\begin{array}{l} \mathbf{if}\ even(n) \\ \mathbf{then}\ n := n \div 2 \\ \mathbf{else}\ n := 3n+1 \\ \mathbf{fi} \end{array}} \\ \mathbf{od}; \end{array} \right\} (n = 1) \Leftrightarrow \left\{ \begin{array}{l} \mathbf{while}\ even(n)\ \mathbf{do}\ n := n \div 2\ \mathbf{od}; \\ \mathbf{while}\ n \neq 1\ \mathbf{do} \\ \quad \boxed{\begin{array}{l} n := 3n+1; \\ \mathbf{while}\ \ even(n)\ \mathbf{do}\ n := n \div 2\ \mathbf{od} \end{array}} \\ \mathbf{od} \end{array} \right\} (n = 1) \right). \tag{20}$$

6.6. Proof of invariant of algorithm $Gr3$

We are going to prove the following implication (22) is a theorem of algorithmic theory of natural numbers $\mathcal{ATN}$.

$$\mathcal{ATN} \vdash (\varphi \implies \{\Delta_3\}\varphi) \tag{21}$$

$$\overbrace{\left(\begin{array}{l} n \cdot 3^i + Y_i = m_i \cdot 2^{Z_i} \wedge \\ Z_i = \sum_{j=0}^{i} k_j \wedge X_i = i \wedge \\ Y_i = \sum_{j=0}^{i-1} \left(3^{i-1-j} \cdot 2^{Z_j}\right) \wedge \\ m_0 = n \wedge\ k_0 = exp(m_0, 2) \wedge \\ \forall_{0<l\leq i} \big(m_l = m_{l-1}/2^{k_l} \wedge \\ k_l = exp(m_l, 2)\big) \end{array}\right)}^{\varphi} \implies \overbrace{\left\{\begin{array}{c} aux := 3 * m_i + 1; \\ k_{i+1} := exp(aux, 2); \\ m_{i+1} := aux/2^{k_{i+1}}; \\ Y_{i+1} := 3Y_i + 2^{Z_i}; \\ Z_{i+1} := Z_i + k_{i+1}; \\ X_i := i;; \\ \hline i := i + 1; \end{array}\right\}}^{\Delta_3} \overbrace{\left(\begin{array}{l} n \cdot 3^i + Y_i = m_i \cdot 2^{Z_i} \wedge \\ Z_i = \sum_{j=0}^{i} k_j \wedge \wedge X_i = i \\ Y_i = \sum_{j=0}^{i-1} \left(3^{i-1-j} \cdot 2^{Z_j}\right) \wedge \\ m_0 = n \wedge\ k_0 = exp(m_0, 2) \wedge \\ \forall_{0<l\leq i} \big(m_l = m_{l-1}/2^{k_l} \wedge \\ k_l = exp(m_l, 2)\big) \end{array}\right)}^{\varphi} \tag{22}$$

We apply the axiom of assignment instruction Ax_{18}. Note, we applied also axiom Ax_{19} of composed instruction. Namely, in the implication (22) we replace the its successor $\{\Delta_3\}\varphi$ by the formula $\{\Delta_3^{(1)}\}\varphi^{(1)}$.

$$\varphi \implies \overbrace{\left\{\begin{array}{c} aux := 3 * m_i + 1; \\ k_{i+1} := exp(aux, 2); \\ m_{i+1} := aux/2^{k_{i+1}}; \\ Y_{i+1} := 3Y_i + 2^{Z_i}; \\ Z_{i+1} := Z_i + k_{i+1}; \\ \hline X_i := i;; \end{array}\right\}}^{\Delta_3^{(1)}} \overbrace{\left(\begin{array}{l} n \cdot 3^{(i+1)} + Y_{(i+1)} = m_{(i+1)} \cdot 2^{Z_{(i+1)}} \wedge \\ Z_{(i+1)} = \sum_{j=0}^{(i+1)} k_j \wedge X_{i+1} = (i+1) \wedge \\ Y_{(i+1)} = \sum_{j=0}^{i} \left(3^{(i+1)-1-j} \cdot 2^{Z_j}\right) \wedge \\ m_0 = n \wedge\ k_0 = exp(m_0, 2) \wedge \\ \forall_{0<l\leq(i+1)} \big(m_l = m_{l-1}/2^{k_l} \wedge k_l = exp(m_l, 2)\big) \end{array}\right)}^{\varphi^{(1)}} \tag{23}$$

$$\varphi \implies \left\{\begin{array}{c} aux := 3 * m_i + 1; \\ k_{i+1} := exp(aux, 2); \\ m_{i+1} := aux/2^{k_{i+1}}; \\ Y_{i+1} := 3Y_i + 2^{Z_i}; \\ \hline Z_{i+1} := Z_i + k_{i+1}; \end{array}\right\} \left(\begin{array}{l} n \cdot 3^{(i+1)} + Y_{(i+1)} = m_{(i+1)} \cdot 2^{Z_{(i+1)}} \wedge \\ Z_{(i+1)} = \sum_{j=0}^{(i+1)} k_j \wedge X_{i+1} = (i+1) \wedge \\ Y_{(i+1)} = \sum_{j=0}^{i} \left(3^{(i+1)-1-j} \cdot 2^{Z_j}\right) \wedge \\ m_0 = n \wedge\ k_0 = exp(m_0, 2) \wedge \\ \forall_{0<l\leq(i+1)} \big(m_l = m_{l-1}/2^{k_l} \wedge k_l = exp(m_l, 2)\big) \wedge \end{array}\right) \tag{24}$$

$$\varphi \implies \left\{\begin{array}{c} aux := 3 * m_i + 1; \\ k_{i+1} := exp(aux, 2); \\ m_{i+1} := aux/2^{k_{i+1}}; \\ \hline Y_{i+1} := 3Y_i + 2^{Z_i}; \end{array}\right\} \left(\begin{array}{l} n \cdot 3^{(i+1)} + Y_{(i+1)} = m_{(i+1)} \cdot 2^{(Z_i + k_{i+1})} \wedge \\ (Z_i + k_{i+1}) = \sum_{j=0}^{(i+1)} k_j \wedge X_{i+1} = (i+1) \wedge \\ Y_{(i+1)} = \sum_{j=0}^{i} \left(3^{(i+1)-1-j} \cdot 2^{Z_j}\right) \wedge \\ m_0 = n \wedge\ k_0 = exp(m_0, 2) \wedge \\ \forall_{0<l\leq(i+1)} \big(m_l = m_{l-1}/2^{k_l} \wedge k_l = exp(m_l, 2)\big) \end{array}\right) \tag{25}$$

$$\varphi \implies \left\{\begin{array}{c} aux := 3 * m_i + 1; \\ k_{i+1} := exp(aux, 2); \\ \hline m_{i+1} := aux/2^{k_{i+1}}; \end{array}\right\} \left(\begin{array}{l} n \cdot 3^{(i+1)} + (3Y_i + 2^{Z_i}) = m_{(i+1)} \cdot 2^{(Z_i + k_{i+1})} \wedge \\ (Z_i + k_{i+1}) = \sum_{j=0}^{(i+1)} k_j \wedge X_{i+1} = (i+1) \wedge \\ (3Y_i + 2^{Z_i}) = \sum_{j=0}^{i} \left(3^{(i+1)-1-j} \cdot 2^{Z_j}\right) \wedge \\ m_0 = n \wedge\ k_0 = exp(m_0, 2) \wedge \\ \forall_{0<l\leq(i+1)} \big(m_l = m_{l-1}/2^{k_l} \wedge k_l = exp(m_l, 2)\big) \end{array}\right) \tag{26}$$

$$\varphi \implies \left\{\begin{array}{c} aux := 3 * m_i + 1; \\ \hline k_{i+1} := exp(aux, 2); \end{array}\right\} \left(\begin{array}{l} n \cdot 3^{(i+1)} + (3Y_i + 2^{Z_i}) = (aux/2^{k_{i+1}}) \cdot 2^{(Z_i + k_{i+1})} \wedge \\ (Z_i + k_{i+1}) = \sum_{j=0}^{(i+1)} k_j \wedge X_{i+1} = (i+1) \wedge \\ (3Y_i + 2^{Z_i}) = \sum_{j=0}^{i} \left(3^{(i+1)-1-j} \cdot 2^{Z_j}\right) \wedge \\ m_0 = n \wedge\ k_0 = exp(m_0, 2) \wedge \\ \forall_{0<l\leq(i+1)} \big(m_l = m_{l-1}/2^{k_l} \wedge k_l = exp(m_l, 2)\big) \end{array}\right) \tag{27}$$

$$\varphi \implies \left\{ \;aux := 3 * m_i + 1;\; \right\} \begin{pmatrix} n \cdot 3^{(i+1)} + (3Y_i + 2^{Z_i}) = (aux/2^{(exp(aux,2))}) \cdot 2^{(Z_i + (exp(aux,2)))} \wedge \\ (Z_i + (exp(aux,2))) = \sum_{j=0}^{(i+1)} k_j \wedge X_{i+1} = (i+1) \wedge \\ (3Y_i + 2^{Z_i}) = \sum_{j=0}^{i} \left(3^{(i+1)-1-j} \cdot 2^{Z_j} \right) \wedge \\ m_0 = n \wedge \; k_0 = exp(m_0, 2) \wedge \\ \underset{0<l\le(i+1)}{\forall} \left(m_l = m_{l-1}/2^{k_l} \wedge k_l = exp(m_l, 2) \right) \end{pmatrix} \tag{28}$$

$$\varphi \implies \{\} \overbrace{\begin{pmatrix} n \cdot 3^{(i+1)} + (3Y_i + 2^{Z_i}) = ((3*m_i+1)/2^{(exp((3*m_i+1),2))}) \cdot 2^{(Z_i + (exp((3*m_i+1),2)))} \wedge \\ (Z_i + (exp((3*m_i+1),2))) = \sum_{j=0}^{(i+1)} k_j \wedge X_{i+1} = (i+1) \wedge \\ (3Y_i + 2^{Z_i}) = \sum_{j=0}^{i} \left(3^{(i+1)-1-j} \cdot 2^{Z_j} \right) \wedge \\ m_0 = n \wedge \; k_0 = exp(m_0, 2) \wedge \\ \underset{0<l\le(i+1)}{\forall} \left(m_l = m_{l-1}/2^{k_l} \wedge k_l = exp(m_l, 2) \right) \end{pmatrix}}^{\psi} \tag{29}$$

The implications (22) –(29) are mutually equivalent.
One can easily verify that the last implication ($\varphi \implies \psi$) (29) is a theorem of Presburger arithmetic $\mathcal{T}$ and hence it is a theorem of $\mathcal{ATN}$ theory.
Therefore the first implication ($\varphi \implies \{\Delta_3\}\varphi$) (22) is a theorem of algorithmic theory of natural numbers $\mathcal{ATN}$.

q.e.d.

Remark 6.3. Note, the process of creation a proof like this can be automatized.
The verification of the above proof can be automatized too.

References

[Grz71] Andrzej Grzegorczyk. Zarys Arytmetyki Teoretycznej. PWN, Warszawa, 1971.

[Lag10] Jeffrey C. Lagarias, editor. The Ultimate Challenge: The 3x+1 Problem. American Mathematical Society, Providence R.I., 2010.

[MF74] M.O.Rabin M.J. Fisher. Super-exponential complexity of Presburger's arithmetic. In Proceedings of AMS Symposia on Complexity of Real Computational Processes, volume VIII, 1974.

[MS87] Grażyna Mirkowska and Andrzej Salwicki. Algorithmic Logic. `http://lem12.uksw.edu.pl/wiki/Algorithmic_Logic`, 1987. [Online; accessed 7-August-2017].

[MS21] G. Mirkowska and A. Salwicki. On Collatz theorem. `http://lem12.uksw.edu.pl/images/2/27/On-Collatz-thm-11-10-21.pdf`. [Online: accessed 17 December 2021], 2021.

[Pre29] Mojżesz Presburger. Über die Vollständigkeit eines gewissen Systems der Arithmetik ganzer Zahlen , in welchem die Addition als einzige Operation hervortritt . pages 92–101,395, 1929.

[Sal23] Andrzej Salwicki. A new proof of Euclids algorithm
. `https://arxiv.org/abs/2311.01474`. [Online: accessed 13 November 2023].

[Sko34] Thoralf Skolem. Über die nicht-charakterisierbarkeit der zahlenreihe mittels endlich oder abzählbar unendlich vieler aussagen mit ausschließlich zahlenvariablen. Fundamenta Mathematicae, 23:150–161, 1934.

[Sta84] Ryan Stansifer. Presburger's Article on Integer Arithmetic: Remarks and Translation. Cornell University, Technical Report TR84-639, 1984. `http://cs.fit.edu/~ryan/papers/presburger.pdf`. [Online; accessed 6-August-2021].

[Tar34] Alfred Tarski. Bemerkung der Redaktion. Fundamenta Mathematicae, 23:161, 1934.