

Na twierdzenie Collatza

Szkic dowodu

Grażyna Mirkowska

Dombrova Research

Partyzantów 19

05-092 Łomianki, POLAND

G.Mirkowska@uksw.edu.pl

Andrzej Salwicki

Dombrova Research

salwicki@mimuw.edu.pl

Streszczenie. Dowodzimy, że dla każdej liczby naturalnej n algorytm Collatza ma obliczenie skończone.

29 grudnia 2018

1. Wprowadzenie

Ten szkic jest raportem z aktualnego stanu pracy nad artykułem o twierdzeniu Collatza. Dowodzimy, *prawdziwości* następującej tezy: *dla każdej liczby naturalnej $n > 0$ obliczenie algorytmu Collatza jest skończone*. Idea dowodu jest bardzo prosta. Pełny dowód wymaga uzupełnienia szczegółów.

Odróżniamy pojęcie prawdziwości od pojęcia dowodu w teorii. W artykule znajdzie się argument wykazujący, że teoria pierwszego rzędu, zwana częst arytmetyką Peano, nie zawiera twierdzenia, które wyrażałoby własność stopu algorytmu Collatza. Skądinąd wiadomo, że algorytmiczna teoria liczb naturalnych zawiera twierdzenie będące formułą stopu algorytmu Collatza.

Na razie umiemy przedstawić dowodu.

Nie potrafimy też oszacować kosztu algorytmu Collatza. Jest to zastanawiające. Znane jest oszacowanie kosztu obliczania funkcji Ackermanna. A w naszym przypadku potrafiliśmy udowodnić, że algorytm zawsze zakończy obliczenie, ale nie potrafimy oszacować długości obliczenia.

Algorytmu Collatza przedstawiać nie trzeba, por. [Lag10].

$$C : \{ \text{while } n \neq 1 \text{ do if } \text{odd}(n) \text{ then } n := 3n + 1 \text{ else } n := n/2 \text{ fi od} \}$$

Mamy zamiar opisać jego obliczenia w dwu, na pozór, różnych strukturach danych: \mathfrak{N} oraz \mathfrak{T} .

Pierwsza struktura \mathfrak{N} to standardowy zbiór liczb naturalnych N ze zwykłymi działaniami następnika i

dodawania. To wystarczy by dobrze (tj. efektywnie) zdefiniować polecenia $n:=3 \cdot n + 1$ oraz $n:= n \div 2$. Dalej zapisywać je będziemy jako polecenie *mult3* i polecenie *div2*. W algorytmie występuje tylko jedna zmienna, nie musimy jej wymieniać z nazwy. W algorytmie Collatza występują też dwie formuły $n \neq 1$ oraz $odd(n)$. Zastąpimy je przez krótkie *equal1* i *odd*.

Druga struktura \mathfrak{T} to zbiór trójek liczb naturalnych z odpowiednio dobranymi poleceniami, zobacz poniżej. Stąd pomysł zapisania tego algorytmu w abstrakcyjny sposób.

$$C' : \{ \text{while not } equal1 \text{ do if } odd \text{ then } mult3 \text{ else } div2 \text{ fi od} \}$$

Jedyna zmienna n występująca w tym programie przyjmuje bądź wartości będące liczbami naturalnymi (gdy obliczenia realizujemy w strukturze \mathfrak{N} bądź trójkami liczb naturalnych (gdy obliczenia prowadzone są w strukturze \mathfrak{T}). Polecenia *mult3* i *div2* to, odpowiednio: pomnóż n przez 3 i dodaj 1 oraz podziel n przez 2, bądź, gdy działania wykonywane są na trójkach polecenia te realizowane są w inny sposób, zob. poniżej.

2. Trójki

Zaczniemy od następującego spostrzeżenia

Fakt 1. Dla każdej liczby naturalnej $n \neq 0$ istnieje nieskończenie wiele trójek liczb naturalnych x, y, z takich, że spełniona jest równość

$$n \cdot 3^x + y = 2^z$$

Mówimy, że trójka x, y, z reprezentuje (albo, koduje) liczbę n i zapisujemy to $\langle x, y, z \rangle \asymp n$.

Dowód:

Dowód tego intuicyjnego faktu wykorzystuje prawo Archimedes¹. Niech n będzie dowolnie ustaloną liczbą naturalną.

Wybierzmy liczbę x , może to być dowolnie duża liczba naturalna.

Wyznamy liczby naturalne y i z tak, by zachodziła równość $n \cdot 3^x + y = 2^z$. Niech liczba $k = n \cdot 3^x$. Niech liczba 2^z będzie najmniejszą potęgą liczby 2 większą od k . Kładziemy $y = 2^z - k$. \square

Przykład 2. Trójka $\langle 1, 7, 6 \rangle$ reprezentuje liczbę 19 ponieważ $19 \cdot 3 + 7 = 2^6$.

Liczba 19 jest także reprezentowana przez inne trójki

$\langle 1, 7, 6 \rangle$	$19 \cdot 3^1 + 7 = 2^6$
$\langle 2, 85, 8 \rangle$	$19 \cdot 3^2 + 85 = 2^8$
$\langle 3, 511, 10 \rangle$	$19 \cdot 3^3 + 511 = 2^{10}$
$\langle 4, 509, 11 \rangle$	$19 \cdot 3^4 + 509 = 2^{11}$
$\langle 5, 3575, 13 \rangle$	$19 \cdot 3^5 + 3575 = 2^{13}$
$\langle 6, 2533, 14 \rangle$	$19 \cdot 3^6 + 2533 = 2^{14}$
$\langle 7, 23983, 16 \rangle$	$19 \cdot 3^7 + 23983 = 2^{16}$
...	

¹Przypomnijmy, prawo Archimedes^a jest własnością algorytmiczną, prawo to nie jest wyrażalne formułą pierwszego rzędu.

Natomiast, nie każda trójka liczb naturalnych reprezentuje jakąś liczbę naturalną, np. trójki $\langle 2, 4, 11 \rangle$, $\langle 2, 4044, 11 \rangle$...

Definicja 3. Dwie trójki $\langle x, y, z \rangle$ i $\langle u, v, t \rangle$ są równoważne gdy reprezentują tę samą liczbę naturalną

$$\langle x, y, z \rangle \equiv \langle u, v, t \rangle \stackrel{\text{df}}{=} \frac{2^z - y}{3^x} \in N \wedge \frac{2^z - y}{3^x} = \frac{2^t - v}{3^u}$$

W zbiorze trójek możemy zdefiniować porządek leksykograficzny \prec .

Definicja 4.

$$\langle x, y, z \rangle \prec \langle u, v, t \rangle \stackrel{\text{df}}{\Leftrightarrow} z < t \text{ lub } z = t \text{ i } y < v \text{ lub } z = t \text{ i } y = v \text{ i } x < u$$

Z faktu 1 wynika, że dla każdej liczby naturalnej n można wskazać kodującą ją trójkę, która ma liczbę y większą od dowolnie wybranej liczby naturalnej k . Z uwagi tej skorzystamy w dalszym ciągu naszego dowodu.

Definicja 5. Nieparzystość trójek wyznaczona jest przez nieparzystość liczby y

$$\text{odd}(\langle x, y, z \rangle) \stackrel{\text{df}}{\Leftrightarrow} y \text{ jest nieparzyste}$$

Kolejne spostrzeżenie

Fakt 6. Liczba n jest nieparzysta wttw gdy reprezentująca ją trójka $\langle x, y, z \rangle$ jest nieparzysta.

Definicja 7. Jedynka

$$\text{equal1}(\langle x, y, z \rangle) \stackrel{\text{df}}{\Leftrightarrow} (2^z - y = 3^x)$$

Jedynka jest reprezentowana przez wiele trójek, np. $\langle 0, 0, 0 \rangle$, $\langle 0, 1, 1 \rangle$, $\langle 1, 1, 2 \rangle$, $\langle 1, 5, 3 \rangle$, ...

Na trójkach możemy określić dwie operacje

Definicja 8. Operacja div2 przeprowadza trójkę $\langle x, y, z \rangle$ w trójkę $\langle x, y \div 2, z - 1 \rangle$

$$\langle x, y, z \rangle \xrightarrow{\{\text{div2}\}} \langle x, y \div 2, z - 1 \rangle$$

Operacja mult3 określona jest na trójkach $\langle x, y, z \rangle$ takich, że liczba y jest parzysta lub y jest liczbą nieparzystą i $x > 0$ oraz $y > 3^{x-1}$. W takim przypadku operacja mult3 przeprowadza trójkę $\langle x, y, z \rangle$ w trójkę $\langle x - 1, y - 3^{x-1}, z \rangle$

$$\langle x, y, z \rangle \xrightarrow{\{\text{mult3}\}} \langle x - 1, y - 3^{x-1}, z \rangle$$

Zauważ

Fakt 9. Trójka $\langle x, y, z \rangle$ reprezentuje liczbę n wttw gdy trójka $\langle x, y \div 2, z - 1 \rangle$ reprezentuje liczbę $n \div 2$. Ponadto $\langle x, y \div 2, z - 1 \rangle \prec \langle x, y, z \rangle$.
Trójka $\langle x, y, z \rangle$ reprezentuje liczbę n wttw gdy trójka $\langle x - 1, y - 3^{x-1}, z \rangle$ reprezentuje liczbę $3n + 1$. Ponadto $\langle x - 1, y - 3^{x-1}, z \rangle \prec \langle x, y, z \rangle$.

Zauważmy z kolei

Lemat 10. Jeśli program $K: \{\text{if odd then mult3 else div2 fi}\}$ przeprowadza trójkę $\langle x, y, z \rangle$ w trójkę $\langle u, v, t \rangle$, to wynikowa trójka $\langle u, v, t \rangle$ jest mniejsza \prec od trójki $\langle x, y, z \rangle$.

$$\langle x, y, z \rangle \xrightarrow{\{\text{if odd then mult3 else div2 fi}\}} \langle u, v, t \rangle \text{ implikuje } \langle u, v, t \rangle \prec \langle x, y, z \rangle$$

Dowód:

Jeśli liczba y jest parzysta to od z odejmujemy jedynkę i dzielimy y przez 2. W przeciwnym przypadku zmniejszana jest liczba x i od y odejmowana jest liczba 3^x . \square

W każdym kroku obliczenia algorytmu Collatza wyrażenie $x+z$ zmniejsza swą wartość o 1, zmniejszana jest też wartość zmiennej y .

Wynika stąd, że każde obliczenie w strukturze \mathfrak{T} jest skończone.

Obserwacje poczynione dotychczas pozwalają stwierdzić, że następujący diagram jest przemienny.

Fakt 11. (Przemiennność diagramu)

Działanie na trójkach prowadzi od trójki kodującej liczbę n do kolejnej trójki, która reprezentuje wynik m instrukcji warunkowej $K : \{\text{if odd}(n) \text{ then } n:=3*n+1 \text{ else } n:=n \text{ div } 2 \text{ fi}\}$.

$$\begin{array}{ccc} n & \xrightarrow{\{\text{if odd}(n) \text{ then } m:=3n+1 \text{ else } m:=n/2 \text{ fi}\}_{\mathfrak{N}}} & m \\ n \cdot 3^x + y = 2^z \downarrow & & \downarrow m \cdot 3^u + v = 2^t \\ \langle x, y, z \rangle & \xrightarrow{\{\text{if odd}(y) \text{ then } u,v,t:=x-1,y-3^{x-1},z \text{ else } u,v,t:=x,y/2,z-1 \text{ fi}\}_{\mathfrak{T}}} & \langle u, v, t \rangle \end{array}$$

Przemiennność tego diagramu umożliwia nam dostrzeżenie, że każdemu obliczeniu algorytmu Collatza w strukturze \mathfrak{T} trójek, odpowiada obliczenie tego samego algorytmu w strukturze \mathfrak{N} liczb naturalnych.

Przyjmijmy oznaczenie K dla programu $\{\text{if odd then mult3 else div2 fi}\}$. Napis $K_{\mathfrak{N}}$ oznacza funkcję realizującą program K w strukturze \mathfrak{N} . Odpowiednio, napis $K_{\mathfrak{T}}$ oznacza funkcję realizującą program K w strukturze \mathfrak{T} .

Lemat 12. Niech n będzie dowolną liczbą naturalną.

Rozpatrzmy dwa obliczenia algorytmu Collatza realizowane w strukturze \mathfrak{T} trójek rozpoczynające się, odpowiednio, od trójek q i r . Oba obliczenia są skończone. Litera q , q_1, q_2, \dots, q_f oznaczają kolejne trójki pierwszego obliczenia. Ciąg r, r_1, r_2, \dots, r_g to drugie obliczenie. Pionowe znaki równości przypominają, że trójka q_i koduje liczbę n_i .

$$\left. \begin{array}{l}
 (* \text{ do liczby } n \text{ dobierz liczby } x, y, z \text{ tak by } n \cdot 3^x + y = 2^z *) \\
 z := \lceil \log n \rceil; zS := z; \\
 x := \text{najwi}ksza \text{ liczba } x \text{ taka, } \text{ze } 2^z > n * 3^x; xS := x; \\
 y := 2^z - n * 3^x; \\
 \text{nieznaleziono} := \text{true}; \text{Err} := \text{false}; \\
 \text{while } \text{nieznaleziono} \text{ do} \\
 \quad (* \text{ wykonaj algorytm IC dla trójki } \langle x, y, z \rangle *) \\
 \quad \left[\begin{array}{l}
 \text{while } 3^x + y \neq 2^z \text{ do} \\
 \quad \text{if } y \text{ nieparzyste and } (x = 0) \text{ then } \text{Err} := \text{true}; \text{exit fi}; \\
 \quad \text{if } y < 3^{x-1} \text{ and } y \text{ nieparzyste then } y := y + 2^z; z := z + 1; \text{fi}; \\
 \quad \text{if } y \text{ parzyste then } z := z - 1; y := y \text{ div } 2; \\
 \quad \text{else } x := x - 1; y := y - 3^x \text{ fi} \\
 \text{od;} \\
 \text{if } \text{Err} \\
 \text{then } (* \text{ dobierz nową trójkę } *) \\
 \quad z := zS + 1; zS := z; \\
 \quad x := \text{najwi}ksza \text{ liczba } x \text{ taka, } \text{ze } 2^z > n * 3^x; \\
 (* Z : \left[\begin{array}{l} \text{if } x = xS \text{ then } z := zS + 1; zS := z; \\ x := \text{najwi}ksza \text{ liczba } x \text{ taka, } \text{ze } 2^z > n * 3^x \text{ fi}; \end{array} \right] *) \\
 \quad xS := x; y := 2^z - n * 3^x; \text{Err} := \text{false} \\
 \text{else} \\
 \quad \text{nieznaleziono} := \text{false} \\
 \text{fi} \\
 \text{od}
 \end{array} \right] \\
 \end{array} \right\} CC :$$

Instrukcja warunkowa Z jest w komentarzu. Odkomentowanie tej instrukcji daje nowy, alternatywny algorytm, nazwijmy go AC , przyda się on później.

Jest oczywiste, że

Fakt 13. *Jeśli obliczenie algorytmu CC w strukturze trójek \mathfrak{T} jest skończone i wolne od błędu, to algorytm C wykonywany w standardowym modelu liczb naturalnych \mathfrak{N} zatrzymuje się po skończonej liczbie kroków.*

Udowodnimy parę faktów o algorytmie CC .

Fakt 14. *Każde obliczenie wychodzi z wewnętrznej pętli IC i albo jest sukces i osiągnięto jedynekę albo jest błąd i obliczenia nie można kontynuować.*

$$\mathfrak{N} \models \forall n (n \cdot 3^x + y = 2^z) \implies \{IC\} \underbrace{((3^x + y = 2^z) \vee (odd(y) \wedge (x = 0 \vee y < 3^{x-1})))}_{\langle x, y, z \rangle \simeq 1} \underbrace{\phantom{((3^x + y = 2^z) \vee (odd(y) \wedge (x = 0 \vee y < 3^{x-1}))}}_{Error}$$

Błąd o jakim tu mowa to niemożność kontynuowania obliczeń. Jest tak gdy y jest nieparzyste i $y < 3x - 1$ lub gdy y jest nieparzyste i $x = 0$. Pierwszy błąd łatwo wyeliminować, zauważ, że trójka $\langle x, y, z \rangle$ reprezentuje tę samą liczbę co trójka $\langle x, y + 2^z, z + 1 \rangle$. Wystarczy więc w sytuacji gdy y jest nieparzyste i $y < 3x - 1$ zastąpić trójkę $\langle x, y, z \rangle$ przez trójkę $\langle x, y + 2^z, z + 1 \rangle$. Wykażemy, że dla każdej liczby naturalnej $n > 0$ algorytm *CC* znajduje taką trójkę liczb naturalnych, która reprezentuje liczbę n , $\langle x_1, y_1, z_1 \rangle \asymp n$, i przy tym obliczenie wewnętrznego algorytmu *IC* kończy się trójką reprezentującą liczbę 1, $\langle x_f, y_f, z_f \rangle \asymp 1$.

Lemat 15. Rozważmy dwa kolejne obliczenia w wewnętrznej pętli *IC*. Załóżmy, że początkową trójką pierwszego obliczenia jest $\langle x, y, z \rangle$, a drugie obliczenie zaczyna się od trójki $\langle x+1, y', z+1 \rangle$. Przyjmijmy, że oba obliczenia są nieudane, w pierwszym obliczeniu wykonano f kroków, a w drugim obliczeniu wykonano g kroków. Pierwsze obliczenie kończy się trójką $\langle 0, 2k + 1, z_f \rangle$, a drugie obliczenie kończy się trójką $\langle 0, 2m + 1, z_g \rangle$.

TEZA. Jeśli spełnione są te założenia, to drugie obliczenie jest dłuższe o co najmniej dwa kroki i końcowa trójka drugiego obliczenia jest mniejsza od końcowej trójki pierwszego obliczenia

$$f + 2 \leq g \quad \text{oraz} \quad \langle 0, 2m + 1, z_g \rangle \prec \langle 0, 2k + 1, z_f \rangle.$$

Dowód:

Zacznijmy od dwu łatwych obserwacji:

- Jeśli trójka $\langle x, y, z \rangle$ reprezentuje (jakaś) liczbę naturalną, to $y < 2^z$.
- Z tego, że obie trójki $\langle x, y, z \rangle$ oraz $\langle x + 1, y', z + 1 \rangle$ reprezentują tę samą liczbę n , wynika, że $y' = 3y - 2^z$.
A więc $3y - 2^z < 2y$.

Porównajmy dwa obliczenia

lp.	Lewe	Prawe	$v \geq y$
1	$\langle x_1, y_1, z_1 \rangle = \langle x, y, z \rangle$	$\langle u_1, v_1, t_1 \rangle = \langle x + 1, 3y - 2^z, z + 1 \rangle$	$y - 2^z < 2y$
...	
i	$\langle x_i, y_i, z_i \rangle$	$\langle u_i, v_i, t_i \rangle$	$v_i < 2y_i$
...	
f	$\langle x_f, y_f, z_f \rangle = \langle 0, 2k + 1, z_f \rangle$	$\langle u_f, v_f, t_f \rangle = \langle 1, 2p + 1, z_f + 1 \rangle$	$2p + 1 < 2(2k + 1)$
$f + 1$		$\langle 0, 2p, z_f + 1 \rangle$	$2p < 2(2k + 1)$
$f + 2$		$\langle 0, p, z_f \rangle$	$p < 2k + 1$

Jeśli p jest liczbą parzystą, to prawe obliczenie będzie dłuższe, ale nie musi tak być. Zanotujmy, $f + 2 \leq g$.

Przypomnijmy, że $y < 3y - 2^z < 2y$ czyli $y_1 < v_1 < 2y_1$. Przez indukcję ze względu na długość lewego obliczenia dowodzimy, że dla $j = 1, \dots, f$ zachodzi $y_j < v_j < 2y_j$. (Bowiem, w każdym kroku prawego obliczenia "naśladujemy" decyzje podejmowane w tym samym kroku obliczenia lewego,

dzieląc przez 2 lub odejmując 3^x .)

Stąd otrzymujemy $2p + 1 < 2(2k + 1)$. Wobec tego $p < 2k + 1$. Po kolejnych dwu krokach prawego obliczenia mamy trójkę $\langle 0, p, z_f \rangle$. Jeśli jest to ostatnia trójka prawego obliczenia, tzn. gdy $p = 2m + 1 \wedge g = f + 2$, to zachodzi $\langle 0, 2m + 1, z_g \rangle < \langle 0, 2k + 1, z_f \rangle$.

Jeśli prawe obliczenie jest jeszcze dłuższe, czyli $g > f + 2$, to końcowa trójka prawego obliczenia jest jeszcze mniejsza od końcowej trójki lewego obliczenia, por. lemat 10. \square

Lemat 16. *Rozważmy teraz przypadek gdy dwa kolejne obliczenia w wewnętrznej pętli IC zaczynają się od trójek $\langle x, y, z \rangle$ i $\langle x, y'', z + 1 \rangle$. Przyjmujemy, że oba obliczenia są nieudane, w pierwszym obliczeniu wykonano f kroków, a w drugim obliczeniu wykonano g kroków.*

TEZA. *Jeśli spełnione są te założenia, to oba obliczenia są tej samej długości $f = g$.*

Dowód:

Oba obliczenia przebiegają podobnie. Oba zaczynają się od trójek reprezentujących tę samą liczbę n i w każdym kroku ta własność zostaje zachowana. Porównajmy te dwa obliczenia

lp.	Lewe	Prawe	$x \geq u$ i $z \geq t$
1	$\langle x_1, y_1, z_1 \rangle$	$\langle u_1, v_1, t_1 \rangle$	$u_1 = x_1$ i $t_1 = z_1 + 1$
...	
$i - 1$	$\langle x_{i-1}, y_{i-1}, z_{i-1} \rangle$	$\langle u_{i-1}, v_{i-1}, t_{i-1} \rangle$	$u_{i-1} = x_{i-1}$ i $t_{i-1} = z_{i-1} + 1$
i	$\langle x_i, y_i, z_i \rangle$	$\langle u_i, v_i, t_i \rangle$	$u_i = x_i$ i $t_i = z_i + 1$
...	
f	$\langle x_f, y_f, z_f \rangle$ $= \langle 0, 2k + 1, z_f \rangle$	$\langle u_f, v_f, t_f \rangle$ $= \langle 0, 2m + 1, z_f + 1 \rangle$	$u_i = x_i$ i $t_i = z_i + 1$

Gdy pierwsze obliczenie zakończy się na trójce, oznaczmy ją przez $\langle 0, 2k + 1, z_f \rangle$, to po tej samej liczbie kroków drugie obliczenie dochodzi do trójki $\langle 0, 2m + 1, z_f + 1 \rangle$, ponieważ oba obliczenia zaczynają z tą samą wartością x i w każdym kroku, parzystości kolejnych trójek w tych dwu obliczeniach są takie same, czyli w tych samych krokach zmniejszamy wartość x . Można tego dowieść przez indukcję ze względu na długość pierwszego obliczenia. Zobacz też diagram w lemacie 12. \square

Chcielibyśmy, by własność zmniejszających się trójek kończących nieudane obliczenia instrukcji IC była niezmiennikiem programu CC, ale tak nie jest, na co wskazuje poprzedzający to zdanie lemat 16. Nie przeszkodzi to nam w udowodnieniu następującego lematu.

Lemat 17. *Dla każdej liczby naturalnej $n > 0$ obliczenie algorytmu CC jest skończone.*

$$\mathfrak{T} \models \forall_n \{CC\} \neg \text{nieznaleziono}$$

Dowód:

Należy wykazać, że istnieje taka trójka $\langle x, y, z \rangle$, że obliczenie wewnętrznej pętli IC będzie udane tzn. po skończonej liczbie kroków otrzymamy trójkę spełniającą warunek $n \cdot 3^x + y = 2^z$. Na każde obliczenie $O = \{\langle x_i, y_i, z_i \rangle\}_{i \in I}$ algorytmu CC możemy też spojrzeć jako na ciąg $O = \{O_j\}_{j \in J}$ obliczeń algorytmu IC, przy rosnących trójkach początkowych. Zauważmy, że w obliczeniu algorytmu CC możemy pominąć te fragmenty, o jakich mówi lemat 16. Co prawda obliczenia algorytmu wewnętrznego IC

kończą się wtedy trójką większą od poprzedniej, ale stanowią wierną imitację poprzedniego obliczenia algorytmu IC.

Czytelnik zechce samodzielnie dokonać odpowiednich zmian w algorytmie CC.

Jeśli rozpatrzmy dowolne obliczenie tak zmodyfikowanego algorytmu AC, to mamy do czynienia z sekwencją obliczeń algorytmu IC o następującej własności: każde takie obliczenie O_j jest coraz dłuższe i wobec tego kończy się coraz mniejszą trójką końcową.

Jeśli rozpatrywać ciąg trójek końcowych obliczeń O_j to jest to ciąg malejący, ograniczony. A więc dla pewnej trójki reprezentującej liczbę $n > 0$ obliczenie algorytmu IC jest skończone i końcową trójką tego obliczenia jest $\langle 0, 0, 0 \rangle$ trójka reprezentująca liczbę 1.

□

2.1. Twierdzenie Collatza

Powyższe uwagi pozwalają nam sformułować następujące

Twierdzenie 18. (Collatza)

Dla dowolnej liczby naturalnej $n > 0$ obliczenie algorytmu Collatza jest skończone.

$$\mathfrak{N} \models \forall_n \{ \text{while } n \neq 1 \text{ do if } \text{odd}(n) \text{ then } n := 3n + 1 \text{ else } n := n/2 \text{ fi od} \} (n = 1)$$

Dowód:

Dowód wynika wprost z lematu 17, że dla każdej liczby naturalnej $n > 0$ obliczenie algorytmu CC (odpowiednio algorytmu AC) jest skończone. □

3. Wnioski

- Nietrudno zauważyć, że realizacja obliczeń w niestandardowym modelu arytmetyki dodawania może mieć obliczenie nieskończone.

Jako wniosek z twierdzenia Collatza uzyskujemy następujący

Fakt 19. Jeśli realizujemy algorytm Collatza w pewnym modelu teorii (Peano) pierwszego rzędu i jeśli w tej strukturze dla pewnego n jest obliczenie nieskończone, to struktura ta jest modelem niestandardowym.

- Algorytm Collatza wyznacza pewną *permutację* zbioru \mathbb{N} liczb naturalnych.
- Można opisać inny algorytm (odwrotny) szukający zadanej liczby n po kolei w warstwach, poczynając od warstwy S_0 .

Fakt 20. Algorytm odwrotny zawsze znajdzie liczbę n .

Algorytm ten przyporządkowuje liczbie n numer i warstwy oraz położenie j liczby n w warstwie. W ten sposób mamy nową *funkcję pary*. Numerem pary liczb $\langle i, j \rangle$ jest j -ta liczba w warstwie S_i .

4. Nowe pytania

1. Jak określić funkcję kosztu?
2. Czy algorytm Collatza wyznacza najmniejszą trójkę x, y, z taką, że obliczenie zaczynające się od niej, będzie wolne od błędów?
3. Rozumowanie przytoczone powyżej wykazuje *prawdziwość* twierdzenia Collatza. Twierdzenie to nie jest zawarte w zbiorze twierdzeń arytmetyki Peano. (...) Jak przeprowadzić *dowód* tego twierdzenia w algorytmicznej teorii liczb naturalnych? W dowodzie nie powinniśmy posługiwać się pojęciem obliczenia algorytmu. W formalizowanym dowodzie twierdzenia Collatza każda formuła powinna być albo aksjomatem rachunku programów albo aksjomatem algorytmicznej teorii liczb naturalnych albo powinna być wnioskiem z formuł poprzednich wyprowadzonym przez zastosowanie pewnej reguły wnioskowania rachunku programów (tj. logiki algorytmicznej), por. [Sal]

Literatura

- [Lag10] Jeffrey C. Lagarias, editor. *The Ultimate Challenge: The $3x+1$ Problem*. American Mathematical Society, Providence R.I., 2010.
- [Sal] Andrzej Salwicki. A new proof of Euclid's algorithm. "<http://lem12.uksw.edu.pl/wiki/0nEuclid'salgorithm>".

Tablica 1. Tabela obliczeń algorytmu Collatza dla n=1079

n	x	y	z
1079	15	1697398531	34
3238	14	1692615562	34
1619	14	846307781	33
4858	13	844713458	33
2429	13	422356729	32
7288	12	421825288	32
3644	12	210912644	31
1822	12	105456322	30
911	12	52728161	29
2734	11	52551014	29
1367	11	26275507	28
4102	10	26216458	28
2051	10	13108229	27
6154	9	13088546	27
3077	9	6544273	26
9232	8	6537712	26
4616	8	3268856	25
2308	8	1634428	24
1154	8	817214	23
577	8	408607	22
1732	7	406420	22
866	7	203210	21
433	7	101605	20
1300	6	100876	20
650	6	50438	19
325	6	25219	18
976	5	24976	18
488	5	12488	17
244	5	6244	16
122	5	3122	15
61	5	1561	14
184	4	1480	14
92	4	740	13
46	4	370	12
23	4	185	11
70	3	158	11
35	3	79	10
106	2	70	10
53	2	35	9
160	1	32	9
80	1	16	8
40	1	8	7
20	1	4	6
10	1	2	5
5	1	1	4
16	0	0	4
8	0	0	3
4	0	0	2
2	0	0	1
1	0	0	0

Tablica 2. Cztery obliczenia dla $n=19$

3	511	10	19	4	509	11	19	5	3575	13	19	6	2533	14	19
2	502	10	58	3	482	11	58	4	3494	13	58	5	2290	14	58
2	251	9	29	3	241	10	29	4	1747	12	29	5	1145	13	29
1	248	9	88	2	232	10	88	3	1720	12	88	4	1064	13	88
1	124	8	44	2	116	9	44	3	860	11	44	4	532	12	44
1	62	7	22	2	58	8	22	3	430	10	22	4	266	11	22
1	31	6	11	2	29	7	11	3	215	9	11	4	133	10	11
0	30	6	34	1	26	7	34	2	206	9	34	3	106	10	34
0	15	5	17	1	13	6	17	2	103	8	17	3	53	9	17
-1	Err			0	12	6	52	1	100	8	52	2	44	9	52
				0	6	5	26	1	50	7	26	2	22	8	26
				0	3	4	13	1	25	6	13	2	11	7	13
				-1	Err			0	24	6	40	1	8	7	40
								0	12	5	20	1	4	6	20
								0	6	4	10	1	2	5	10
								0	3	3	5	1	1	4	5
								-1	Err			0	0	4	16
												0	0	3	8
												0	0	2	4
												0	0	1	2
												0	0	0	1

Rozpoczęcie obliczeń od zbyt “małej” początkowej trójki reprezentującej liczbę n może zakończyć się błędem. Zauważ, dla większych trójek początkowych obliczenia są dłuższe, a końcowe trójki tworzą ciąg malejący.

$$\langle 0, 15, 5 \rangle \succ \langle 0, 3, 4 \rangle \succ \langle 0, 3, 3 \rangle \succ \langle 0, 0, 0 \rangle$$