Theoretical Computer Science 407 (2008) 596–600

Contents lists available at ScienceDirect

Theoretical Computer Science

journal homepage: www.elsevier.com/locate/tcs

Note

Two undecidable variants of Collatz's problems

Eero Lehtonen *

Department of Mathematics, University of Turku, FI-20014 Turku, Finland

ARTICLE INFO

Article history:
Received 21 September 2007
Received in revised form 5 August 2008
Accepted 25 August 2008
Communicated by F. Cucker

Keywords:
Collatz's conjecture
Collatz's original problem
Undecidability

ABSTRACT

The paper introduces a simple way to show that certain iterative, number theoretic problems are undecidable. As applications, variants of the Collatz's conjecture and the so-called Collatz's original problem are shown to be undecidable.

1. Introduction

In 1930s a German mathematician, Lothar Collatz, posed his famous conjecture known as the $3n + 1$ *conjecture* [2]. It states that if the function

$$C(n) = \begin{cases} 3n+1, & \text{if } n \text{ is odd} \\ n/2, & \text{if } n \text{ is even} \end{cases}$$

is iterated on any positive integer n, the iteration will lead at some point to 1. As of today, the Collatz conjecture remains an open problem. Although there has been some progress [2], it seems that Paul Erdös was quite right when he stated that "Mathematics is not yet ready for such problems".

In this paper, the undecidablility of a variant of the Collatz conjecture is shown. More precisely, it will be shown that for the function

$$f(n) = \begin{cases} 3n+t & \text{if } n \in A_t, \text{ where } t = -9, -8, \ldots, 8, 9 \\ n/2 & \text{if } n \text{ is even,} \end{cases}$$

where the sets A_t are recursive and form a partition of odd natural numbers, it is undecidable which iterations lead at some point to 1. This will be shown by using a certain coding of a computation of a Turing machine. The coding itself is a versatile tool and as another application, the undecidability of a variant of the so-called *Collatz's original problem* is shown as well. This problem is about finite cycles in a permutation of natural numbers.

The main purpose of this paper is to present a simple method for obtaining undecidability results for iterative number theoretic problems. This paper might also motivate further study of undecidable variants of the Collatz's problems.

* Tel.: +358 505 487 860.
E-mail address: elleht@utu.fi.

0304-3975/$ – see front matter
doi:10.1016/j.tcs.2008.08.029

2. An undecidable variant of Collatz's conjecture

In this section we introduce a coding of a computation of a Turing machine and with it show an undecidability result related to Collatz's conjecture. First, however, we define the conjecture and motivate our upcoming considerations.

Problem 1. Let $f : \mathbb{N} \to \mathbb{N}$ and $n \in \mathbb{N}$ be given. Decide, whether for some $k \in \mathbb{N}$ $f^{(k)}(n) = 1$, where

$$f^{(k)}(n) = \underbrace{f \circ f \circ \cdots \circ f}_{k \text{ times}}(n).$$

If such a k exists, we say that it solves this problem.

Using the previous Problem, one may formulate the well-known conjecture by Lothar Collatz [2].

Conjecture 2 (*Collatz*). *Let the function $C : \mathbb{N} \to \mathbb{N}$ be defined as*

$$C(n) = \begin{cases} 3n+1 & \text{if } n \text{ is odd,} \\ n/2 & \text{if } n \text{ is even.} \end{cases}$$

For every n there is a $k \in \mathbb{N}$ which solves Problem 1.

As a weaker statement we give

Conjecture 3. *Let C be defined as in Conjecture 2. Then there is an algorithm, which decides for every n, whether there exists a $k \in \mathbb{N}$ that solves Problem 1. In other words, Problem 1 is decidable for the function C.*

At present, Conjecture 2 remains unsolved [2] and up to our knowledge, so does Conjecture 3. A variant of Problem 1 involving so-called piecewise linear functions was proved undecidable by Conway [1].

For the undecidability result of this section, let us first define a coding $\mathcal{C}$ from the set of legal descriptions of computations of Turing machines to the set of natural numbers written in ternary representation.

We assume familiarity with Turing Machines and their configurations and computations as in [3]. We will assume that the input and tape alphabets coincide, hence $\Sigma = \Gamma = \{a_1, a_2, \ldots a_k\}$. We will be denoting the set of states by $Q = \{q_1, q_2, \ldots, q_m\}$ and assume that q_1 and q_2 are the initial and final state, correspondingly. We also define a list $\Delta = [u_1 \vdash v_1, \ldots, u_n \vdash v_n]$ of the transition rules.

Now such a Turing machine can be given as a triplet

$$\mathcal{M} = [[a_1, \ldots, a_k], [q_1, q_2, \ldots, q_m], [u_1 \vdash v_1, \ldots, u_n \vdash v_n]]. \tag{1}$$

Definition 4. Recall the presentation (1). First, define a coding $\mathcal{C}'$ to the binary alphabet as follows:

$$\mathcal{C}'(a_i) = 20^{5+i} \quad \text{for } i = 1, \ldots, k \quad \text{and} \quad \mathcal{C}'(q_i) = 20^{5+k+i} \text{ for } i = 1, \ldots, m,$$

where the codewords are regarded as words in $\{0, 2\}^*$. We also need codings for the special letters:

$$\mathcal{C}'([\,]) = 20 \qquad \mathcal{C}'(]\,) = 200 \qquad \mathcal{C}'(\,,\,) = 20^3$$
$$\mathcal{C}'(\vdash) = 20^4 \qquad \mathcal{C}'(\#) = 20^5.$$

For words $w = w_1 w_2 \cdots w_l$ whose length is greater than one, define

$$\mathcal{C}'(w) = \mathcal{C}'(w_1)\mathcal{C}'(w_2) \cdots \mathcal{C}'(w_l).$$

Note, that this applies to the elements of the list Δ. In the same way we now know how to encode the description (1) of a Turing machine.

Finally, for any word w whose encoding has been defined, let

$$\mathcal{C}(w) = 1\,\mathcal{C}'(w). \tag{2}$$

Notice, that $\mathcal{C}(w)$ is always an odd natural number when understood as written in ternary representation. Indeed,

$$\mathcal{C}(w) = 1 \cdot 3^{l(\mathcal{C}(w))-1} + s,$$

where $l(C(w))$ is the length of the ternary representation of $\mathcal{C}(w)$ and s is an even number.

In addition to the coding $\mathcal{C}$, we need also the following technical lemma. For it, consider ternary representations of odd natural numbers of the form

$$n = \mathcal{C}(\mathcal{M}\#w_0\#w_1\#\cdots\#w_l\#\alpha), \tag{3}$$

where $\mathcal{M}$ is a Turing machine given in the form (1), w_0 is a legal initial ID of $\mathcal{M}$, $w_0 \vdash w_1 \vdash \cdots \vdash w_l$, α does not contain the special letter #, and n (as a word) is a prefix of $\mathcal{C}(\mathcal{M}\#w_0\#w_1\#\cdots\#w_l\#w_{l+1}\#)$, where $w_l \vdash w_{l+1}$. Here we allow α to be the empty word.

Thus in (3), the number n is an (incomplete) encoding of a computation of a Turing machine $\mathcal{M}$.

Lemma 5. *Let m be a natural number. Then for some $t \in \{-9, -8, \ldots, 8, 9\}$, $3m + t$ is*

(1) *of the form $4 + 8r$, i.e., divisible by four but not by eight, and*
(2) *$(3m + t)/4$ is* not *of the form (3).*

Proof. It is clear, that for any m, there are at least two numbers $t_1, t_2 \in \{-9, -8, \ldots, 8, 9\}$ such that $3m + t_i$ fulfills the requirement 1. Let t_1 and t_2 be the smallest such numbers, hence $t_2 = t_1 + 8$. Now, if $(3m + t_1)/4$ is not of the form (3), also the requirement 2 is fulfilled.

Let us then assume that $(3m + t_1)/4$ actually is of the form (3). Now

$$(3m + t_2)/4 = (3m + t_1 + 8)/4 = (3m + t_1)/4 + 2.$$

If the ternary representation of $(3m+t_1)/4$ ends with digit 0, we deduce that the ternary representation of $(3m+t_2)/4$ ends with digit 2. The ternary representations of $(3m + t_1)/4$ and $(3m + t_2)/4$ agree in all the other digits. Since the computation of a Turing machine is *deterministic*, this implies that $(3m + t_2)/4$ is not of the form (3). If, on the other hand, the ternary representation of $(3m + t_1)/4$ ends with digit 2 it follows that the ternary representation of $(3m + t_2)/4$ ends with digit 1. Now by the definition of the coding $\mathcal{C}$ it is clear that $(3m + t_2)/4$ cannot be of the form (3). □

Now we are ready to state the main result of this section. In the following, by a recursive set we mean a set $A \subset \mathbb{N}$ for which it is effectively decidable whether $n \in A$ for any natural number n. Notice, that the recursive sets A_t given in the proof of the following Theorem form a partition of the set of odd natural numbers.

Theorem 6. *There exists a choice of recursive sets A_t such that the Problem 1 is undecidable for the function f defined as*

$$f(n) = \begin{cases} 3n + t & \text{if } n \in A_t, \text{ where } t = -9, -8 \ldots, 8, 9, \\ n/2 & \text{if } n \text{ is even.} \end{cases} \tag{4}$$

Proof. Let first $n \in \mathbb{N}$ be an odd number whose ternary representation is of the form (3),

$$n = \mathcal{C}(\mathcal{M}\#w_0\#w_1\#\cdots\#w_l\#\alpha).$$

In particular there exists a computation step $w_l \vdash w_{l+1}$ and thus $w_0 \vdash^* w_l$ is not a halting computation of $\mathcal{M}$.

Define now $f(n)$ to be the unique number whose length in ternary representation is one digit longer than that of the number n and for which

$$f(n) \text{ is a prefix of } \mathcal{C}(\mathcal{M}\#w_0\#w_1\#\cdots\#w_l\#w_{l+1}\#).$$

Thus $f(n)$ continues the encoding of the computation by one digit. Due to the encoding, $f(n) = 3n + 0$ or $f(n) = 3n + 2$.

Consider now any other odd number $n > 9$, for which $f(n)$ is yet not defined and let $f(n) = 3n+t$, where $t \in \{-9, \ldots, 9\}$, be the smallest number such that

(1) $f(n)$ is divisible by four but not by eight, and
(2) $f(n)/4$ is *not* of the form (3).

The fact that this can be done was proved in Lemma 5. The reason behind this definition is that after the Turing machine's computation halts, we want to descend down to 1 in our iteration. This descending should be controlled in such a way that the iteration does not collide with any other Turing machine's computation.

Finally for even n we define $f(n) = n/2$ and for odd $n \leq 9$ we define $f(n) = 3n + t$, $t \in \{-9, \ldots, 9\}$ to be the smallest possible power of 2, i.e. $f(9) = 32, f(7) = 16, f(5) = 8$ and $f(3) = f(1) = 1$.

Now consider n of the form

$$n = \mathcal{C}(\mathcal{M}\#w_0\#), \tag{5}$$

where w_0 is an initial ID of the Turing machine $\mathcal{M}$. If $\mathcal{M}$ does not halt on its input, clearly

$$\lim_{k\to\infty} f^{(k)}(n) = \infty.$$

Assume then, that the computation halts at some point. Now for some $h \in \mathbb{N}$,

$$f^{(h)}(n) = \mathcal{C}(\mathcal{M}\#w_0\#w_1\#\cdots\#w_{l-1}\#w_l\#),$$

where $w_0 \vdash^* w_l$ is a halting computation of $\mathcal{M}$. By the definition of the function f we know that $f^{(h+1)}(n)$ is divisible by four but not by eight. Moreover,

$$f^{(h+2)}(n) = f^{(h+1)}(n)/2 \quad \text{and}$$
$$f^{(h+3)}(n) = f^{(h+2)}(n)/2 = f^{(h+1)}(n)/4.$$

Notice that $(3n + t)/4 < n$ for every $t = -9, -8, \ldots, 8, 9$ whenever $n > 9$. This yields the inequality

$$f^{(h+3)}(n) = f^{(h+1)}(n)/4 < f^{(h)}(n).$$

By the same reasoning we have $f^{(h+6)}(n) < f^{(h+3)}(n)$ and more generally,

$$f^{(h+3(i+1))}(n) < f^{(h+3i)}(n), \quad \text{if } f^{(h+3i)}(n) > 9.$$

Finally, by the definition of f we find a natural number k, for which

$$f^{(k)}(n) = 1.$$

Thus we have proved, that for any natural number n of the form (5),

there exists a k which solves the Problem 1 $\iff \mathcal{M}$ halts on its initial ID w_0.

Since f can be written in the form (4) and the Halting problem is undecidable for Turing machines, the claim follows. □

3. On a bijection $\mathbb{N} \to \mathbb{N}$

Conjecture 2 was not the first iterative problem studied by Lothar Collatz. One of these problems that he investigated in 1932 [2] was about the permutation $\mathbb{N} \to \mathbb{N}$ defined as

$$f(3n) = 2n, \qquad f(3n-1) = 4n-1, \qquad f(3n-2) = 4n-3. \tag{6}$$

Collatz was interested about its cycle structure and asked in particular, whether or not the cycle containing $n = 8$ was finite. Thus it is justifiable to call this permutation problem *the original Collatz's problem*.

Next we will show that the technique introduced in the previous section gives an undecidability result concerning a generalization of the original Collatz's problem.

Problem 7. Let $f : \mathbb{N} \to \mathbb{N}$ be a recursive bijection and $n \in \mathbb{N}$. Decide, whether there exists a $k \in \mathbb{N}$, $k > 1$ such that

$$f^{(k)}(n) = n.$$

Theorem 8. *There exists a recursive bijection f for which Problem 7 is undecidable.*

Proof. Let $n \in \mathbb{N}$ be an odd number of the form (3),

$$n = \mathcal{C}(\mathcal{M}\#w_0\#w_1\#\cdots\#w_l\#\alpha).$$

Define $f(n)$ now exactly as in the previous section, i.e., $f(n)$ continues the encoding of the computation by one ternary digit. Otherwise, if

$$n = \mathcal{C}(\mathcal{M}\#w_0\#w_1\#\cdots\#w_l\#),$$

where $w_0 \vdash^* w_l$ is a halting computation of $\mathcal{M}$, define

$$f(n) = \mathcal{C}(\mathcal{M}\#w_0\#)\,1\,1\ldots\,1$$

to be the natural number for which $l(f(n)) = l(n)$, where $l(w)$ is the length of w. Here n and $f(n)$ are treated as words over the ternary alphabet.

Let now n be any number of the form

$$n = \mathcal{C}(\mathcal{M}\#w_0\#)\,1\ldots\,1 \tag{7}$$

for which there is no halting computation $w_0 \vdash w_1 \vdash^* w_l$ of $\mathcal{M}$ such that

$$l(\mathcal{C}(\mathcal{M}\#w_0\#w_1\#\cdots\#w_l\#)) < l(n).$$

Then define $f(n)$ to be the largest proper prefix of n (when n is seen as a word over $\{0, 1, 2\}^*$). Finally for those $n \in \mathbb{N}$, for which $f(n)$ has not been yet defined, set $f(n) = n$.

It is easy, although a bit tedious, to show that the hereby defined f is surjective. Consider for example the case where $m \in \mathbb{N}$ is of the form (3) and $l > 0$ or α is not the empty word. Now if n is the largest proper prefix of m, $f(n) = m$. If, on the other hand,

$$m = \mathcal{C}(\mathcal{M}\#w_0\#),$$

where w_0 is a legal initial ID of $\mathcal{M}$ then $f(m1) = m$. The remaining cases can be dealt with in similar fashion.

Let us show that the function f is injective as well. For this, assume that for some natural numbers n_1 and n_2, $n_1 < n_2$, we have $f(n_1) = f(n_2)$. Now if $l(f(n_1)) > l(n_1)$, n_1 must be of the form (3). Thus n_2 cannot be of the same form, since the computation of a Turing machine is deterministic. Now, either $f(n_2) = n_2$ or the ternary representation of $f(n_2)$ contains at least two times the digit 1. Clearly both cases are impossible by the definition of the function f and the coding $\mathcal{C}$.

Consider next the case $l(f(n_1)) < l(n_1)$. Now n_1 must be of the form (7). Moreover, n_2 cannot be of the form (3) and since we assumed that $n_2 > n_1$, this case is also impossible.

Finally consider the case $l(f(n_1)) = l(n_1)$. This is possible if either $f(n_1) = n_1$ or n_1 is a coding of a halting computation of a Turing machine. If $f(n_1) = n_1$ we know by definition of the function f that $n_2 = n_1$, a contradiction. The same holds for the second case.

Hence, f is a recursive bijection. Consider now a natural number n of the form

$$n = \mathcal{C}(\mathcal{M}\#w_0\#),$$

where w_0 is an initial ID of the Turing machine $\mathcal{M}$. If the computation of the machine $\mathcal{M}$ does not halt on the initial ID, it is clear that

$$\lim_{k\to\infty} f^{(k)}(n) = \infty$$

and moreover, there cannot exist any natural number $k > 0$ for which $f^{(k)}(n) = n$. If, on the other hand, the computation halts we know that there exists a $h \in \mathbb{N}$ for which

$$f^{(h)}(n) = \mathcal{C}(\mathcal{M}\#w_0\#)\,1\,1\ldots\,1.$$

Now, due to the definition of f, $f^{(h+1)}(n)$ is the largest non-trivial prefix of $f^{(h)}(n)$. Also $f^{(h+2)}(n)$ is the largest non-trivial prefix of $f^{(h+1)}(n)$ and hence, by iterating f we come to a natural number $k > 0$ for which $f^{(k)}(n) = n$. Thus we have proved that for such a n,

$$\text{There exists a } k > 1 \text{ which solves the Problem 7} \iff \mathcal{M} \text{ halts on its initial ID } w_0.$$

Again, since the Halting problem for Turing machines is undecidable, the claim follows. □

Acknowledgements

The author thank Vesa Halava and Prof. Tero Harju for useful discussions.

References

[1] J.H. Conway, Unpredictable Iterations, in: Proceedings of the 1972 Number Theory Conference, University of Colorado, Boulder, Colorado, 1972, pp. 49–52.
[2] Jeffrey C. Lagarias, The $3x + 1$ problem and its generalizations, Amer. Math. Monthly 92 (1985) 3–23.
[3] G. Rozenberg, A. Salomaa (Eds.), Handbook of Formal Languages, vol. 1, Springer-Verlag, Berlin, Heidelberg, 1997, pp. 177–179.