

Questions Concerning the Usefulness of Small  
Universal Systems.

Liesbeth De Mol

Department for Philosophy

Centre for Logic and Philosophy of Science

Universiteit Gent, Blandijnberg 2, 9000 Gent, Belgium

[elizabeth.demol@ugent.be](mailto:elizabeth.demol@ugent.be)

## General Outline

- *Some questions concerning the usefulness of small universal computational systems: Work in progress.*
- *Results concerning tag systems: Work in progress.*

## Purpose

- *Convince you of fact that tag systems beg for more research.*

## Why are universal computational systems interesting?

- **Turing's seminal paper:** Unsolvability of the halting problem
- **Main technique proof unsolvable decision problems** for certain classes of systems: often through reduction of the halting problem (direct/indirect) to these systems.

## Are *small* universal machines interesting? Some opinions.

- “As a matter of “programming appreciation” it seems interesting that such a small structure can be so complicated” (Minsky, 1962)
- “We then proceed to represent the two-tape machine as a Post normal system with the object of obtaining particularly simple unsolvable decision problems (Minsky, 1961)”
- (...) “there are systems whose rules are simple enough to describe in just one sentence that are nevertheless universal. And this immediately suggests that the phenomenon of universality is vastly more common and important – in both abstract systems and nature – than has ever been imagined before.” (Wolfram, 2002)
- “We thought if we were to find the smallest universal universal machine then we could learn a great deal about computability – of course that wouldn’t be so” (J. McCarthy, quoted by Brady, 1988)
- “the question is an intensely tricky puzzle and has essentially no serious mathematical interest.” (Minsky, 1967)

## Why are (small) universal computational systems sometimes not interesting?

**Usual Method:** Find an encoding from a class of systems known to contain a universal system to a class of systems for which you want to prove this, using I/O encoding-decoding.

- A universal machine developed to emulate a specific computational system, will **exponentially slow down** when it has to emulate other computational systems. The initial conditions (configurations) get too long.
- **“Reasonable” and/or arbitrary initial conditions, in most of the cases lead to predictable behaviour.** Thus, if one would be interested in studying the behaviour of a certain class of computational systems, it could be more interesting to simply look at the behaviour of the machines they emulate)
- **Uninteresting when certain decision problems are involved.** (Two-way infinite UTM’s offer no challenge at all in Busy Beaver game competitions, since (as far as I know) the known UTM’s get all into an infinite loop when started with a blank tape.)

**Is it possible to prove the universality of certain computational systems which produce non-trivial behaviour with “reasonable” initial conditions in a reasonable time?**

- **Cellular Automata**

- There are very small universal CA which lead to non-trivial behaviour when started with an arbitrary initial condition.
- E.g. the universality proof of the 1-dimensional automaton, with 2 symbols and radius 1 (rule 110) by Matthew Cook.
- “*Must one compute something or does one just compute?*”: (Baldwin, FOM-list)

- **Other possibilities?** Open problem (as far as I know)

## Do we need other methods to prove the universality of such systems?

- **Reducibility of Collatz-like problems to small Busy Beaver record holders** Pascal Michel, 2004: Collatz-like problems in small Turing machines. *“Proof: The result is given by a tedious analysis of the behaviour of the machine”*
- **Universality of two-symbolic tag systems,  $v > 2$ ?** Open problem. The best method at present seems to be a combination of certain theoretical work and programming. (work in progress)

## But why would one be interested at all in such small universal computational systems?

- **A way to study universality and unsolvability in a more abstract way:** not through an explicit construction through a class of systems for which it is already known that they contain universal systems (and thus have an unsolvable decision problem), but through a study of the properties and behaviour of the systems themselves. After an analysis, one can begin to search for a more mathematically rigorous explanation of why the system gives rise to e.g. intractable behaviour. (however, respecting I/O conventions)

## Tag systems

- **Interesting framework to tackle the above mentioned questions, because of their abstractness:** they were developed by **Emil Post** in the process of searching for the most general and abstract form of mathematics.
- Another reason why they deserve more research is their intricate **connection with number theory**: their intractability seems to be caused by an asynchronization between the shift number and the tagging operation – you can't keep track of the remainder.

## Factorization of periods of Post's tag system

- 6 periods found which can not be factorized: 2, 4, 6, 10, 40, 66:

001101	$Str = 10$	$p = 2$
110111010000	$Str = 1100$	$p = 4$
10111011101000000	$Str = 111000$	$p = 6$
10111011101000011011101000000	$Str = 1110011000$	$p = 10$
0110100000000001101110111010011011101110100	$Str = 000001101011100$	$p = 40$

- All the other periods found (up to 124) are compositions of 2, 4, 6 and 10. Example:

$$10100110111011101000000 \quad Str = 10111000 \quad p = 8$$

# A simple encoding of Collatz-like functions into tag systems

## The $3x + 1$ -problem

Let  $T : \mathbb{N} \rightarrow \mathbb{N}$  be defined by:

$$T(x) = \begin{cases} \frac{x}{2} & \text{if } x \text{ is even} \\ 3x + 1 & \text{if } x \text{ is odd} \end{cases}$$

The  $3x + 1$ -problem is to determine for any  $x \in \mathbb{N}$ , whether  $T(x)$  will end in a loop  $T(2) = 1, T(1) = 1$ , after a finite number of iterates.

## Collatz-like functions: General form of the $3x+1$ -problem.

Given integers  $d \geq 2; a_0, a_1, \dots, a_{d-1}; r_0, r_1, \dots, r_{d-1}; x \in \mathbb{N}$  a Collatz-like function can be defined as follows:

$$G(x) = \begin{cases} m_0 & \text{If } n \equiv 0 \pmod{d} \\ m_1 & \text{If } n \equiv 1 \pmod{d} \\ \vdots & \\ m_{d-1} & \text{If } n \equiv (d-1) \pmod{d} \end{cases}$$

where  $m_i$  is either undefined or denotes an operation of the following form:

$$\frac{a_i(n-i)}{d} + b_i$$

## Encoding of the $3x+1$ -problem in a tag system.

The  $3n+1$ -problem can be easily encoded into a tag system, with  $v = 2, \mu = 3, \Sigma = \{\alpha, c, y\}$  when rewritten as follows:

$$T(2m) = m,$$

$$T(2m + 1) = 3m + 2.$$

If  $x$  is the natural number to be processed by  $T$ , the string corresponding with this input for the tag system is  $\underbrace{\alpha\alpha\alpha \dots \alpha}_x$  (denoted by  $(\alpha)^x$ ). The production rules of the tag system are:

$$\alpha \rightarrow cy$$

$$c \rightarrow \alpha$$

$$y \rightarrow \alpha\alpha\alpha$$

Now, if  $x \equiv 0 \pmod{2}$  then:

$$(\alpha)^x \rightarrow (cy)^{\frac{x}{2}}$$

$$(cy)^{\frac{x}{2}} \rightarrow \alpha^{\frac{x}{2}}$$

If  $x \equiv 1 \pmod{2}$  then:

$$(\alpha)^x \rightarrow y(cy)^{\frac{x-1}{2}}$$

$$y(cy)^{\frac{x-1}{2}} \rightarrow (\alpha)^{3(\frac{x-1}{2})+2}$$

## Encoding of arbitrary Collatz-like functions into Tag systems

Based on the above given encoding it is easy to derive a general scheme to encode Collatz-like function into Tag systems. The number of symbols, the alphabet as well as the shift number  $v$  are determined by the modulus  $d$  of the Collatz-like function  $G(dn + m) = a_m n + r_m$  with  $m = 0, \dots, d - 1$  to be emulated.

$$v = d$$

$$\mu \leq 2d + 3$$

$$\Sigma = \{h, \alpha, \alpha_0, \beta_0, \beta_1, \dots, \beta_{d-1}, b_0, b_1, \dots, b_{d-1}\}$$

$h$  being a halting symbol produced when  $G(dn + m)$  is undefined. If  $x$  is the natural number to be processed by  $G$ , the input for the tag system corresponding to this number is:

$$\alpha_0(\alpha)^x$$

The production rules corresponding to  $\alpha_0, \alpha$  and  $h$  are given by:

$$\alpha_0 \rightarrow \beta_{d-1}\beta_{d-2}\dots\beta_0$$

$$\alpha \rightarrow b_{d-1}b_{d-2}\dots b_0$$

$$h \rightarrow NIL$$

If  $G(dn + m)$  is defined, the production rules for  $\beta_m$  and  $b_m$ ,

with  $i = -(m + 1) \bmod d$  and  $m = x \bmod d$ , are:

$$\begin{aligned}\beta_m &\rightarrow (\alpha)^i \alpha_0 (\alpha)^{r_m} \\ b_m &\rightarrow (\alpha)^{a_m}\end{aligned}$$

If  $G(dn + m)$  is undefined, the production rules for  $\beta_m$  and  $b_m$  are:

$$\begin{aligned}\beta_m &\rightarrow h \\ b_m &\rightarrow h\end{aligned}$$

Applying these production rules to a given string  $\alpha_0 \alpha^x$  we get (with  $i = -(m + 1) \bmod d$  and  $m = x \bmod d$ ):

$$\begin{aligned}\alpha_0 \alpha^x &\rightarrow \beta_m \beta_{m-1} \dots \beta_0 (b_{d-1} b_{d-2} \dots b_0)^{\frac{x-m}{d}} \\ \beta_m \beta_{m-1} \dots \beta_0 (b_{d-1} b_{d-2} \dots b_0)^{\frac{x-m}{d}} &\rightarrow b_m b_{m-1} \dots b_0 (b_{d-1} b_{d-2} \dots b_0)^{\frac{x-m}{d}-1} (\alpha)^i \alpha_0 (\alpha)^{r_m} \\ b_m b_{m-1} \dots b_0 (b_{d-1} b_{d-2} \dots b_0)^{\frac{x-m}{d}-1} (\alpha)^i \alpha_0 (\alpha)^{r_m} &\rightarrow \underbrace{b_m b_{m-1} \dots b_0 (\alpha)^i}_{d} \alpha_0 (\alpha)^{(a_m-1) \frac{x-m}{d} + r_m} \\ \underbrace{b_m b_{m-1} \dots b_0 (\alpha)^i}_{d} \alpha_0 (\alpha)^{(a_m-1) \frac{x-m}{d} + r_m} &\rightarrow \alpha_0 (\alpha)^{a_m \frac{x-m}{d} + r_m}\end{aligned}$$

## The class of Tag systems with $v = 2$ and $\mu = 2$ is decidable

- In his *Absolutely unsolvable problems and relatively undecidable propositions – Account of an anticipation* (posthumously published in 1965 by Martin Davis) Post states that he has proven the decidability of the class of tag systems with  $v = 2$  and  $\mu = 2$ . However, Post never published this proof.
- The author wanted to know whether the  $3n + 1$ -problem is reducible to this class of tag systems. In order to do so, a proof of the solvability of this class seemed necessary.

## Proof of the solvability of 2x2-tag systems: Informal outline

- **Notational conventions**

- The alphabet  $\Sigma = \{0, 1\}$
- The words corresponding with "0" and "1" are respectively denoted by  $w_0$  and  $w_1$
- The length of a word  $w_x$  is denoted as  $l_x$ .
- The shortest of both words is denoted as  $l_{\min}$ ; the longest as  $l_{\max}$
- The total number of 0's in  $w_0w_1$  is indicated by  $\#0$ , the number of 1's by  $\#1$ .

- One only has to consider those cases for which  $l_0 = 1$  given a theorem proven by Maslov and Wang: for any tag system  $T$ , if  $v \geq l_{\max}$  or  $v \leq l_{\min}$ , then the decision problem for  $T$  is solvable.
- If every symbol in the words:

$$w_0 = \alpha_{0,1}$$

$$w_1 = \alpha_{1,1}\alpha_{1,2}\dots\alpha_{1,l_1}$$

has an equal chance to be “processed” by the tag system, then it is clear that if there are proportionally more 0’s than 1’s or vice versa the tag system will respectively halt or grow in a predictable fashion (it can be proven that from a given point on, possible contractions are bounded).

To be more precise, if the following set of equations does not have an integer solution and every symbol has an equal chance to be “processed” by the tag system:

$$\begin{cases} \frac{l_1 - v}{v - l_0} = \frac{m}{n} \\ a(m + n) = l_0 + l_1 \end{cases}$$

than the tag system is decidable. In these equations  $m$  and  $n$  respectively denote the proportion between #0 and #1. For example, with  $l_1 = 5$ , #0 has to be 3 times #1.

A 2x2-tag system, for which every symbol of  $w_0$  and  $w_1$  has an equal chance to be processed, with  $l_0 = 1$  thus must at least fulfill the following condition in order not to be solvable:

$$\begin{cases} \frac{l_1 - 2}{1} = \frac{m}{1} \\ a(m + 1) = l_0 + l_1 \end{cases}$$

It can be easily proven that these equations are only solvable when  $l_1 = 3$ , #0 = 2, #1 = 2.

- **Case:**  $l_1 = 3$ ,  $\#0 = 2$ ,  $\#1 = 2$

There are six possible tag systems in this class:

$$1 \rightarrow 110 \quad 0 \rightarrow 0$$

$$1 \rightarrow 011 \quad 0 \rightarrow 0$$

$$1 \rightarrow 101 \quad 0 \rightarrow 0$$

$$1 \rightarrow 001 \quad 0 \rightarrow 1$$

$$1 \rightarrow 100 \quad 0 \rightarrow 1$$

$$1 \rightarrow 010 \quad 0 \rightarrow 1$$

For all these cases it was proven that they are solvable: given any initial condition each of these tag systems will become periodic or halt (production of NILL-string)

- While it was proven that for  $l_1 > 3$  one can never find a tag system for which one has the “right” proportion between  $\#0$  and  $\#1$ , this does not immediately imply the solvability of the class of 2x2-tag systems with  $l_0 = “0”$  and  $l_1 > 3$ . Indeed, even if these equations do not have a solution for this class of tag systems, it might still be possible that not every symbol from the two words has an equal chance to be “processed”. From this it might follow that when the tag system is actually run, the equations are still satisfied.

- **Case:**  $l_1 > 3, w_0 = 1$

It is trivial to prove that this class is solvable: it will grow ad infinitum in the above mentioned provably decidable way. **Case**  $l_1 > 3, w_0 = 0$

- It can be easily proven that tag systems from this class, with  $\#1 = 1$ , always come to an end.
- In trying out several other tag systems belonging to this class, testing them with several initial conditions, the author observed that systems with  $\#1 = 2$ , always became periodic or halted while systems with  $\#1 > 2$ , always grew ad infinitum in a predictable fashion, or halted (with very specific initial conditions).
- It was proven that these observations can indeed be generalized (not completely formalized yet).