A Simple Discrete System with Chaotic Behavior*

Peter R.J. Asveld

Department of Computer Science, Twente University of Technology
P.O. Box 217, 7500 AE Enschede, The Netherlands

Abstract — We discuss the behavior of a particular discrete system, viz. Post's system of tag with alphabet $\{0,1\}$, deletion number $d=3$, and rules: $0 \to 00$, $1 \to 1101$. As initial string we consider all strings of length less than or equal to 15 as well as all ''worst case'' inputs of the form $(100)^m$ with $1 \le m \le 128$.

1. Introduction

During the past few decades some discrete time systems have been investigated that are relatively easy to describe or, equivalently, the rules according to which state changes take place are simple and concisely to formulate. But contrary to their descriptional simplicity their behavior can only be referred to in terms of unpredictable, irregular and chaotic. A typical example of this kind of systems is the $3x+1$-problem [4], i.e., given the function $f: \mathbb{N} \to \mathbb{N}$ defined by $f(1)=1$ and for each $x \ge 2$,

$$f(x) = \textbf{if } x \text{ is even } \textbf{then return } f(x/2) \textbf{ else return } f(3x+1)$$

then the question is: ''Is $f(x)$ defined for each $x \in \mathbb{N}$?''. As another example of this sort of systems we mention the difference equation $x_{n+1} = ax_n(1-x_n)$ with $0 < a \le 4$ and $0 < x_n < 1$; cf. [5] for the behavior of this equation and for similar examples.

In this note we consider Post's system of tag, i.e., a system T of the form $T = (\Sigma, d, P, \omega_0)$, where Σ is an alphabet (a finite set of symbols), d is a natural number (the *deletion number* of T), $P: \Sigma \to \Sigma^*$ is a function from symbols from Σ to strings over Σ, and ω_0 ($\omega_0 \in \Sigma^+$) is the *input* or *initial string* of T. Each such a system is deterministic and possesses a space of possible states which consists of all strings over Σ. We will now describe the way in which state changes take place or, equivalently, how we obtain the string ω_1 from ω_0, and subsequently ω_2 from ω_1, and so on. In general we derive ω_{t+1} from ω_t ($t \ge 0$) as follows:

- If the left-most symbol of ω_t equals σ, then we concatenate the string $P(\sigma)$ to the right end of ω_t. We denote the string obtained in this fashion by ω_t'.
- We delete the first d symbols from ω_t'; this yields ω_{t+1}.

Post's systems of tag have been introduced in [8,9]; cf. also [7] for an introduction. They are quite powerful devices; in particular they are able to simulate Turing-machine computations [6] and thus – assuming the Church-Turing Thesis – to perform every effective computation.

However, in this note we restrict our attention to a single, very simple instance of such a system. Viz. we consider the case in which $\Sigma = \{0,1\}$, $d = 3$, $P(0) = 00$, $P(1) = 1101$, and inputs ω_0 are taken from $\{0,1\}^+$. The first time this system occurs in the literature is in [8]; cf. also [9] and [7]. The amount of theory developed for this particular system is restricted to [10]. Examples of its behavior are in Figure 1 for the input string ω_0 equal to 1000 and to 1001, respectively.

* This paper will be included in D. Kleima et al (Eds.): *Liber Amicorum for E.W. Gröneveld* (1988), Department of Electrical Engineering, Twente University of Technology, Enschede, The Netherlands.

t	$\omega_0 = 1000$	$\omega_0 = 1001$
0	1000	1001
1	01101	11101
2	0100	011101
3	000	10100
4	00	001101
5	0	10100
6		001101
7		10100
8		. . .
9		. . .

Figure 1.

In general we can distinguish three types of behavior for this system; viz.
(A) ω_t vanishes. As an example consider the input $\omega_0 = 1000$; see Figure 1 where $\omega_6 = \lambda$ (λ denotes the empty string).
(B) There exist (minimal) natural numbers τ and π such that $\omega_\tau = \omega_{\tau+\pi}$, or – in other words – the behavior becomes cyclic with period π and threshold τ. See Figure 1 where the sequence starting with input $\omega_0 = 1001$ is ultimately cyclic with $\tau = 3$ and $\pi = 2$.
(C) ω_t does not vanish and neither does the sequence become ultimately periodic. This means that as time t proceeds, ω_t becomes longer and longer. For our system under consideration no examples of this behavior are known up to now.

Since theoretical results on this system are either absent or hard to produce, we will simulate this system for a number of input strings, classify its behavior into the categories (A), (B) and (C), and finally determine characteristic values like the time at which ω_t vanishes (case (A)), period and threshold (case (B)). Of course, our ultimate goal is to observe some regularities or patterns as the input string ω_0 varies over $\{0,1\}^+$. To this end §2 contains a few necessary definitions. In §3 we consider all input strings ω_0 the length of which – denoted by $|\omega_0|$ – satisfies $|\omega_0| \le 15$. §4 is devoted to input strings of the form $(100)^m$ where $1 \le m \le 128$ and $m \ne 110$. The case $m = 110$ is treated separately in §5 because it has not yet been resolved.

The present note is a survey of our simulations rather than a full account; the latter one will appear in [2]. Some preliminary results are mentioned in [1].

2. Concepts and Notation

Since the bit values in the input string at the $(3k+2)^{\text{nd}}$ and at the $(3k+3)^{\text{rd}}$ position ($k \ge 0$) are immaterial for the ultimate behavior of T, we assume that the bit values at these positions are equal to 0. So the shortest input strings with which we deal are 0, 1, 00, 10, 000, 100, 0000, 0001, 1000, 1001, 00000, 00010, 10000, 10010, and so on up to $(100)^5$. These strings will be numbered in this order by n; so we have $1 \le n \le 186$ and we write $\omega_0(n)$ for the n^{th} string in this list of input strings. The elements in the sequence generated by $T_n = (\Sigma, d, P, \omega_0(n))$ – where $\Sigma = \{0,1\}$, $d = 3$, $P(0) = 00$ and $P(1) = 1101$ – are denoted by $\omega_0(n)$, $\omega_1(n)$, $\omega_2(n)$, . . . , $\omega_t(n)$, ...

$H(n)$ is the time at which the sequence defined by T_n vanishes – cf. §1 (A) –

$$H(n) = \min\{t \mid \omega_t(n) = \lambda\}.$$

We take $H(n)$ equal to ∞ when the sequence defined by T_n does not vanish. In case the sequence does not vanish two other possibilities remain. First, it may happen that T_n generates at each

moment of time a new string; see §1 (C). However, no examples of input strings giving rise to such a behavior are known up to now. Secondly, it may occurs that T_n generates a string twice. Since the rewriting process is deterministic, T_n enters at that moment a cycle which it will never leave; cf. §1 (B). Characteristic quantities of such a cycle are the *period* $\pi(n)$, and the *threshold* $\tau(n)$ of the cycle,

$$\pi(n) = \min\{p \mid \exists t \in \mathbb{N}\colon \omega_t(n) = \omega_{t+p}(n)\},$$

$$\tau(n) = \min\{t \mid \omega_t(n) = \omega_{t+\pi(n)}(n)\}.$$

Finally, we use $M(n)$ to denote the maximum length of the string in the sequence

$$M(n) = \max\{|\omega_t(n)| \mid t \geq 0\},$$

whereas $F(n)$ is the first time that a string of maximum length occurs

$$F(n) = \min\{t \mid |\omega_t(n)| = M(n)\}.$$

In case T_n started with an input string $\omega_0(n)$ does not become ultimately periodic and its sequence does not vanish either, we have $M(n) = \infty$. On the other hand if the sequence does vanish we write $\tau(n) = H(n)$, $\pi(n) = 0$ and $|\omega_t(n)| = 0$ for each $t \geq \tau(n)$.

3. Short Input Strings

For small input strings ($1 \leq |\omega_0(n)| \leq 15$ or, equivalently, $1 \leq n \leq 186$) a straightforward approach happened to be sufficient. Viz. a small Pascal program has been written to generate the first thousand strings of a sequence. A file containing these strings can be sorted suppressing all but one in each number of equal strings by means of the UNIX[1] `sort -u` command. Then the UNIX[1] word count command `wc -l` gives the position where the first repetition in the sequence T_n occurs. A separate program determines the values of $M(n)$ and $F(n)$.

The detailed results of these simulations are mentioned in [2]; here we only present a summary in Figure 2, where we use the following notation. If $x(n)$ is an entity which depends on n, then $\min(x(n))$ [$\max(x(n))$, respectively] is the minimum [maximum] value of $x(n)$ on the interval $1 \leq n \leq 186$. The columns labeled by # and 1^{st} contain the number of times x equals this minimal [maximal] value and the minimal n for which $x(n)$ reaches this value, respectively. The last two lines in this table contain the values of $\pi(n)$ and of $|\omega_{\tau(n)}|$, i.e., the length of the first string that reappears in the sequence. Finally, l_n is an abbreviation of $|\omega_0(n)|$.

$x(n)$	$\min(x(n))$	#	1^{st}	$\max(x(n))$	#	1^{st}
$\tau(n)$	1	2	1	422	1	130
$M(n)$	1	1	1	56	16	89
$F(n)$	0	39	1	108	1	130
$\tau(n)/(l_n.M(n))$	0.01	1	119	1.50	1	2
$M(n)/l_n$	1.00	39	1	4.67	1	89
$F(n)/(l_n.M(n))$	0.00	39	1	0.50	1	2
$\pi(n)$	0, 2, 4, 6, 10					
$\lvert\omega_{\tau(n)}\rvert$	0, 5, 6, 11, 12, 13, 14, 15, 16, 17, 19, 31					

Figure 2.

[1] UNIX™ is a trademark of AT&T/Bell Laboratories.

Note that the fractions in Figure 2 have no theoretical foundation. Although dividing entities by l_n is natural, the division by $l_n.M(n)$ is just an ad hoc measure to obtain values in a reasonable interval.

4. Long Input Strings

Next we consider input strings of the form $(100)^m$ with $1 \le m \le 128$. These strings may be considered as ''worst case'' inputs because the first $m+1$ state changes will result in longer strings (So from the point of view that a sequence $\{\omega_t\}_{t \ge 0}$ ought to show a regular behavior – that means either it ought to vanish or it ought to become ultimately periodic – they have a bad start). Of course, this list of inputs is a sublist of the list introduced in the previous section, provided we omit the restriction ''$|\omega_0(n)| \le 15$'' or, equivalently, ''$1 \le n \le 186$''. In other words, we can express n as function of m; it is not difficult to show that $n_0 = 0$, and for each $m \ge 1$, we have $n_m = n_{m-1} + 3.2^m$, and hence $n_m = 6(2^m - 1)$.

For many values of m the naive approach to determine T_m's ultimate behavior as sketched in the previous section does not apply unless one has astronomic amounts of memory and computing time available. Therefore we followed a slightly different approach, viz. we simply count the lengths of all $\omega_t(m)$ in a variable of type `array[1..`$M(m)$`] of integer` – $M(m)$ and also $F(m)$ have been determined by a separate program in advance – while we let t range from 0 to C and from 0 to $C + C'$. Here C and C' are two large constants that we obtain by trial and error. Then these two arrays are compared; if C has been chosen large enough their entries are equal except for those with indices i in an interval $k \le i \le l$. Since a cycle can only be entered by rewriting a string of length equal to either $k-1$ or $l+1$ we use a separate program to determine the largest value of t for which $\omega_t(m)$ has length equal to either $k-1$ or $l+1$. In the neighborhood of this value of t we proceed as in §3.

$x(m)$	$\min(x(m))$	#	1st	$\max(x(m))$	#	1st
$\tau(m)$	4	1	1	67682437	1	114
$M(m)$	6	1	1	21300	1	114
$F(m)$	3	1	1	46494268	1	114
$\tau(m)/(l_m.M(m))$	0.03	1	16	13.62	1	24
$M(m)/l_m$	1.74	1	13	65.62	1	53
$F(m)/(l_m.M(m))$	0.01	1	109	7.30	1	120
$\pi(m)$	0, 2, 6, 10, 22, 28, 36, 52					
$\lvert\omega_{\tau(m)}\rvert$	0, 5, 15, 19, 31, 33, 37, 55, 67, 69, 73, 85, 87, 91, 109, 127, 157, 163, 559					

Figure 3.

The results for $1 \le m \le 128$ with $m \ne 110$ are listed in Figure 3 which is organized in the same way as Figure 2. Of course, l_m now denotes $|\omega_0(m)|$. The case $m = 110$ is postponed to the next section because it behaves differently.

5. An Open Problem — The Case $m = 110$

The computing time for the programs used in §3 is in the order of (tens of) minutes, whereas for the worst case inputs of §4 we need more time: viz. up to a few hours ($m = 114$).[2] The computing

[2] All programs have been executed on a DEC Micro-VAX-2000™ using the Ultrix™ V2.0 operating system.

time for the case $m = 110$ will probably be expressed in months or even in years. In four months time, we have simulated 40.10^9 steps and it is still not clear which of the cases (A), (B) or (C) applies to $m = 110$. The only definite facts which we obtained so far, are

$$M(110) \geq 1227766, \quad F(110) \geq 34830458284, \quad H(110) \geq \tau(110) \geq 34830458288.$$

θ	$M(\theta)$	$F(\theta)$	$\theta/l_{110}.M(\theta)$	$M(\theta)/l_{110}$	$F(\theta)/l_{110}.M(\theta)$
4.10^9	434384	3101002790	27.9	1316	21.6
8.10^9	477430	6835757946	50.8	1447	43.4
12.10^9	489954	11606868974	74.2	1484	71.8
16.10^9	939356	15961096134	51.6	2847	51.5
20.10^9	995762	16520573206	60.9	3017	50.3
24.10^9	1104084	22911015168	65.9	3346	62.9
28.10^9	1104084	22911015168	76.8	3346	62.9
32.10^9	1104084	22911015168	87.8	3346	62.9
36.10^9	1227766	34830458288	88.9	3721	86.0
40.10^9	1227766	34830458288	98.7	3721	86.0

Figure 4.

In Figure 4 the values of fractions similar to those in Figures 2 and 3 are displayed. Instead of $\tau(110)$ we take the interval length θ; $M(\theta)$ is the maximal string length achieved in the interval $0 \leq t \leq \theta$, whereas $F(\theta)$ equals the largest t such that $t \leq \theta$ and $|\omega_t(110)| = M(\theta)$. Comparing these values with the corresponding entries in Figures 2 and 3, emphasizes the observation that the case ''$m = 110$'' is particular.

6. Concluding Remarks

Even a long and careful consideration of the detailed tables in [2] yields nothing but a few trivial observations. So the behavior of this instance of Post's system of tag may be called chaotic indeed, since hardly any pattern or regularity emerges. Probably the only exception is the fact that π is always even; but this trivial fact can be explained in advance.

	(A)		(B)		(C)		(D)		Total	
	#	%	#	%	#	%	#	%	#	%
§3 ($1 \leq n \leq 186$)	67	36.0	119	64.0	0	0.00	0	0.00	186	100.00
§4 ($1 \leq m \leq 128$)	20	15.6	107	83.6	0	0.00	1	0.8	128	100.00
Total*	86	27.8	222	71.9	0	0.00	1	0.3	309	100.00

* Duplicates have been excluded.

Figure 5.

As a conclusion we consider the distribution over the categories introduced in §1. Apart from the categories (A), (B) and (C) distinguished in §1 we define a fourth category (D) for those cases that have not been resolved up to now. The division over these categories is given in Figure 5. Note that the lists of inputs from §3 and §4 are not disjoint: they both contain the cases $n = 6$, 18, 42, 90 and 186 ($1 \leq m \leq 5$).

As already mentioned a couple of times before, the classification of the case $m = 110$ is still open. It remains a suitable or – better – a promising candidate for category (C).

References

1. P.R.J. Asveld: On a Post's system of tag, Memorandum INF-88-20, Department of Computer Science, Twente University of Technology, Enschede, The Netherlands.
2. P.R.J. Asveld: On Post's system of tag with $0 \to 00$ and $1 \to 1101$ as productions, Memorandum INF-89-??, Department of Computer Science, Twente University of Technology, Enschede, The Netherlands, (in preparation).
3. M. Davis: *The Undecidable – Basic Papers on Undecidable Propositions, Unsolvable Problems and Computable Functions* (1965), Raven, New York.
4. J.C. Lagarias: The $3x+1$ problem and its generalizations, *Amer. Math. Monthly* **92** (1985) 3-23.
5. H.A. Lauwerier: Chaos and order, *Nieuw Archief voor Wiskunde* **IV**, **2** (1984) 373-401.
6. M.L. Minsky: Recursive unsolvability of Post's problem of ''tag'' and other topics in the theory of Turing machines, *Annals of Math.* **74** (1961) 437-455.
7. M.L. Minsky: *Computation – Finite and Infinite Machines* (1972), Prentice-Hall, Englewood Cliffs, N.J.
8. E.L. Post: Formal reductions of the general combinatorial decision problem, *Amer. J. Math.* **65** (1943) 197-215.
9. E.L. Post: Absolutely unsolvable problems and relatively undecidable propositions – account of an anticipation, pp. 340-433 in [3].
10. S. Watanabe: Periodicity of Post's normal process of tag, Proceedings of the Symposium on Mathematical Theory of Automata 1962, Polytechnic Press, Brooklyn, N.Y. (1963) pp. 83-99.