Notre Dame Journal of Formal Logic
Volume 49, Number 2, 2008

# An Undecidable Property of Recurrent Double Sequences

Mihai Prunescu

**Abstract** For an arbitrary finite algebra $(A, f(\cdot,\cdot), 0, 1)$ one defines a double sequence $a(i, j)$ by $a(i, 0) = a(0, j) = 1$ and $a(i, j) = f(a(i, j-1), a(i-1, j))$. The problem if such recurrent double sequences are ultimately zero is undecidable, even if we restrict it to the class of commutative finite algebras.

## 1 Introduction

**Definition 1.1** Consider an arbitrary finite algebra $\mathfrak{A} = (A, f, 0, 1)$ where $f : A \times A \to A$ is a binary operation and 0, 1 are two constants. We call 1 the start symbol. The recurrent double sequence associated to $\mathfrak{A}$ is a function $a : \mathbb{N} \times \mathbb{N} \to A$ defined as follows:

$$a(i, j) = \begin{cases} 1 & \text{if } i = 0 \ \vee \ j = 0, \\ f(a(i, j-1), a(i-1, j)) & \text{if } i > 0 \ \wedge \ j > 0. \end{cases}$$

If $f$ is commutative, the recurrent double sequence shall be symmetric:

$$a(i, j) = a(j, i).$$

**Definition 1.2** The recurrent double sequence $a(i, j)$ is said to be ultimately zero if

$$\mathfrak{A} \models \exists N \in \mathbb{N} \, \forall i, j \in \mathbb{N} \ \ i > 0 \wedge j > 0 \wedge i + j > N \longrightarrow a(i, j) = 0.$$

**Theorem 1.3** *It is undecidable if the recurrent double sequence defined by an algebra $\mathfrak{A}$ is ultimately zero. This question remains undecidable if it is restricted to the class of commutative finite algebras.*

Received November 28, 2007; accepted December 3, 2007; printed March 12, 2008
2000 Mathematics Subject Classification: Primary, 03D10
Keywords: recurrent computation, double sequence, Turing machine, undecidable property, finite commutative algebra
 10.1215/00294527-2008-004

Of course, the commutative case implies the general one. However, I believe that starting with an easier sketch of proof for the general case provides a natural introduction to the matter.

**Definition 1.4** We call Z the class of algebras $\mathfrak{A}$ such that the corresponding double sequence $a$ is ultimately zero. We call CZ the class of *commutative* algebras $\mathfrak{A} \in$ Z.

**Definition 1.5** The recurrent double sequence $a : \mathbb{N} \times \mathbb{N} \to A$ of $\mathfrak{A}$ shall be seen as an infinite matrix with $a(0,0) = a_{0,0} = 1$ in the left upper corner. Let

$$D_n = \{a_{i,j} \mid i + j = n\}$$

be the $n$th diagonal. Then $a = \cup D_n$ is a partition of the recurrent double sequence $a$.

This small research started with a sequence of computer experiments done by the author. The algebras $(A, f, 0, 1)$ were finite fields $A$ equipped with ternary polynomials $f \in A[x, y, z]$ as operations. The recurrent sequences were generated by the rule $a(i, j) = f(a(i, j-1), a(i-1, j-1), a(i-1, j))$. Finite field elements have been interpreted as colors and the resulting matrices were displayed as colored images.[1] For the sake of symmetry, only polynomials that are symmetric in $x$ and $z$ were applied. Applying the interpolation over finite field, one sees that to consider only polynomials does not represent any restriction of the generality. In the special case of the linear polynomials $x + my + z$ produces self-similar sets; see [2] for a proof. Linear symmetric polynomials in $x$ and $z$ of the form $ax + by + az$ are no more self-similar in general—one recognizes self-similarity partially overlapped with other periodic and quasi-periodic phenomena. The unbelievable variety of images produced by general polynomials (even for small fields like $\mathbb{F}_5$) suggested that the recurrent double sequences are Turing complete and that a lot of their properties are undecidable.

Other relevant results on topics related to the result proved here can be found in the monograph [1]. The unique result used here is the Theorem of Rice [3] in its modern formulation concerning sets of (codes of) Turing machines, as stated, for example, in [4].

## 2 The General Problem

In this section we present a straightforward interpretation of Turing machines in the recurrent double sequences.

**Definition 2.1** An instance of the Halting Problem is a pair $(M, w)$, where $M = (\Sigma, Q, q_0, q_s, \bar{b}, \delta)$ is a Turing machine and $w \in \Sigma^*$ is an input for $M$. Here the tape of $M$ is infinite in both directions; $\Sigma$ is the alphabet of $M$; $Q$ is $M$'s set of states; $q_0$ and $q_s$ are the start state and, respectively, the stop state; $\bar{b} \in \Sigma$ is the blank symbol; and $\delta : \Sigma \times Q \to \Sigma \times Q \times \{R, L, S\}$ is the transition function.

**Lemma 2.2** *To every instance $(M, w)$ of the Halting Problem one can algorithmically associate a finite algebra $\mathfrak{A} = (A, f, 0, 1)$ such that $\mathfrak{A} \in$ Z if and only if, for input $w$, the machine $M$ stops and after stopping the tape is cleared.*

**Proof** If Lemma 2.2 is true, then the problem Z is not algorithmically solvable. This is true because to stop with a cleared tape is an undecidable property of Turing machines, according to the Theorem of Rice.

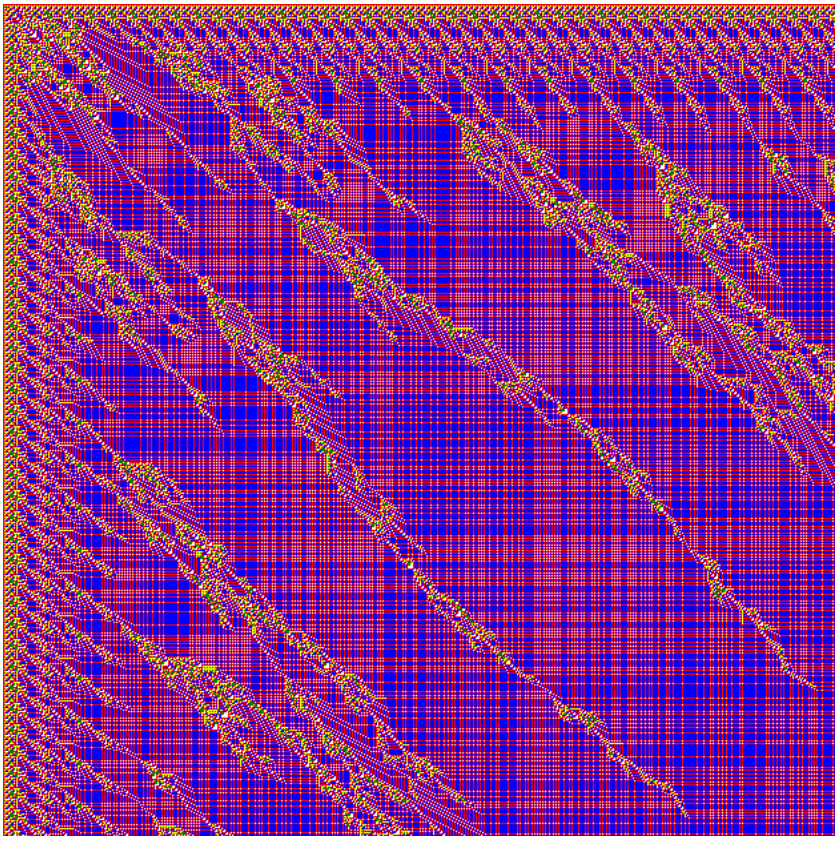

**Figure 1** $x^4 + z^4 - x^4yz^4 + x^3z^3 \in \mathbb{F}_5[x, y, z]$.

Let $(M, w)$ be an instance of the Halting Problem. The algebra $\mathfrak{A}$ shall be constructed step by step following its generated double sequence.

**Step 1** The start symbol of $\mathfrak{A}$ is a new letter 1 that doesn't belong to the Turing machine $M$. On the other hand, $0 \in \mathfrak{A}$ shall be the blank symbol $\bar{b}$ of $M$.

**Step 2** Let the input $w \in \Sigma^*$ be the word $w_1 \dots w_n$. Using a set $U$ of new letters one defines $f$ such that the content of a diagonal $D_w$ of the double sequence is exactly

$$1\ 0\ 0\ \delta_0\ w_1\ \dots\ w_n\ 0\ 0\ 1\,.$$

The letters of $U$ shall be used only for this goal and never again. The simulation of the Turing machine starts with the diagonal $D_w$.

**Step 3** The just constructed diagonal $D_w$ is said to be a diagonal of type 0. Starting with $D_w$, diagonals are alternatively of types 0, 1, 0, 1, and so on. Successive diagonals of type 0 simulate successive configurations of the Turing machine. The diagonals of type 1 between them are used to transfer the information from a simulated configuration to the next one.

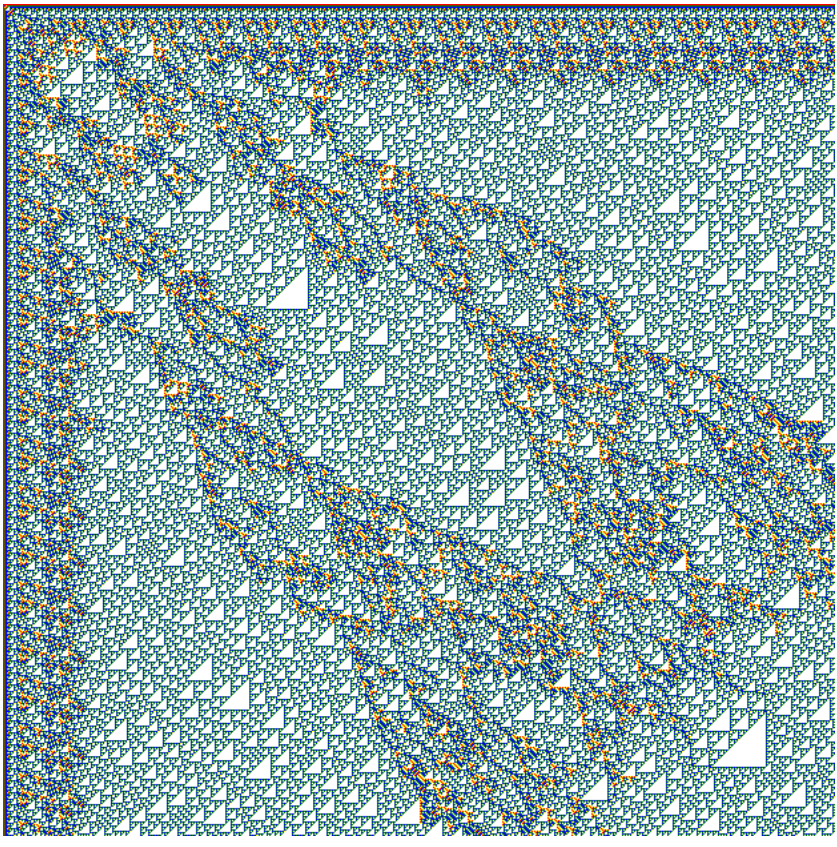

**Figure 2** $2x^3 + 2z^3 + y^2 - x^2z^2 \in \mathbb{F}_5[x, y, z]$.

**Step 4** The alphabet used for diagonals of type 0 contains $\Sigma \cup (\Sigma \times Q)$. We denote letters from $\Sigma \times Q$ with $\delta_i$. The meaning of the letter '$(a, q)$' is that the head of $M$ reads $a$ with $M$ being in the state $q$.

**Step 5** If $\Gamma_0$ is the alphabet used for diagonals of type 0, the alphabet used for diagonals of type 1 will be $\Gamma_1 = (\Gamma_0 \times \Gamma_0 \setminus \{(0, 0)\}) \cup \{0\}$.

**Step 6** The function $f$ is defined on diagonals of type 0 in the obvious way $f(a, b) = (a, b)$ if at least one of $a$ and $b$ is not 0 or 1, and $f(0, 0) = f(1, 0) = f(0, 1) = 0$.

**Step 7** The function $f$ is defined on diagonals of type 1 such that if the element $a_{i,j}$ of the last diagonal of type 0 simulated a certain cell of the tape of $M$ at a given time $k$, then the cell $a_{i+1,j+1}$ simulates the same cell at the time $k + 1$. The following example shows how the diagonal of type 1 in between makes possible that the element $a_{i+1,j+1}$ gets information from three successive cells: $a_{i,j}$ and the

elements $a_{i-1,j+1}$ and $a_{i+1,j-1}$ simulating its neighbors on the tape of $M$.

$$\begin{array}{ccc} & & b \\ & \delta & (\delta, b) \\ a & (a, \delta) & c \end{array}$$

**Step 8** Every diagonal of type 0 is with two cells longer than the precedent one and the head makes one step per time, so there is no danger that the simulation leaves the matrix or even that the simulation meets the first row or the first column.

**Step 9** For the special letter $\delta = (0, q_s)$ we define $f$ such that $f(\delta, 0) = f(0, \delta) = 0$. This makes the double sequence ultimately zero if and only if the machine stops with a cleared tape.

**Step 10** Now take $A$ to be $\{1\} \cup U \cup \Gamma_0 \cup \Gamma_1$ and take an $f : A \times A \to A$ respecting all the conditions given above. □

## 3 The Commutative Problem

**Definition 3.1** Let $\Gamma \neq \varnothing$ be a set and $\equiv$ be the partition of $\Gamma \times \Gamma$ consisting of the following sets: for all $a \in \Gamma$ the singleton sets $\{(a, a)\}$ and for all $a, b \in \Gamma$ with $a \neq b$ the two-element sets $\{(a, b), (b, a)\}$. Then $\equiv$ is an equivalence relation over $\Gamma$. Consider the set of equivalence classes,

$$\Gamma \cdot \Gamma = (\Gamma \times \Gamma)/\equiv,$$

which is the set of unordered pairs of elements of $\Gamma$. We denote the equivalence class of $(a, b)$ with $[a, b]$ and call this the unordered pair of $a$ and $b$.

**Lemma 3.2** *To every instance $(M, w)$ of the Halting Problem one can algorithmically associate a commutative finite algebra $\mathfrak{A} = (A, f, 0, 1)$ such that $\mathfrak{A} \in$ CZ if and only if, for input $w$, (the machine $M$ stops with cleared tape without having done any step in the negative side of the tape) or (the machine $M$ makes at least one step in the negative side of its tape and the first time when $M$ makes such a step the tape of $M$ is cleared).*

**Proof** If Lemma 3.2 is true, then CZ is not algorithmically solvable. This is true because the given condition is an undecidable property of Turing machines, according to the Theorem of Rice.

The function $f$ shall be constructed again in parallel with the recurrent double sequences. It shall be again so that on some special (half of) diagonals one simulates successive configurations of the Turing machine on input $w$. The function $f$ being commutative, one cannot make directly the difference between Left and Right. To overcome this difficulty one can try to double the number of letters of $\Gamma$ and write every letter $c$ as $cc'$. The function $f$ should now act symmetrically on diagonals of type 0, so we define $f(a, b)$ to be the unordered pair $[a, b]$. This strategy is not sophisticated enough: if we look at words $aba$ and $bab$ on a diagonal of type 0, they both produce a word $xx$ on the following diagonal of type 1, where $x = [a, b]$. This means that this encoding may lose essential information. The solution shall be to triple the number of letters and to encode every letter $c$ in a sequence $cc'c''$, where $c'$ and $c''$ are new letters used only for this goal.

**Step 1** The start symbol of $\mathfrak{A}$ is a new letter 1 that does not belong to the Turing machine. Also the letter 0 of $\mathfrak{S}$ is now a new letter.

**Step 2** Let the input $w \in \Sigma^*$ be the word $w_1 \dots w_n$. Using a set $U$ of new letters one defines $f$ commutatively such that the content of a diagonal $D_w$ of the recurrent double sequence is exactly

$$10^9\ w_n'' w_n' w_n\ \dots\ w_1'' w_1' w_1\ \delta_0'' \delta_0' \delta_0 \delta_0\ 000\ \delta_0 \delta_0' \delta_0''\ w_1 w_1' w_1''\ \dots\ w_n w_n' w_n''\ 0^9 1\,.$$

Here $0^9$ means a word built up by 9 zeros. The letters of $U$ shall be used only for this goal and then never again. The simulation of the Turing machine starts with this diagonal.

**Step 3** From now on we use the words Left and Right relatively to the recurrent double sequence, as Left and Right of the current diagonal, which is written down as in the preceding item.

**Step 4** The just constructed diagonal $D_w$ is a diagonal of type 0. This time there are 8 types of diagonals: types $0, 1, \dots, 7$. Starting with $D_w$ diagonals are of types $0, 1, \dots, 7, 0, 1, \dots, 7$, and so on. Successive diagonals of type 0 simulate successive configurations of the Turing machine. The diagonals of types $1, \dots, 7$ between them are used to transfer the information from a simulated configuration to the next one.

**Step 5** The alphabet $\Gamma_0$ used for diagonals of type 0 contains three disjoint copies of the set $\Sigma \cup (\Sigma \times Q) \setminus \{\bar{b}\}$. Every letter $c$ or $\delta$ has copies $c'$, $c''$ and $\delta'$, $\delta''$, respectively. The fact that $c \in \Sigma$ is contained in a cell of the Turing machine is encoded by the connected word $cc'c''$ occurring in the right-hand side of a diagonal of type 0. $\delta \in \Sigma \times Q$ is encoded in the right-hand side of a diagonal of type 0 by a word $\delta\delta'\delta''$. The blank symbol $\bar{b}$ as content of a cell of the Turing machine is always encoded in the simulation by the word 000 on a diagonal of type 0. In the left-hand side of the development the codes for $c$ and $\delta$ are $c''c'c$, $\delta''\delta'\delta$, respectively.

**Step 6** Let $\Gamma_0$ be the alphabet used for diagonals of type 0. For $i = 1, 2, \dots, 7$ the alphabet used for diagonals of type $i$ will be $\Gamma_i = (\Gamma_{i-1} \cdot \Gamma_{i-1} \setminus \{[0,0]\}) \cup \{0\}$.

**Step 7** The function $f$ is defined on diagonals of type $i = 0, 1, \dots, 6$ in the obvious way $f(a,b) = [a,b]$ if at least one of $a$ and $b$ are not 0 or 1, and $f(0,0) = f(1,0) = f(0,1) = 0$.

**Step 8** The function $f$ shall be defined on diagonals of type 7 such that *if the element $a_{i,j}$ of the last diagonal of type* 0 *contains a letter ($c$, $c'$, $c''$, $\delta$, $\delta'$, $\delta''$, or* 0*) that appears in a subword of length* 3 *simulating a cell of the tape of $M$ at a given time $k$, then the element $a_{i+4,j+4}$ of the recurrent double sequences shall be the corresponding letter of the diagonal word simulating the configuration of $M$ at time $k+1$.* This is done in the following way.

**Step 9** Let $a_{i,j} = \delta' \in \Gamma_0$ be a part of the following segment of simulation in a diagonal of type 0: $\ldots cc'c''\delta\delta'\delta''dd'd'' \ldots$, and suppose that in the next configuration the tape-cell containing $\delta$ shall contain $e \in \Sigma$. As proved in the postponed Lemma 4.6, $a_{i+3,j+4} = \alpha$ and $a_{i+4,j+3} = \beta$, with $\alpha, \beta \in \Gamma_7$, such that $\alpha$ encodes the word $cc'c''\delta\delta'\delta''dd'$ or its reverse and $\beta$ encodes the word $c'c''\delta\delta'\delta''dd'd''$ or its reverse. One has either the words themselves (if we look at the right-hand side of the double sequence) or the reversed words (if we look at the left-hand side of the double sequence). If we are in the right-hand side, the matching of the encoded words looks like

$$\begin{array}{cccccccccc} c & c' & c'' & \delta & \delta' & \delta'' & d & d' & & \\ & c' & c'' & \delta & \delta' & \delta'' & d & d' & d'' & . \end{array}$$

If we are in the left-hand side, the matching is

$$\begin{array}{cccccccccc} & & d' & d & \delta'' & \delta' & \delta & c'' & c' & c \\ d'' & & d' & d & \delta'' & \delta' & \delta & c'' & c' & . \end{array}$$

In both cases the matching is a word of length 7 around $\delta'$, so the value $f(\alpha, \beta)$ is uniquely determined to be $e'$, where $e \in \Sigma$ is the letter that shall replace $\delta$ in the corresponding tape-cell of $M$ in the next configuration. The same arguments work for every connected subword of length 8 which is disjoint from the central 000 word, like $c''\delta\delta'\delta''dd'd''e$, and so on.

**Step 10** Every diagonal of type 0 is with eight elements longer then its predecessor of type 0 (four elements left-hand and four elements right-hand) and the simulation needs at most three elements more per step, so there is no danger that the simulation leaves the matrix or even that it meets the first row.

**Step 11** If the connected subword $\delta\delta'\delta''$ with the special letter $\delta = (a, q_s)$ arises, then we define $f$ such that the corresponding elements in the next diagonal of type 0 are all $aaa''$ for $a \neq \bar{b}$ and 000 for $a = \bar{b}$. From this moment the simulation of the Turing machine ended.

**Step 12** For the connected subwords of length 8 containing the central 000 the function $f$ is defined such that *words of the type* $c''c'c000cc'c''$ *are preserved in the next configuration*. Words $\delta''\delta'\delta000\delta\delta'\delta''$ are replaced with $e''e'e000ee'e''$ if $\delta = (a, q) \to (e, q', R)$. Words $\delta''\delta'\delta000\delta\delta'\delta''$ are replaced with $a''a'a000aa'a''$ if $\delta = (a, q) \to (e, q', L)$. This is the other legal way to stop the computation, as already stated in Section 3.

**Step 13** Now take $A$ to be $\{1\} \cup U \cup_{i=0}^{7} \Gamma_i$ and $f : A \times A \to A$ to respect all the conditions given above. □

## 4 Symmetric Codes

**Definition 4.1** Let $\Gamma_0$ be a finite alphabet with $\geq 2$ letters and $0 \in \Gamma_0$ a special letter. We define the sequence of alphabets $\Gamma_i$ such that $\Gamma_{i+1} = (\Gamma_i \cdot \Gamma_i \setminus \{[0, 0]\}) \cup \{0\}$ and $f : \Gamma_i \times \Gamma_i \to \Gamma_{i+1}$ such that $f(a, b) = [a, b]$ if at least one of the arguments is not 0, respectively, $f(0, 0) = 0$. For an alphabet $\Gamma$ let $\Gamma^*$ be the set of words over $\Gamma$ and $\Gamma^{\geq k}$ the set of words of length $\geq k$ over $\Gamma$. The set of words of length $k$ shall be simply denoted $\Gamma^k$.

**Definition 4.2** Let $\pi_i : \Gamma_i^{\geq 2} \to \Gamma_{i+1}^*$ given as $\pi_i(w_1 \dots w_n) = f(w_1, w_2) f(w_2, w_3) \dots f(w_{n-1}, w_n)$. Let $\pi : \Gamma_0^8 \to \Gamma_7$ given as $\pi(w) = \pi_6\pi_5\pi_4\pi_3\pi_2\pi_1\pi_0(w)$. We call the words $w$, $\pi_0(w)$, $\pi_1\pi_0(w)$, $\dots$, $\pi_5\pi_4\pi_3\pi_2\pi_1\pi_0(w) = \pi(w)$ the coding steps.

**Definition 4.3** Now let $\Gamma_0$ be the alphabet defined in Section 3. Let $E$ be the set of all words in $\Gamma_0^8$ that can possibly arise as diagonal words during a simulation. They are exactly the connected subwords of length 8 in all words $aa'a''bb'b''cc'c''dd'd''$ where $a, b, c, d \in \Sigma \cup \Sigma \times Q$ are not necessarily different, and if some $e \in \{a, b, c, d\}$ are 0, then the corresponding $e' = e'' = 0$. The restriction of $\pi$ to $E \to \Gamma_7$ shall be called simply $\pi$.

**Definition 4.4** Let $S$ be the set of connected subwords of length 8 in all words $ba''a'a000aa'a''b$ where $a, b \in \Sigma \cup \Sigma \times Q$ are not necessarily different, and if $a = 0$, then the corresponding $a' = a'' = 0$. Again the restriction of $\pi$ to $S \to \Gamma_7$ shall be called simply $\pi$.

**Definition 4.5** For a word $w \in \Gamma^*$, $w = w_1 \dots w_n$, call $\sigma(w)$ the reversed word $w_n \dots w_1$.

**Lemma 4.6** *For all $v, w \in E \cup S$, if $\pi(w) = \pi(v)$, then $w = v$ or $w = \sigma(v)$.*

**Proof** The proof works by direct checking. Given $\pi(w)$, one reconstructs $w$ backward.

**Words in $v \in E$:** It is enough to check the worst cases with repetitions of letters. Start with the word $cc'c''cc'c''cc'$. The coding steps have the following form: $x_1x_2x_3x_1x_2x_3x_1x_2$, $y_1y_2y_3y_1y_2y_3y_1$, $z_1z_2z_3z_1z_2z_3$, $v_1v_2v_3v_1v_2$, $t_1t_2t_3t_1$, $u_1u_2u_3$, $s_1s_2$, $\alpha$, where $\alpha \in \Gamma_7$. Starting with $\alpha$, one gets back $\alpha = [s_1, s_2]$ so the two possibilities for the diagonal of type 6 are $s_1s_2$ and $s_2s_1$. The first one leads directly to $w$, the other one directly to $\sigma(w)$; there are not other possibilities to reconstruct the word.

For a complete proof of the lemma, one has to check the following worst cases: (a) all the connected subwords of length 8 in $cc'c''000cc'c''000000$ and (b) all the connected subwords of length 8 in $cc'c''cc'c''cc'c''000$. All these cases have the following common property: at all levels of coding, including the level 0, two successive letters are equal if and only if they are 0.

**Words in $v \in S$:** The words occurring here are the exceptions in our symmetric encoding: They do not enjoy the property that successive letters at every level are equal if and only if they are both 0, but they are, however, well behaved even by being the only words in question that don't enjoy this property. Look at $c''c'c000cc'$. The coding steps are $x_1x_2x_300x_3x_2$, $y_1y_2y_30y_3y_2$, $z_1z_2z_3z_3z_2$, $v_1v_2v_3v_2$, $t_1t_2t_2$, $s_1s_2$, $\alpha$. In decoding we have again the choice $s_1s_2$ or $s_2s_1$. If we choose $s_1s_2$, that can backward develop only in $t_1t_2t_2$, and so on. One easily checks all other words in question. □

To recapitulate: Letters have been encoded by directed words of the form $cc'c''$ or $\delta\delta'\delta''$ in order to make the difference between the left and the right neighbor in a symmetric double sequence. For symmetrically encoded words of length 8 one needs 7 supplementary types of diagonals. A letter of type 7 encodes a word of length 8 and so always has information from three successive simulated Turing cells. The

common part of two successive subwords of length 8 has length 7 and so always has a central letter: this is the letter to copy or replace on the next diagonal of type 0.

The referee pointed out that the technique applied in the last proof is related with the set-theoretic definition of ordered pairs given by Wiener [5] and Kuratowski $(a, b) = \{\{a\}, \{a, b\}\}$. Indeed, the principle of backward reconstruction used here is based on this idea: one chooses an ordering $(a, b)$ for some unordered pair $[a, b]$ by looking for an occurrence of $a$ or $b$ in the unordered pairs in the neighborhood. Both constructions done here work in polynomial time.

## Note

1. The full color output of the images on pages 145 and 146 can be viewed in the online version of this journal issue at Project Euclid (http://projecteuclid.org/ndjfl).

## References

[1] Börger, E., E. Grädel, and Y. Gurevich, *The Classical Decision Problem*, Perspectives in Mathematical Logic, Springer-Verlag, Berlin, 1997. Zbl 0865.03004. MR 1482227. 144

[2] Prunescu, M., "Self-similar carpets over finite fields," presented at CiE 2007, Siena, 2007. 144

[3] Rice, H. G., "Classes of recursively enumerable sets and their decision problems," *Transactions of the American Mathematical Society*, vol. 74 (1953), pp. 358–66. Zbl 0053.00301. MR 0053041. 144

[4] Schöning, U., *Theoretische Informatik—Kurz Gefaßt*, Spektrum Akademischer Verlag, Heidelberg, 1997. MR 1244107. 144

[5] Wiener, N., "A simplification of the logic of relations," Proceedings of the Cambridge Philosophical Society, vol. 17 (1914), pp. 387–90. Zbl 45.0122.16. 151

## Acknowledgments

The author extends his thanks to the American Mathematical Society.

Brain Products
Freiburg
GERMANY
and
Institute of Mathematics of the Romanian Academy
Bucharest
ROMANIA
mihai.prunescu@math.uni-freiburg.de