TWO RESULTS REGARDING NONCOMPUTABILITY FOR UNIVARIATE CELLULAR AUTOMATA

A. L. Toom and L. G. Mityushin

UDC 62-507:518.5

Operator P with local interaction acts on an ensemble of islands, i.e., finite subsets of an integer-valued straight line. Two assertions regarding algorithmic noncomputability for such operators are proved.

§1. BASIC FORMULATIONS

A univariate cellular automaton is an operator $P: X_n \to X_n$ of special type. It acts on space X_n consisting of all bilaterally infinite sequences whose members are integers from 0 to n:

$$X_n = S_n^{\mathbb{Z}} = \{x\}, \quad x = (x_i), \quad x_i \in S_n = \{0, \ldots, n\}, \quad i \in \mathbb{Z}.$$

Automaton P acting on X_n (we will generally omit the word "cellular") is specified by a natural radius number r and function $f: S_n^{2r+1} \to S_n$, and it has the following form. For any $x \in X_n$ the i-th component of Px is

$$(Px)_i = f(x_{i-r}, \ldots, x_{i+r}). \tag{1}$$

This paper, together with its predecessor [1], is concerned with the possibility of algorithmically

Translated from Problemy Peredachi Informatsii, Vol. 12, No. 2, pp. 69-75, April-June, 1976. Original article submitted June 19, 1974.

predicting the behavior of such automata. We will always assume that

$$f(0,\ldots,0)=0. \tag{2}$$

Definition 1. We will call $x\in X_n$ an island if only a finite number of its components x_i are nonzero; x will be called an empty island if all its components are zeros.

We denote the ensemble of islands in X_n by X_n'. By virtue of condition (2), P always carries X_n' into itself, and we will always consider the contraction of P on X_n'. It is also obvious that P always carries an empty island into itself.

Definition 2. We will say that automaton P washes out island x if there exists a natural t such that $P^t x$ is an empty island.

Definition 3. We will call automaton P binary if it acts on X_1. Binary automaton P will be called monotonic if the function f that specifies it (a Boolean function in this case) is monotonic, i.e.,

$$a_{-r}\leqslant a_{-r}',\ldots,a_r\leqslant a_r'\Rightarrow f(a_{-r},\ldots,a_r)\leqslant f(a_{-r}',\ldots,a_r'). \tag{3}$$

The basic results of this paper are embodied in the following two propositions.

Proposition 1. There exists an island $x^0\in X_1'$, e.g., an island with two ones, of the form

$$\ldots 001100\ldots, \tag{4}$$

such that the determination of whether P washes out island x^0 by any monotonic binary automaton P is noncomputable.

Proposition 2. There exists a monotonic binary automaton P_0 such that the determination of whether P_0 washes out island x is noncomputable by any island $x\in X_1'$.

For these two assertions to have a precise meaning, we must choose some method of numbering islands in X_1' and monotonic binary automata. To each island $x=(x_i)\in X_1'$ we assign a number equal to

$$x_0+\sum_{k=1}^{\infty}(2^{2k-1}x_k+2^{2k}x_{-k}).$$

We number all binary automata in order of increasing radius r; for equal r we arrange them in increasing order of the number whose binary notation is provided by the table that specifies the function $f(a_{-r},\ldots,a_r)$. Noncomputability in the formulations of both propositions means that the corresponding predicate functions are nonrecursive in the numbering systems we have chosen.

Propositions 1 and 2 are in contrast with the results of [1]; these results essentially assert that certain properties of monotonic binary automata that are also linked to the notion of washing out islands are computable. Another computability result, also related to washout of islands, is given in [2]. A variant of Proposition 1 for nonmonotonic operators is given in Sec. 2 of this paper. The following basic lemma is decisive in the proof.

BASIC LEMMA. There exists a computable mapping $\varphi(T)$ that carries every Turing machine T into a monotonic binary automaton, and a computable mapping $\Psi_T(\pi)$ that carries every program π for T into island X_1' (and depending on T as a parameter) such that automaton $\varphi(T)$ washes out island $\Psi_T(\pi)$ if and only if machine T is applicable to program π. The empty program is carried into island (4) for all T. Computability of both mappings is understood as recursiveness of the functions in our numbering systems for automata and islands and the standard Godel numbering for Turing machines and their programs.

Before deriving the basic lemma, let us derive our propositions from it. Let us prove Proposition 1. Given the existence of an algorithm that determines, on the basis of any monotonic binary automaton P, whether P washes out island (4), then when applied to $\varphi(T)$ this algorithm would determine whether an arbitrary Turing machine T is applicable to an empty program, something that is impossible (see, e.g., [3]).

Let us prove Proposition 2. Let T_0 be a universal Turing machine. Then $P_0=\varphi(T_0)$ is the desired automaton. Indeed, given an algorithm that would determine, on the basis of any island x, whether T_0 washes out island x, this algorithm when applied to $\Psi_{T_0}(\pi)$ would determine whether T_0 is applicable to an arbitrary program, π, something that is impossible.

§2. PROOF OF BASIC LEMMA

We will construct mappings $\varphi(T)$ and $\Psi_T(\pi)$ in five stages, each in the form of a composition of five

mappings. Each of these ten mappings is computable (we have in mind certain specific numberings that we will not write out). Computability of all these mappings can be readily judged from the explicit mode of specifying them, and we will not dwell on its proof.

Stage I

LEMMA 1. There exists a computable mapping $\varphi^1(T)$ that carries every Turing machine T into another Turing machine, and a computable mapping $\Psi_T{}^1(\pi)$ that carries every program π for T into a program for $\varphi^1(T)$, such that: a) $\varphi^1(T)$ is applicable to $\Psi_T{}^1(\pi)$ if and only if T is applicable to π; b) if $\varphi^1(T)$ is applicable to $\Psi_T{}^1(\pi)$, then the tape is empty after its operation is over — there is no nonzero symbol on it.

The proof of Lemma 1 is simple, and we will omit it.

Stage II

LEMMA 2. There exists a computable mapping $\varphi^2(T)$ that carries every Turing machine T into a cellular automaton with radius r = 1, and a computable mapping $\Psi_T{}^2(\pi)$ that carries every program π for T into an island in the state on which this automaton acts, such that automaton $\varphi^2(T)$ washes out island $\Psi_T{}^2(\pi)$ if and only if T is applicable to program π and the tape is empty after its operation is over. The empty program is carried by mapping $\Psi_T{}^2$ for all T into an island with one 1:

$$\ldots 0001000\ldots . \tag{5}$$

To prove Lemma 2 it suffices to take the mappings that were described in the proof of Theorem 4 in [4], with the minor modification that an island which represents a finite machine state on an empty tape is carried into an empty island as a result of operator $\varphi^2(T)$.

Stage III

LEMMA 3. There exists a computable mapping $\varphi^3(P)$ that carries every automaton P with radius r = 1 into a binary automaton, and a computable mapping $\Psi P^3(x)$ that carries every island in the space on which P acts into an island in X_1 such that automaton $\varphi^3(P)$ washes out island $\Psi P^3(x)$ if and only if automaton P washes out island x. Here $\Psi P^3(x)$ for all P carries an island of the form (5) into an island of the same form (5).

Proof. Analogous constructs described in [5] are of no use to us, since they do not carry islands into islands. Therefore, we must give a complete description. Assume that automaton P with radius r = 1 acts on space X_n and is specified by the function $f(a_1, a_2, a_3)$, $f: S_n{}^3 \to S_n$. Let us first describe the mapping $\Psi P^3(x)$. We associate with each symbol $a \in S_n$ its code $C(a)$, a string of zeros and ones of length n + 3, in accordance with the following rule:

$$\begin{aligned} C(0) &= \underbrace{0\ldots0}_{n+3 \text{ zeros}},\ C(1) = \underbrace{0\ldots01}_{n+2 \text{ zeros}}, \\ C(k) &= 110\ldots0,\quad 2 \leqslant k \leqslant n \end{aligned} \tag{6}$$

at the (k — 1)-th position of the n — 1 positions encompassed by the braces, we insert a 1 for C(k), $2 \le k \le n$; zeros are inserted in the remaining places.

Mapping $\Psi P^3(x)$ associates with each island $x = (x_i)$ an island that is the sequence that results if we successively write the code ..., $C(x_{-1})$, $C(x_0)$, $C(x_1)$, the first position of code $C(x_0)$ being placed in the zero position.

Let us now define automaton $\varphi^3(P)$. It is specified by the radius R = 3n + 8 and Boolean function $F(a_{-R}, \ldots, a_R)$, defined by the following condition (Condition A).

Condition A. Assume that there exists a k, $0 \le k \le n + 2$ such that sequence $a_{-R+k}, \ldots, a_{R-(n+2)+k}$ consists of five successively written codes $C(b_1)$, $C(b_2)$, $C(b_3)$, $C(b_4)$, $C(b_5)$, where $b_1, b_2, b_3, b_4, b_5 \in S_n$. Then $F(a_{-R}, \ldots, a_R)$ should be equal to the (n + 3 — k)-th position of code $C(f(b_2, b_3, b_4))$. The value of the function $F(a_{-R}, \ldots, a_R)$ is not important in the cases not accounted for by this condition, and we will set it equal to zero.

Remark 1. Assume that two successively written codes $C(b_1)$, $C(b_2)$ form a sequence $a_1, \ldots, a_{2n+6}$, and assume that there exists a k such that $2 \le k \le n + 3$ and that $a_k, \ldots, a_{k+n+2}$ form code $C(b_3)$. Then $b_3 = 0$. This can be readily proved merely by using (6).

TABLE 1

b_{-1}	b_0	b_1	$F_3(b_{-1}, b_0, b_1)$
Arbitrary	1	Arbitrary	1
Not 1	∅	Not 1	∅
1	∅	∅	)
∅	∅	1	(
He ∅	0	Not ∅	0
∅	0	Not ∅	(
He ∅	0	∅	)
∅	0	∅	∅

COROLLARY. Condition (A) is noncontradictory.

Thus we have specified mapping $\varphi^3(P)$. Using condition (A), we can readily prove the assertion of Lemma 3.

In effect we have already proved analogs of Propositions 1 and 2, but for arbitrary binary automata rather than for monotonic ones. We will formulate one of these.

Proposition 1'. Determination of whether an arbitrary binary cellular automaton washes out island (5) is not computable.

Remark 2. Determination of whether any monotonic binary cellular automaton P washes out island (5) is computable. Indeed, if island P of (5) is nonempty, then (monotonic) P does not wash out (5). Thus, taking account of our subsequent constructs, we obtain that the minimum number of ones in island x^0 for which Proposition 1 is valid is two.

Stage IV

Mapping $\varphi^4(P)$ carries any binary automaton into an automaton of special form (convenient for subsequent stimulation by a monotonic binary automaton). Automaton $\varphi^4(P)$ has a parameter n equal to 5 for all P. For greater clarity, we will denote the six states 0, 1, 2, 3, 4, 5, respectively, as follows:

$$\varnothing, E, 0, 1, (,). \tag{7}$$

Mapping $\Psi^4(x)$ carries every island $x \in X_1'$ into an island in X_5' in accordance with the following rule:

a) if x = ... 000 ..., then $\Psi^4(x)$ = ... ∅∅∅ ...,

b) if island x contains one 1, x = ... 00100 ..., then $\Psi^4(x)$ = ... ∅∅E∅∅ ..., where symbol E is in the same position in which there is a 1 in x,

c) if island x contains at least two ones, then island $\Psi^4(x)$ is obtained from x by successively employing the following operations:

1) replace all ones in x by 1,

2) replace all zeros between ones by 0,

3) replace the rightmost zero such that there are no ones to the left of it by (,

4) replace the leftmost zero such that there are no ones to the right of it by),

5) replace the remaining zeros by ∅.

Example. If x = ... 0001001 0001 000 ..., then $\Psi^4(x)$ = ... ∅∅ (10010001) ∅∅ ....

Mapping $\Psi^4(x)$ has been described. Now let us describe mapping $\varphi^4(P)$. Assume that automaton P is specified by radius r and function $f(a_{-r}, \ldots, a_r)$. Then automaton $\varphi^4(P)$ is specified by radius 2r + 1 and function $F(b_{-(2r+1)}, \ldots, b_{2r+1})$, which we describe as follows. First we will say that automaton $\varphi^4(P)$ is underdefined, i.e., the function $F(b_{-(2r+1)}, \ldots, b_{2r+1})$ is not defined on all groups of argument values.

Specifically, $F(b_{-(2r+1)}, \ldots, b_{2r+1})$ is defined on those and only those groups $b_{-(2r+1)}, \ldots, b_{2r+1}$ for which the definitions can be supplemented up to a complete island $b = (b_i)$, $i \in Z$ that can be represented as $b = \tau^k \Psi^4(x)$, where $x \in X_1'$, and τ^k is a shift operator denoting shift along a straight line k units to the right. It is easy to write this condition for group $b_{-(2r+1)}, \ldots, b_{2r+1}$ in explicit form.

Remark 3. Let us convert group (7) into a partially ordered set, in accordance with the following rules: $\emptyset < b$ and $b < b$ for all b belonging to (7). All the remaining pairs of elements (7) are noncomparable.

Then if function F is defined on two different groups $b_{-(2r+1)}$, b_{2r+1} and $b'_{-(2r+1)}$, ..., b'_{2r+1} and $b_k < b_k'$ for all k from $-(2r+1)$ to $2r+1$, then we have $b_{-(2r+1)} = \ldots = b_{2r+1} = \emptyset$.

To define the automaton $\varphi^4(P)$, we introduce the three auxiliary operators Q_1, Q_2, Q_3 that act on X_5'. We introduce a mapping $g: S_5 \to S_1$ of the form

$$g(E)=g(1)=1, \quad g(\varnothing)=g(0)=g(\text{(})=g(\text{)})=0.$$

Operator Q_1 with radius r is specified by the following function $F_1(b_{-r}, \ldots, b_r)$:

$$F_1(b_{-r},\ldots,b_r)=\begin{cases}1, & \text{if } f(g(b_{-r}),\ldots,g(b_r))=1,\\ 0, & \text{if } f(g(b_{-r}),\ldots,g(b_r))=0,\\ & \text{and } b_0 \text{ is equal to 0 or 1,}\\ \varnothing & \text{otherwise.}\end{cases}$$

Operator Q_2 with radius r is specified by the following function $F_2(b_{-r}, \ldots, b_r)$:

$$F_2(b_{-r},\ldots,b_r)=\begin{cases}0, & \text{if } b_0=\varnothing \text{ and there exist}\\ & k, l \text{ such that } -r\leqslant k<0<l\leqslant r\\ & \text{and } b_k\neq\varnothing \text{ and } b_l\neq\varnothing,\\ b_0 & \text{otherwise.}\end{cases}$$

Operator Q_3 with radius 1 is specified by the function $F_3(b_{-1}, b_0, b_1)$ (see the table).

Function F_3 is not defined in the cases not given in the table.

Let us define the function $F(b_{-(2r+1)}, \ldots, b_{2r+1})$ that specifies automaton $\varphi^4(P)$ as follows:

a) if group $b_{-(2r+1)}, \ldots, b_{2r+1}$ can be represented as part of an infinite group [an island $b = (b_i)$ of the form $b = \tau^k\psi^4(x)$], where the number of ones in x is not equal to 1, then $F(b_{-(2r+1)}, \ldots, b_{2r+1})$ coincides with the function that specifies the automaton-operator $Q_3Q_2Q_1$ that is the composition of the three operators described above;

b) if group $b_{-(2r+1)}, \ldots, b_{2r+1}$ has the form "$b_k = E$, all other b being equal to $\emptyset$ for all $l \neq k$," then the value of $F(b_{-(2r+1)}, \ldots, b_{2r+1})$ coincides with that of $F(b'_{-(2r+1)}, \ldots, b'_{2r+1})$, where $b_k' = 1$, $b'_{k-1} =$ ((if $k > -(2r+1)$), $b'_{k+1} =$) (if $k < 2r+1$); the other b_l are equal to $\emptyset$;

c) in the remaining cases, $F(b_{-(2r+1)}, \ldots, b_{2r+1})$ is not defined.

Definition 4. Assume that P is an underdefined automaton that acts on X_n. We will say that P is applicable to $x = (x_i) \in X_n$ if the function $f(a_{-r}, \ldots, a_r)$ that specifies it is defined on all groups $x_{i-r}, \ldots, x_{i+r}$, where $i \in Z$. If P is applicable to x, then we define Px by formula (1). We will say that P is infinitely applicable to $x \in X_n$ if P is applicable to $P^t x$ for all $t = 0, 1, 2, \ldots$.

LEMMA 4. Automaton $\varphi^4(P)$ is always infinitely applicable to island $\Psi^4(x)$. Automaton $\varphi^4(P)$ washes out island $\Psi^4(x)$ if and only if P washes out x.

The proof is based on the fact that for all $t > 0$

$$\{[\varphi^4(P)]^t\Psi^4(x)\}_i=1 \Leftrightarrow (P^t x)_i=1$$

(the E symbol is impossible for $t > 0$).

If there is not a single 1 symbol on nonempty island $[\varphi^4(P)]^t\psi^4(x)$, where $t > 0$, then the number of positions not occupied by $\emptyset$ decreases in each cycle, and the island will become empty.

Stage V

LEMMA 5. There exists a computable mapping $\varphi^5(P)$ that carries every automaton P from the domain of values of mapping φ^4 into a monotonic binary automaton, and a computable mapping $\Psi^5(x)$ that carries every island x from the domain of values of mapping Ψ^4 into an island in X_1 such that $\varphi^5(P)$ washes out $\Psi^5(x)$ if and only if P washes out x. Here mapping Ψ^5 carries an island of the form (5) into an island of the form (4).

Proof. First let us define mapping $\Psi^5(x)$. For this we assign all of the symbols from (7) the following codes of length 11 consisting of zeros and ones:

$$\begin{cases} C(\varnothing)=00000000000, \\ C(E)=11000000000, \\ C(0)=10100100000, \\ C(1)=10100010000, \\ C(\text{(})=10100001000, \\ C(\text{)})=10100000100. \end{cases} \tag{8}$$

Mapping $\Psi^5(x)$ carries every island $x = (x_i) \in \Psi^4(X_1')$ into the sequence that is obtained if we write out the codes ..., $C(x_{-1})$, $C(x_0)$, $C(x_1)$, ..., successively, the first position of code $C(x_0)$ occupying the zero position.

Now let us define mapping $\varphi^5(P)$. Assume that operator P is characterized by radius r and function $f(a_{-r}, \ldots, a_r)$. Then operator $\varphi^5(P)$ is specified by radius $R = 11r + 21$ and function $F(b_{-R}, \ldots, b_R)$, defined as follows. First let us impose on it the following constraint, which we will call Condition B.

Condition B. If there exists a k, $0 \leq k \leq 10$ such that the sequence b_{-R+k}, ..., b_{R-10+k} constitutes $2r + 3$ successively written codes $C(a_{-(r+1)})$, ..., $C(a_{r+1})$, where $a_{-(r+1)}$, ..., a_{r+1} belongs to group (7) and $f(a_{-r}, \ldots, a_r)$ is defined, then the value of $F(b_{-R}, \ldots, b_R)$ should be equal to the $(k + 1)$-th position of code $C(f(a_{-r}, \ldots, a_r))$.

Remark 4. Assume that the sequence b_1, ..., b_{22} consisting of zeros and ones constitutes two successively written codes $C(a_1)$, $C(a_2)$. Assume that we are given k, where $2 \leq k \leq 11$, and assume that we have chosen b_k', ..., b_{k+10}' such that $b_k' \leq b_k$, ..., $b_{k+10}' \leq b_{k+10}$. Assume, furthermore, that b_k', ..., b_{k+10}' form the code $C(a_3)$. Then $a_3 = \phi$.

COROLLARY 1. Condition B is noncontradictory.

For the proof it suffices to note that if for any group b_{-R}, ..., b_R there exists k_1 and k_2 (two different k values) for which Condition B holds, then the values of a_{-r}, ..., a_r are equal to ϕ in both cases, and therefore $f(a_{-r}, \ldots, a_r)$ is also equal to ϕ in both cases; therefore, the value of $F(b_{-R}, \ldots, b_R)$ required by Condition B is equal to zero in both cases.

COROLLARY 2. Assume that b_{-R}, ..., b_R and b_{-R}', ..., b_R' are two groups consisting of zeros and ones, where

a) $b_{-R} \leq b_{-R}'$, ..., $b_R \leq b_R'$,

b) $F(b_{-R}, \ldots, b_R)$ and $F(b_{-R}', \ldots, b_R')$ are defined by Condition B. Then $F(b_{-R}, \ldots, b_R) \leq F(b_{-R}', \ldots, b_R')$.

Proof. Assume that k and k' are the values of k for which the condition involved in Condition B holds for group b_{-R}, ..., b_R and b_{-R}', ..., b_R', respectively. Let $a_{-(r+1)}$, ..., a_{r+1} and $a_{-(r+1)}'$, ..., a_{r+1}' be the corresponding groups of elements from (7). Let us consider two cases.

1. $k = k'$. Then condition a) and coding rules (8) yield that $a_{-r} \prec a_{-r}'$, ..., $a_r \prec a_r'$. From this, by Remark 3, we have either $a_{-r} = a_{-r}'$, ..., $a_r = a_r'$, or $a_{-r} = \ldots = a_r = \phi$. In both cases the required assertion is obvious.

2. $k \neq k'$. Then $a_{-r} = \ldots = a_r = \phi$ by Remark 4, and this also yields what is required.

Let us now define the function $F(a_{-R}, \ldots, a_R)$ by the following condition.

Condition C. $F(a_{-R}, \ldots, a_R) = 1$ if and only if there exists a group a_{-R}', ..., a_R' such that

a) $a_k' \leq a_k$ for all k from $-R$ to R,

b) $F(a_{-R}', \ldots, a_R')$ is defined in accordance with Condition B and is equal to 1.

Using Corollary 2, we can readily show that function $F(a_{-R}, \ldots, a_R)$ thus defines Condition B. The fact that it is monotonic is obvious.

Operator $\varphi^5(P)$ is specified. The assertion of Lemma 5 can be readily derived from Condition B. Obviously, the basic lemma follows from Lemmas 1-5.

The author wishes to thank A. N. Kolmogorov on whose initiative this study was undertaken.

LITERATURE CITED

1. A. L. Toom, "Monotonic binary cellular automata," Probl. Peredachi Inform., 12, No. 1, 48-54 (1976).
2. G. A. Gal'perin, "Univariate local monotonic operators with memory," Dokl. Akad. Nauk SSSR, 228, No. 2 (1976).

3. M. L. Minsky, Computation and Automata [Russian translation], Mir, Moscow (1971), p. 188.
4. A. R. Smith III, "Simple computation-universal cellular spaces," J. Assoc. Comp. Mach., 18, No. 3, 339-353 (1971).
5. H. Jamada and S. Amoroso, "Structural and behavioral equivalences of tessellation automata," Inf. Control, 18, No. 1, 1-31 (1971).