THE EXISTENCE OF COMPUTABLE SEQUENCE THAT CANNOT BE DESCRIBED BY FINITE AUTOMATA

R. Serikzhan, T. Bakibayev

Al-Farabi Kazakh National University,
480078, Almaty, Kazakhstan

УДК 510.52

The goal of the project is to construct an infinite sequence that cannot be generated by any simple automatic device, and to estimate its complexity. The conjecture on the existence of such a sequence is based on the idea of superiority of Turing machines over finite automata. In the project, a new notion of automaton martingale is introduced, and the existence of an infinite binary random sequence that cannot be generated by a finite automaton is proved. In order to reach the goal of the project one had to study Turing machines, finite automata, computable martingales, and the diagonalization method.

Key words: algorithmic complexity, computable martingales, finite automata.

Introduction. The aim of this project is to construct an infinite random sequence.

We introduce here a new notion of automaton martingale, construct a universal automaton martingale, and an infinite sequence is constructed against it. In order to do this we use the classical diagonalization method.

1. Martingale and Automaton Martingale.

***Definition* 1.** [1] A martingale is a function $d : \{0,1\}^* \to [0,\infty)$ such that $d(\lambda) > 0$, for every $x \in \{0,1\}^*$, the following equality (called fairness condition) holds.

$$\frac{d(x0) + d(x1)}{2} = d(x).$$

$d(\lambda)$ is called the norm of d. d is normed if $d(\lambda) = 1$. A martingale d succeeds on a set A if

$$\limsup_{n \geqslant 0} d(A \upharpoonright n) = \infty.$$

$S^\infty[d]$ denotes the class of sets on which the martingale d succeeds. A martingale d succeeds on a class C if $C \subseteq S^\infty[d]$.

***Definition* 2.** A (betting) strategy s is a function $s : \{0,1\}^* \to [0,1]$. *The strategy* s_d *underlying the martingale* d is the function

$$s_d(x) = \begin{cases} \dfrac{d(x0)}{2d(x)} & \text{if } d(x) \neq 0 \\ 0 & \text{otherwise.} \end{cases} \tag{1}$$

Conversely, for every strategy s and every real $\alpha > 0$, *t*he martingale $d[s,\alpha]$ of norm α induced by s is defined by $d(\lambda) = \alpha$ and, for any string $X \upharpoonright (n+1)$ where $n \geqslant 0$,

$$d(X \upharpoonright (n+1)) = \begin{cases} 2 \cdot s(X \upharpoonright n) \cdot d(X \upharpoonright n) & \text{if} \quad X(n) = 0 \\ 2 \cdot (1 - s(X \upharpoonright n)) \cdot d(X \upharpoonright n) & \text{if} \quad X(n) = 1. \end{cases} \tag{2}$$

We defined a new notion of automaton martingale in this paper in order to define sequences that can be generated by a DFA. Since an automaton martingale is based on a DFA, i. e. it can either accept or reject an expression, we have to consider the bets according to outcome. In case if an automaton accepts the expression (input), the corresponding automaton martingale bets 0.75 (on zero). In case if an automaton rejects the expression, the corresponding automaton martingale bets 0.25.

***Definition* 3.** An *automaton martingale* is a martingale with a strategy that can be described by a deterministic finite automaton.

2. Main Theorem.

***Theorem* 1.** There exists a universal Turing machine that generates an infinite sequence X that cannot be generated by any automaton martingale.

In order to prove theorem 1 we need to prove the following 3 lemmas.

***Lemma* 1.** There exists a universal Turing machine which enumerates all deterministic finite automata. Moreover, by given string of length n Turing machine checks whether it is a coded automaton in linear time.

Proof. Let the following string be a code of an automaton:

$$0^m 10^l 1 x f \delta,$$

where m is the number of states, l is the code length of each state, x is a sequence of all coded states ($|x| = m * l$), f is a binary string of length m where the i-th bit equals 1 if and only if the state number i is final, and $\delta = q_{\langle 0,0\rangle} q_{\langle 1,0\rangle} ... q_{\langle m,1\rangle}$ represents the following transition table ($|\delta| = 2 * m * l$):

Table

Transition table

	q_0	q_1	...	q_m
0	$q_{\langle 0,0\rangle}$	$q_{\langle 1,0\rangle}$	...	$q_{\langle m,0\rangle}$
1	$q_{\langle 0,1\rangle}$	$q_{\langle 1,1\rangle}$	...	$q_{\langle m,1\rangle}$

for transitions of type $q_i, j \rightarrow q_{\langle i,j\rangle}$.

Without lost of generality we assume that the first state is the initial one, because any finite automaton with another initial state can be rewritten as an (isomorphic) automaton with the first state as initial and that accepts the same language.

So, the total length of the code is $n = m+1+l+1+m*l+m+2*m*l = 3ml+2m+l+2$. In order to check whether a given string represents a coded automaton we have to do the following:

1) Check whether the string is of the form $0^m 10^l 1 x f \delta$ (in n steps).

2) Check whether the length of $x f \delta$ equals $3ml + m$ (in $k * m * l$ steps).

Checking whether δ contains only codes from x is not necessary because automata with wrong states will be stopped during simulation once a wrong state is met.

***Lemma* 2.** For any martingale d_1 and automaton martingale d_2 there exists one martingale $d_{1,2}$, that succeeds on all sequences on which any or both martingale and automaton martingale succeed.

Proof. In order to create a new martingale $d_{1,2}$ we emulate s_{d_1} and s_{d_2} from computable set of all automaton martingales on some sequence X. An initial capital is divided into equal parts between both martingales such that each of them can apply its own strategy on the sequence. Each of s_{d_1} and s_{d_2} bets according to its strategy (on zero), and the new martingale strategy $s_{d_{1,2}}$ summarizes the bets and bets the total sum. Whether $d_{1,2}$ wins or lose on current step, it divides the obtained capital between d_1 and d_2 corresponding to their new capitals. Thus, martingales can win or lose on a sequence independently from each other. We should note that the new martingale $d_{1,2}$ succeeds if at least one of the martingales d_1 and d_2 succeeds:

$$\limsup_{n \geqslant 0} d_{1,2}(X \upharpoonright n) = \limsup_{n \geqslant 0} d_1(X \upharpoonright n) + \limsup_{n \geqslant 0} d_2(X \upharpoonright n).$$

Therefore, if one of the martingales succeeds then

$$\limsup_{n \geqslant 0} d_1(X \upharpoonright n) = \infty \text{ or } \limsup_{n \geqslant 0} d_2(X \upharpoonright n) = \infty.$$

It implies that

$$\limsup_{n \geqslant 0} d_{1,2}(X \upharpoonright n) = \infty.$$

Note, that the definition of the new martingale $d_{1,2}$ is as follows:

$$d_{1,2}(x) = d_1(x) + d_2(x),$$

and

$$d_{1,2}(\lambda) = 2d_1(\lambda).$$

***Lemma* 3.** Given a finite set of automaton martingales $\{d_1, d_2, ..., d_n\}$ there exists one combined martingale $d_{1,2,...,n}$ that succeeds on all sequences on which any of these automaton martingales succeeds.

Proof. Lemma is proved by induction. In the proof of lemma 2 we took some fixed d_1 and d_2 and combined them into one martingale. In the same way we can merge a set of other automaton martingales into a combined martingale $d_{1,2,...,n}$. For example, we combine martingale $d_{1,2}$, which is based on d_1 and d_2, with the martingale d_3. As a result, if one of $d_{1,2}$ and d_3 wins, the new martingale $d_{1,2,3}$ based on them also wins. So, we will get the universal martingale $d_{1,2,...,n}$ that succeeds on all sequences on which any automaton martingale succeeds. Thus, the definition of the combined martingale $d_{1,2,...,n}$ is

$$d_{1,2,...,n}(x) = d_1(x) + d_2(x) + d_3(x) + ... + d_n(x)$$

***Definition* 4.** Let z_s be a string number s in length-lexicographical ordering.

As the coding of finite automata and combined martingales $d_{1,2,...,n}$ are defined we can now turn to the construction of the desired set.

Proof (of theorem 1).

It is easy to see that we can simply take one general combined martingale for all automaton martingales and diagonalize over it. But for complexity estimation reasons it is better to have the exact procedure for building the set and hence the exact algorithm of $s_{d_{1,2,...}}$. We build the set step by step as follows.

On step s of the construction we define the value of $X(z_s)$. In order to do this, we do the following (diagonalize against all automaton martingales up to z_s):

1. Check which of the strings $z_{s'}$ with $s' \leqslant s$ are coded automata (by lemma 1).

2. By lemma 3 combine all of the corresponding automaton martingales into one martingale $d_{1,2,\dots,k}$.

3. Run the emulation process of the corresponding strategy $s_{d_{1,2,\dots,k}}$ on string $X(z_0)X(z_1)...X(z_{s-1})$.

4. In case if $s_{d_{1,2,\dots,k}}(X(z_0)X(z_1)...X(z_{s-1})) > 0.5$ (it bets more on zero) let $X(z_s) = 1$, otherwise let $X(z_s) = 0$.

Thus we guarantee that $\lim_{n\geqslant 0}\sup d_{1,2,\dots,k}(X \upharpoonright n) \neq \infty$ for all $k > 0$ and that $\lim_{n\geqslant 0}\sup d_i(X \upharpoonright n) \neq \infty$ for any $i > 0$. Therefore, no automaton martingale succeeds on the constructed set (on the characteristic sequence of the set).

3. Estimation of complexity.

***Lemma* 4.** Given $X \upharpoonright z_s$, all strings $z'_s < z_s$ that represent finite automata,and $d_i(z_{s-1})$ for each such automaton ($0 \leqslant i \leqslant s$) it takes $k \cdot s$ steps to compute $X(z_s)$.

Proof. By the proof of theorem 1 we have to emulate each finite automaton z'_s such that $s' \leqslant s$ on string $X \upharpoonright z_{s-1}$. Since we are given all strings $z'_s \leqslant z_s$ that represent finite automata it suffices to check if z_s is a finite automaton. And by lemma 1 it takes linear time, i. e., less than $k \cdot \log s$ steps.

In order to construct sequence X, that contains some strings (for instance, $z_1 = 0$, $z_2 = 1$, $z_3 = 00$ and so on) we need to start from λ and check whether it represents an automaton. As λ is not a coded automaton(by lemma 1) we add 0 to X, and we do the same with other strings until we find an automaton. As soon as we meet one we need to emulate that automaton on already constructed X in order to find the next member of X. We do the same actions until another automaton is met. In this case we give them the capital equal 1 and emulate each of them on $X \upharpoonright z_s$ in order to find $X(z_s)$. So, each automaton „bets“ according to its corresponding automaton martingale's strategy, and we sum up these bets.

Note that we emulate an automaton on X by emulating its first defined symbol, then the first two symbols and so on in order to let capital of an automaton rise or fall, and we always give automata initial capital 1 irrespectively of the number of automata. So, we do the following: 1) if there are no automata we let $X(z_s) = 0$; 2) emulate the last step of each of these automata one by one; 3) sum up the bet; 4) if we bet more on 0 then let $X(z_s) = 1$ otherwise let $X(z_s) = 0$.

Now we have to check how many strings $s' \leqslant s$ represent coded finite automata. Since the code of an automaton is $0^m10^l1xf\delta$, it is easy to see that there are not more than $2^{|z_s|-2-m-l}$ or $2^{|xf\delta|}$ automata. Hence, there exists some k, such that for the first, third and fourth steps of the above procedure it takes k steps, and for the second step — $k \cdot \dfrac{s}{4}$ steps. Without lost of generality we can assume that it takes $k \cdot s$ steps to complete the whole procedure.

In order to simplify the explanation let a be the capital of the first automaton and b be the capital of the other. Let $s(d_{A_1}) = 0.7$ and $s(d_{A_2}) = 0.25$ then the total bet is $s(d_{A_{1,2}}) = \dfrac{0.7a + 0.25b}{a+b}$. Thus, we combine two automata into one on a fixed segment of X, and the same way we combine all the following automata step by step. If $s(d_{A_{1,2}}) > 0.5$ we let $X(z_s) = 1$ for the diagonalization, otherwise we let $X(z_s) = 0$.

***Theorem* 2.** Given z_s it takes $2^{2|z_s|}$ steps to compute $X(z_s)$.

Proof. By lemma 4 given $X \upharpoonright z_s$ and all strings $s' < s$ that represent finite automata, it takes $k \cdot s$ steps to compute $X(z_s)$. So, by induction it takes $s \cdot k \cdot s$ steps to compute $X(z_s)$. Since $s \leqslant 2^{|z_s|}$, the total number of steps is less than $2^{|z_s|} \cdot 2^{|z_s|} \cdot k \leqslant k \cdot 2^{2|z_s|} \leqslant 0(2^{2|z_s|})$.

References

1. Rodney G. Downey, Denis R. Hirschfeldt. Algorithmic Randomness and Complexity, Theory and Applications of Computability, New York: Springer, 2010.

Raushan Serikzhan —
SU Manager al-Farabi Kazakh National University;
e-mail: r.serikzhan@gmail.com

Timur Bakibayev —
Ph.D. al-Farabi Kazakh National University;
e-mail: timurbakibayev@gmail.com

Дата поступления — 11.11.2014