The Seventeenth International Symposium on Artificial Life and Robotics 2012 (AROB 17th '12),
B-Con Plaza, Beppu, Oita, Japan, January 19-21, 2012

# Real-Time Generation of Prime Sequence by One-Dimensional Cellular Automaton with 8 States

K. Miyamoto [1], H. Umeo [2]

[1] Japan Advanced Institute of Science and Technology, Ishikawa 923-1292, Japan
[2] Univ. of Osaka Electro-Communication, Osaka 572-8530, Japan
(Tel: 0761-51-1111)
(k-miyamoto@jaist.ac.jp)

**Abstract:** In the present paper, we study a prime sequence generation problem on one-dimensional cellular automata. We show that an infinite prime sequence can be generated in real-time by a one-dimensional cellular automaton with 8 states. The algorithm that we propose is based on the well-known sieve of Eratosthenes, and its implementation is realized on an 8-state cellular automaton using 301 transition rules.

**Keywords:** one-dimensional cellular automaton, real-time prime generation algorithm, sieve of Eratosthenes

## 1 INTRODUCTION

Cellular automaton is considered to be a good model of complex systems in which an infinite one-dimensional array of finite state machines (cells) updates itself in a synchronous manner according to a uniform local rule. In the present paper, we study a prime sequence generation problem on one-dimensional cellular automata. The sequence generation problem has been studied for many years and a variety of interesting sequences has been shown to be generated in real-time by one-dimensional cellular automata [1-9].

Arisawa [1971], Fischer [1965], Korec [1997, 1998] and Mazoyer and Terrier [1999] have considered the sequence generation problem on the cellular automata model. Umeo and Kamikawa [2002] showed that infinite non-regular sequences such as $\{2^n | n = 1, 2, 3, ..\}$, $\{n^2 | n = 1, 2, 3, ..\}$ and Fibonacci sequences can be generated in real-time and the prime sequence in twice real-time by $\mathrm{CA}_{1-\mathrm{bit}}$. Umeo and Kamikawa [2003] showed that the prime sequence can be generated in real-time by a $\mathrm{CA}_{1-\mathrm{bit}}$ having 34 internal states and 107 transition rules. The cellular automaton model $\mathrm{CA}_{1-\mathrm{bit}}$ is a special subclass of *conventional* (i.e., *constant-bit-communication*) cellular automata. Each cell having the 1-bit inter-cell communication link can communicate with its nearest neighbor cells by sending and receiving a 1-bit information at each step.

Korec [1997] has shown that the prime sequence generation problem can be solved in real-time by using sieve of Eratosthenes. He also presented a one-dimensional cellular automaton with 11 states which can generate the prime sequence. Later, Korec [1998] gave a one-dimensional cellular automaton with 9 states generating the prime sequence in real-time.

In this paper we improve the implementation by presenting a smaller implementation on the same computational model. We show that an infinite prime sequence can be generated in real-time by a one-dimensional cellular automaton with 8 states.

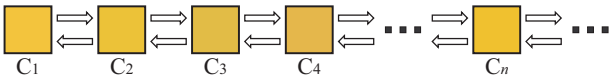


Fig. 1. One-dimensional cellular automaton.

First, in Section 2 we introduce a one-dimensional cellular automaton and define the sequence generation problem on the cellular automaton. In Section 3, we give a real-time prime sequence generation algorithm on a one-dimensional cellular automaton. The algorithm is based on the well-known sieve of Eratosthenes, and its implementation will be made on a one-dimensional cellular automaton with 8 states. The number of states implemented is the smallest one, known at present.

## 2 SEQUENCE GENERATION PROBLEM ON ONE-DIMENSIONAL CELLULAR AUTOMATON

A one-dimensional constant-bit-communication cellular automaton consists of an infinite array of identical finite state automata, each located at a positive integer point. See Fig. 1. Each automaton is referred to as a cell. A cell at point $i$ is denoted by $\mathrm{C}_i$, where $i \geq 1$. Each $\mathrm{C}_i$, except for $\mathrm{C}_1$, is connected to its left- and right-neighbor cells by a communication link.

A cellular automaton $\mathcal{A} = (\mathcal{Q}, \delta, \mathcal{F})$, where

1. $\mathcal{Q}$ is a finite set of internal states.
2. $\delta : \mathcal{Q}^3 \rightarrow \mathcal{Q}$ is a transition function. A quiescent state $q \in \mathcal{Q}$ has a property such that $\delta(q, q, q) = q$.
3. $\mathcal{F}$ is a special subset of internal states $\mathcal{Q}$.

The set $\mathcal{F}$ is used to specify a designated state of $C_1$ in the definition of the sequence generation.

We now define the sequence generation problem on the cellular automaton. Let $M$ be a cellular automaton and $\{t_n | \, n = 1, 2, 3, ...\}$ be an infinite monotonically increasing positive integer sequence defined on natural numbers such that $t_n \geq n$ for any $n \geq 1$. We then have a semi-infinite array of cells, as shown in Fig. 1, and all cells, except for $C_1$, are in the quiescent state at time $t = 0$. The communication cell $C_1$ assumes a special state "0" (zero) in $Q$ and starts its operation at time $t = 0$ for initiation of the sequence generator. We say that $M$ generates a sequence $\{t_n | \, n = 1, 2, 3, ...\}$ in $k$ *linear-time* if and only if the leftmost end cell of $M$ falls into a special state "1" (one) in $Q$ at time $t = kt_n$, where $k$ is a positive integer. We call $M$ a *real-time* generator when $k = 1$.

In designing algorithms for a cellular automaton, for ease of understanding and description of algorithms on cellular automata, we often use *signals* or *waves*, instead of transition tables. A signal (wave) is an information flow that is described as a straight line in the space-time diagram. Note that any signal cannot propagate at speed more than one cell per one step due to the definition of the transition function.

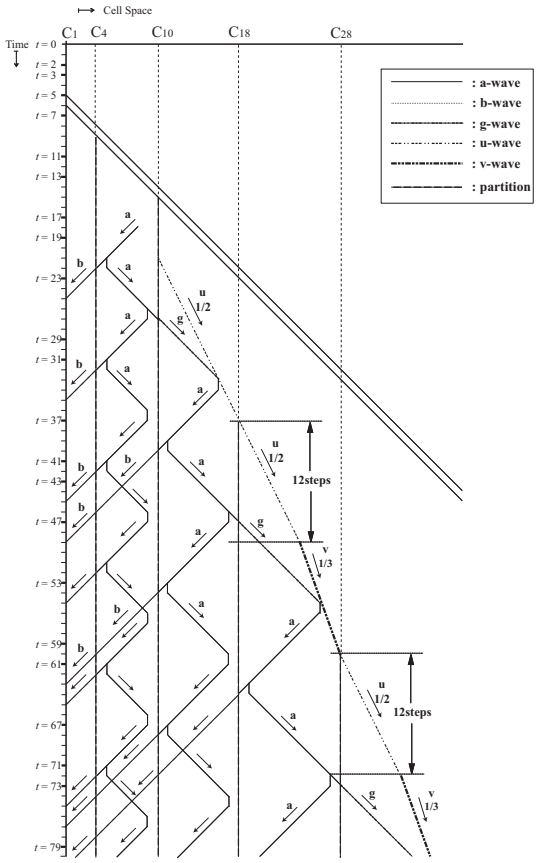


Fig. 2. Space-time diagram for real-time generation of prime sequence.

## 3 REAL-TIME GENERATION OF PRIME SEQUENCE ALGORITHM

In this section, we present a real-time generation algorithm for prime sequence on cellular automaton. The algorithm is implemented on an 8 state cellular automaton using 301 transition rules. Our prime generation algorithm is based on the classical sieve of Eratosthenes. Its cross out technique in the sieve of Eratosthenes is as follows. Imagine a list of all integers greater than 2. The first member, 2, becomes a prime and every second member of the list is crossed out. Then, the next member of the remainder of the list, 3, is a prime and every third member is crossed out. In Eratosthenes' sieve, the procedure continues with 5, 7, 11, and so on. In our procedure, given below, for any odd integer $k \geq 3$, every 2$k$-th member of the list beginning with $k^2$ will be crossed out, since the $k$-th members less than $k^2$ (that is, $\{i \cdot k | \, 2 \leq i \leq k - 1\}$) and 2$k$-th members beginning with $k^2 + k$ (that is, it is an even number such that $\{(k+2i-1) \cdot k | i = 1, 2, 3, ...\}$) should have been crossed out in the previous stages. Thus, every $k$-th member beginning with $k^2$ is successfully crossed out in our procedure. Those integers never being crossed out are the primes. Figure 2 is a space-time diagram that shows a real-time detection of odd multiples of odds.

Our implementation that $C_1$ translate for state 0 when b-wave arriving $C_1$ and $C_1$ translate for state 1 when b-wave don't arriving $C_1$. The b-wave is generated by three ways. First, b-wave is generated when $C_2$ translate for state L. Second, b-wave is generated when $C_3$ translate for state L. Finally, b-wave is generated when a-wave collides with left partition.

### 3.1 Removing multiples of 2 and 3

In this section, we present a cross-out operation for multiples of 2 and 3. In the case of multiples of 2, the array repeats two states: R and L, alternatively on $C_3$. As for the case of multiples of 3, it repeats a state R, V and L on $C_2$ at each step. The cells $C_2$ and $C_3$ generate a b-wave, then they fall into the state L. A state transition into the state L on $C_2$ is done at time $t = 3k - 1 (k \geq 2)$ and the state transition into L on $C_3$ is done at time $t = 2k (k \geq 2)$. In this way the multiples of 2 and 3 can be removed.

We implemente Removing multiples of 2 and 3 algorithm.

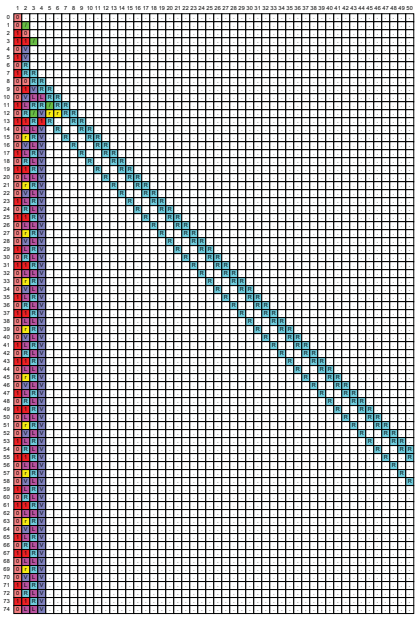

Fig. 3. Implementation of removing multiples of 2 and 3.

See the Figure 3 for implementation of removing multiples of 2 and 3. This figure is first partition generation on $C_4$, but this partition isn't need removing multiples of 2 and 3 algorithm. We generate first partition that it is easy to look.

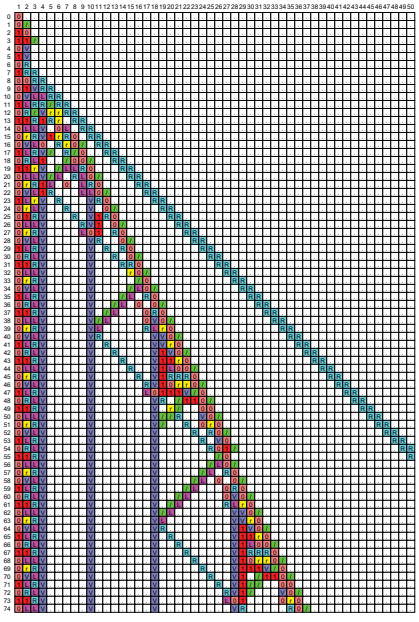

Fig. 4. Generation of partition.

### 3.2 Removing algorithm for odd multiple numbers greater than 5

In this section, we present a removing algorithm for odd multiples greater than 5. Those cross-out operations are done by a signal which moves around between partitions generated by the cellular automaton. The CA generates partitions, denoted by the state V on $C_{i^2+3i}(i \geq 1)$. We define the subspace $S_i, i \geq 1, i \in \mathcal{N}$, as the $i$th cellular space delimited by the consecutive partitions. The length of the space of $S_i$ is $|S_i| = 2i + 3$. The subspace $S_i, i \geq 1$ is used to generate a signal for removing odd multiples greater than 5, that is, $(2i + 3)k, i \geq 1, k \geq 1$. The a-wave goes left and right between those partitions at a unit speed, one cell per one step. It changes into the b-wave, then it collides with the left partition. Those odd multiples greater than 5 are crossed out by the b-wave that goes through the partition in the left direction. The first partition is generated on $C_4$. See the Figure 4 for the real-time generation of those partitions.

We must generate the a-wave in each $S_i$. The a-wave generation in each partition is as follows: Thus the a-wave in $S_2$ is generated at $t = 34$ on $C_{16}$ and the a-wave in $S_3$ is generated at $t = 56$ on $C_{26}$. The a-wave in $S_1$ is specially described in terms of finite states. At $t = 34$ the cell $C_{16}$ generates a-wave which arrives on $C_1$ at $t = 49(= 7^2)$, and the a-wave generated on $C_{26}$ at $t = 56$ arrives on $C_1$ at $t = 81(= 9^2)$. We use these a-waves to remove odd multiples greater than 5. The implementation of the generation of partitions can be done shown in Figure 4.

### 3.3 Real-time generation of prime sequence

In this section, we present a real-time generation of prime sequence algorithm. The algorithm is combination by "Removing multiples of 2 and 3" and "Removing algorithm for odd multiple numbers greater than 5". Those algorithms are implemented follows: section 3.1 and 3.2. Space-time diagram is shown in Figure 2. Figure 5 shows the transition table consisting of the 8 states and 301 transition rules where state '*' is a wall state of the left neighbor of $C_1$. Figure 6 shows some snapshots of the prime generation on 50 cells. Now we have:

**Theorem** There exists a cellular automaton having 8-states and 301-rules that can generate prime sequence in real-time.

## 4 CONCLUSIONS

We have studied a real-time prime sequence generation problem on a cellular automaton. The proposed real-time prime generator, based on the classical sieve of Eratosthenes, is implemented on a cellular automaton using 8 states and 301 transition rules. The implementation is the smallest one known at present.

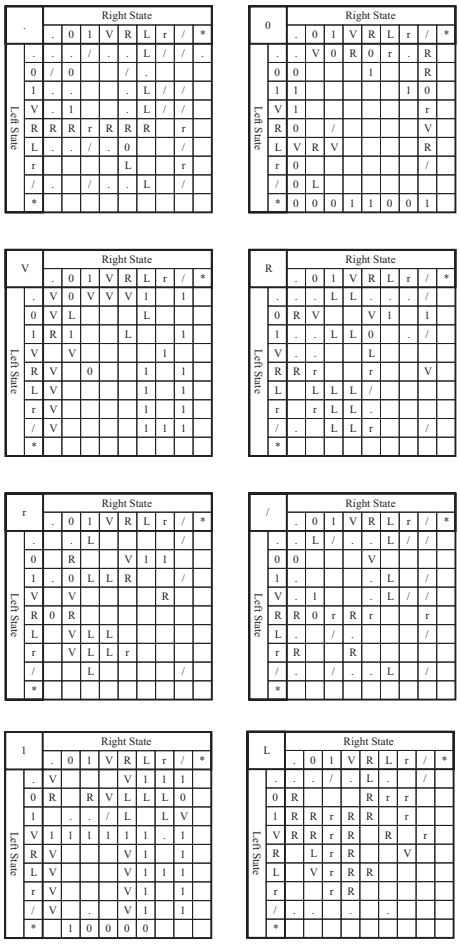


Fig. 5. Transition table for the 8-state real-time prime generator.

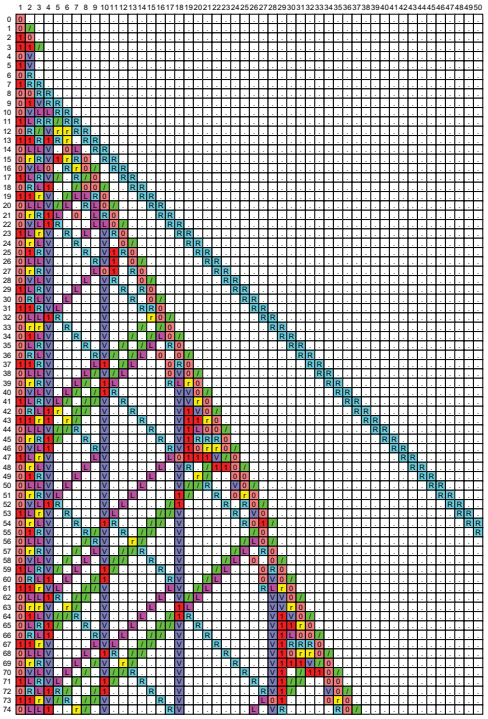

Fig. 6. Snapshots for the real-time generation of prime sequence on 50 cells.

## ACKNOWLEDGEMENTS

A part of this work of the second author is supported by Grant-in-Aid for Scientific Research (C) 21500023.

## REFERENCES

[1] M. Arisawa: On the generation of integer series by the one-dimensional iterative arrays of finite state machines (in Japanese). *Trans. of IECE*, 71/8 Vol.54-C, No.8, pp.759-766, (1971).

[2] C. R. Dyer: One-way bounded cellular automata. *Information and Control*, Vol.44, pp.261-281, (1980).

[3] P. C. Fischer: Generation of primes by a one-dimensional real-time iterative array. *J. of ACM*, Vol.12, No.3, pp.388-394, (1965).

[4] I. Korec: Real-time generation of primes by a one-dimensional cellular automaton with 11 states. *Proc. of 22nd Intern. Symp. on MFCS '97*, *LNCS*, 1295, pp.358-367, (1997).

[5] I. Korec: Real-time generation of primes by a one-dimensional cellular automaton with 9 states. *Proc. of MCU'98*, pp.101-116, (1998).

[6] J. Mazoyer and V. Terrier: Signals in one-dimensional cellular automata. *Theoretical Computer Science*, 217, pp.53-80, (1999).

[7] A. R. Smith: Real-time language recognition by one-dimensional cellular automata. *J. of Computer and System Sciences*, 6, pp.233-253, (1972).

[8] H. Umeo and N. Kamikawa: A design of real-time non-regular sequence generation algorithms and their implementations on cellular automata with 1-bit inter-cell communications. *Fundamenta Informaticae*, 52, No.1-3, pp.255-273, (2002).

[9] H. Umeo and N. Kamikawa: Real-time generation of primes by a 1-bit-communication cellular automaton. *Fundamenta Informaticae*, 58(3, 4)421-435, (2003).