A Simple Model of Asynchronous Cellular Automata Exploiting Fluctuation

Article *in* Journal of cellular automata · January 2011

Source: DBLP

CITATIONS
7

READS
42

1 author:

Jia Lee
Chongqing University
70 PUBLICATIONS **694** CITATIONS

SEE PROFILE

A Simple Model of Asynchronous Cellular Automata Exploiting Fluctuation

JIA LEE[1,2⋆]

[1] *College of Computer Science, Chongqing University, Chonqqing, China*
[2] *Nano ICT Group, National Institute of Information and Communications Technology, Kobe, Japan*

Asynchronous cellular automata (ACAs) allow cells to undergo state transitions independently at random times. Computation in ACAs usually requires some special mechanism to control the unpredictable transitions of cells, which in turn causes the increase in the complexity of ACAs as compared to their synchronous counterpart. Nevertheless, inclusion of fluctuation into ACAs promises models with less complexity, e.g., the Brownian cellular automaton (Lee and Peper, 2008) which uses three cell states and three kinds of transition rules to conduct universal computation. By taking this approach one step further, we propose a novel cellular automaton in which local configurations representing signals will propagate randomly in the cellular space, as if they were subject to Brownian motion. Depending on merely two cell states and two kinds of transition rules, our new ACA can be used to construct any arbitrary asynchronous circuit in the cellular space, and hence, it is computationally universal. The reduced complexity of the new ACA may offer possibility of simple physical realizations.

Key words: cellular automaton, Brownian motion, computational complexity, universal computation, asynchronous circuit

⋆ email: lijia@cqu.edu.cn

1 INTRODUCTION

A *cellular automaton* (CA) is a discrete dynamical system consisting of an d-dimensional array of identical finite-state automata (cells) ($d \geq 1$). Each cell is connected uniformly to a neighborhood of a finite number of cells, and has a state from a finite state set. It updates its state according to a local transition function, which determines a cell's state based on the states of the cells in its neighborhood. Most CA models require all cells undergo state transitions simultaneously, so they are called synchronous cellular automata. As a generalized model of CA, asynchronous cellular automata (ACAs) allow cells to update their states independently at random times.

Because of the randomness in the update order of cells, computing on ACAs are usually accomplished by simulating a timing mechanism to force all cells into synchronicity [15, 9], the dark side of the coin is the rapid increase in the numbers of cell states as well as transition rules. More effective approach is based on the embedding of delay-insensitive (DI) circuits directly into asynchronous cellular spaces [8, 11, 18, 19]. A DI-circuit is a special type of asynchronous circuits whose function is robust to arbitrary delays involved in wires or operators (e.g. [5]). For example, the so-called *self-timed cellular automaton* (STCA) model [19] uses six kinds of transition rules and is able to construct any arbitrary DI-circuit. An STCA can be regarded as a block cellular automaton (e.g. [14]) in which a transition rule not only accesses the neighbors of a cell, but also updates their states simultaneously.

What is the minimal complexity of asynchronous cellular automata capable of universal computation? Models with the least complexity achieved thus far include the *Brownian cellular automaton* (BCA) [12]. The BCA is a 2-dimensional array of cells, each of which accesses its four adjacent non-diagonal neighboring cells along with itself, and update their states at the same time. That is, the BCA is a kind of block cellular automata with von Neumann neighborhood. In addition, local configurations representing signals may propagate randomly fluctuating between forward and backward directions in the cellular space, as if they were subject to Brownian motion. Inclusion of fluctuation in this way results in much less complexity of the BCA (also see [21]) as compared to other comparable ACA models. The BCA requires only three states per cell and three kinds of transition rules. Such particularly low complexity is due to the implementation of circuit elements that can exploit the fluctuations of signals. Called Brownian circuit (BC) elements [13, 12], these elements can be used to realize any arbitrary DI-circuit, i.e., they form a universal set for DI-circuits.

To further reduce the numbers of cell states as well as transition rules, this paper proposes a novel cellular automaton with Moore neighborhood. We start by demonstrating that our new ACA can employ two cell states and three kinds of transition rules to implement all BC-elements as well as Brownian signals. The decrease in the number of cell states from the BCA is due not only to the use of a larger neighborhood of cells, but also to a somewhat new scheme for designing transition rules.

Witness the fixed number as well as the functionalities of the BC-elements (see Section 2), however, making a reduction in the number of transition rules seems much more difficult than in the number of cell states. Fortunately, previous constructions of DI-circuits, as we observe, do not make full use of the entire functionalities of each BC-element. Thus, for the sake of realizing the whole set of DI-circuits, our observation offers opportunity to ignore those redundant functions of each element during their implementations. The decrease in the complexities of BC-elements eventually leads to a further reduction in the number of transition rules. As a result, our new ACA requires merely two cell states and two transition rules and still can be used to construct any arbitrary DI-circuit.

This paper is organized as follows: Section 2 gives a brief overview of delay-insensitive circuits and Brownian circuit elements. More details can be found in [7, 10, 16, 22] and [13, 12, 20], respectively. Section 3 presents our new cellular automaton and shows that it can implement the BC-elements. A further reduction in the number of the transition rules of our ACA is described in Section 4, followed by the conclusions given in Section 5.

2 DELAY-INSENSITIVE CIRCUITS AND BROWNIAN CIRCUIT ELEMENTS

A *delay-insensitive circuit* is an asynchronous circuit, whose correct function is not affected by arbitrary delays involved in interconnection lines or elements. Communications between the circuit and the outside world are done via exchanging tokens(signals) through the input and output lines. A DI-circuit is called *conservative* if it conserves the number of tokens between inputs and outputs [17]. Like conventional synchronously-timed circuits, all conservative DI-circuits can be constructed from a fixed set of primitives. Figure 1 shows three kinds of primitive elements that can be used to realize any arbitrary conservative DI-circuit, i.e., they form an universal set for conservative DI-circuits.

A *Brownian circuit* [20, 13, 12] is a special type of conservative DI-circuit,

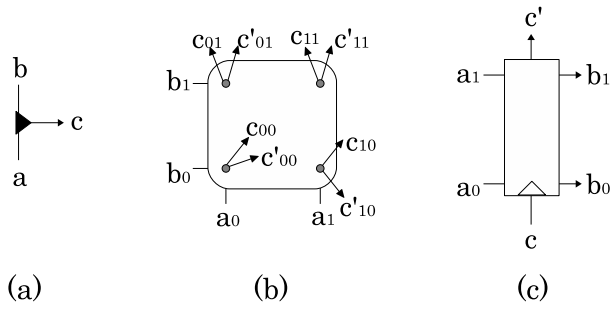

FIGURE 1
Universal set of conservative DI-circuit elements [17]. (a) **Merge**: A signal arriving on input line a or b is transferred to output line c. (b) **Conservative 2×2-Join**: A pair of signals with one arriving on input line a_i and another one arriving on b_j ($i, j \in \{0, 1\}$), is assimilated and results in one signal on each of the output lines c_{ij} and c'_{ij}. (c) **CSequencer** (Conservative Sequencer): An input signal on line a_0 (or a_1) together with an input signal on line c are assimilated, resulting in an output signal on line b_0 (resp. b_1) and a signal on output line c'. If there are input signals on both a_0 and a_1 at the same time as well as an input signal on c , then only one of the signals on a_0 and a_1 (possibly chosen arbitrarily) is assimilated together with the signal on c. The remaining input signal will be processed at a later time, when a new signal is available on line c.

in which the movements of tokens on lines fluctuate randomly between going forward and backward. Due to the Brownian-like behavior, the possible reversal movements of tokens enable the circuit to backtrack from the deadlock states. Deadlocks may take place in token-based asynchronous systems, like the Petri net (e.g. [24]), and usually requires special functionality in the systems to resolve them. Random fluctuations of tokens, however, can provide this functionality as part of their nature, thereby allowing for less complex primitive elements and circuit constructions.

Figure 2 gives three kinds of Brownian circuit elements each of which has much less complex functionality than those elements given in Fig. 1. In addition, Fig. 3 illustrates that the Merge, Conservative 2×2-Join, and CSequencer can be realized by the BC-elements, respectively. That is, {CJoin, Hub, Ratchet} forms a universal set from which any arbitrary conservative DI-circuit can be constructed [13, 12].

In the next section we present a novel model of asynchronous cellular au-

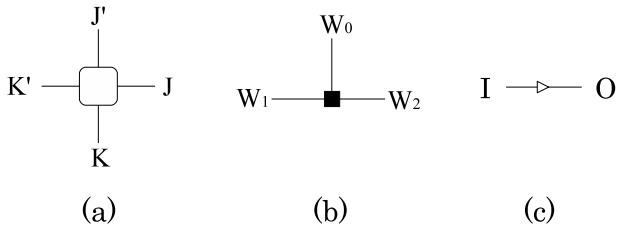

FIGURE 2
Primitive elements of Brownian circuits [13, 12]. (a) **CJoin** (Conservative Join): Two signals with one arriving on line J (or K) and another one on line J' (resp. K') are processed and give rise to one signal on each of the lines K and K' (resp. J and J'). (b) **Hub**: A signal arriving on line W_i will be transferred to one of the other lines W_j, where $i, j \in \{0, 1, 2\}$ and $i \neq j$. (c) **Ratchet**: A signal arriving on line I is transferred to line O. This element works as a diode in the sense that though a signal may fluctuate on lines I and O, after it pass the Ratchet, the signal can not return.

tomata that is capable of universal computation. We achieve the universality by implementing the BC-elements in Fig. 2 as well as signals undergoing Brownian motion in the cellular space, which enables the constructions of any arbitrary DI-circuit.

3 ASYNCHRONOUS CELLULAR AUTOMATON IMPLEMENTING BROWNIAN CIRCUIT ELEMENTS

The cellular automaton model here is a 2-dimensional array of cells each of which takes a state from the state set $\{0,1\}$, denoted by white and black colors, respectively. Each cell undergoes state transitions based on its own state and the eight nearest neighboring cells (Moore neighborhood). The local transition function is described by transition rules given in the form as illustrated in Fig. 4. That is, a transition rule will update not only the state of the central cell, but also the states of all cells in the neighborhood.

In addition, the local transition function satisfies rotation as well as reflection symmetry (isotropy), i.e., for each rule its rotational and reflectional equivalences are also included in the function (see Fig. 4). Figure 5 lists three kinds of transition rules: R_1, R_2, R_3 that are used to update the states of cells of our new cellular automaton. Moreover, cells are updated asynchronously

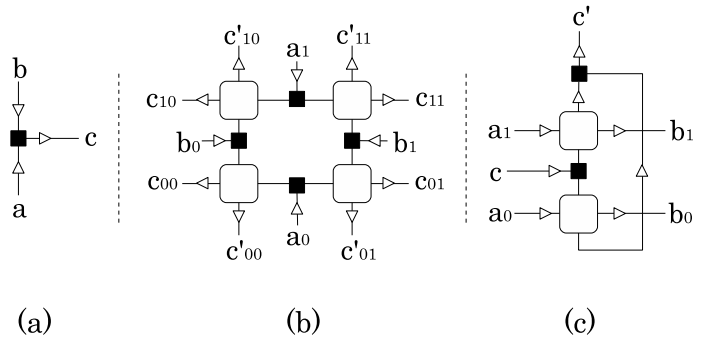

FIGURE 3
Realizations [12] of (a) Merge, (b) Conservative 2×2-Join, and (c) CSequencer by CJoin, Hub and Ratchet.

as follows: At each time step, one cell is selected randomly from the cellular space. If the state of the selected cell and the states of its neighboring cells match the lefthand side of a transition rule, then the rule is applied to update the states of the cell as well as its neighbors.

A line is represented as a linearly continuous sequence of cells that are in state 0 and state 1 alternately. Lines are arranged in diagonal directions in the cellular space, on which a signal is represented as a cell in state 1 (see Fig. 6a). Because a signal has no distinct head and tail [12, 21], driven by rule R_1, it fluctuates on lines between going forward and backward directions, as if the signal being subject to Brownian motion. The fluctuation provides a natural way to backtrack from the deadlock situation, e.g., collision of signals at a crosspoint of two crossing lines, as shown in Fig. 6c. On the contrary, non-Brownian ACA models usually require special functionality to avoid the occurrence of such deadlocks [1], which in turn causes the increase in the complexity of the model.

Implementations of the BC-elements: Hub, CJoin, and Ratchet in the cellular automaton are demonstrated in Fig. 7, respectively. In particular, implementation of the Hub in the cellular space is based on the rule R_1, as well as implementation of the CJoin being mainly based on rule R_3. In addition, the only purpose of the rule R_2 is used to implement the Ratchet.

Having implemented all BC-elements, we can construct more complex configurations in the cellular space to implement all primitive elements of

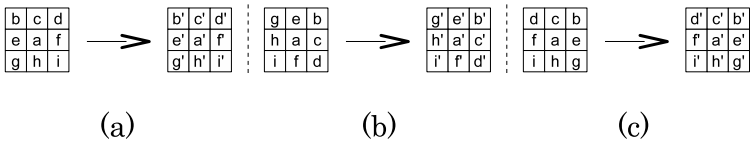

FIGURE 4
(a) Basic form of a transition rule where $a, \cdots, i, a', \cdots, i' \in \{0, 1\}$. (b) A rotational equivalence to the rule on the left of which both the lefthand and righthand sides are rotated 90 degree in a clockwise direction. (c) Similarly, a reflectional equivalence to the rule in (a).

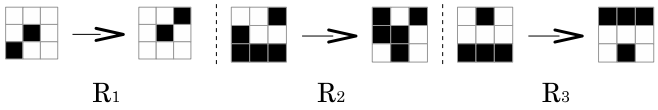

FIGURE 5
List of transition rules, with their rotational and reflectional equivalences left out.

conservative DI-circuits. The constructions are shown in Fig 8. Because the set of elements {Merge, Conservative 2×2-Join, CSequencer} is universal [17], any arbitrary DI-circuit can be embedded into the cellular space, which proves the computational universality of our cellular automaton.

Our ACA achieved a decrease in the number of cell states from the previous BCA. The key to success is due not only to the change of the neighborhood (from Neumann to Moore), but also to a somewhat different scheme for designing the transition function. The BCA model uses two distinct rules for signals: one for propagation on a line and another one for crossing. Implementation of the Ratchet is realized by ultilizing the difference between these two rules [12]. Our cellular automaton, on the other hand, explicitly prepares a specific rule (R_2) to construct the Ratchet. The lack of a second rule to cope with crossing, however, may bring difficulties to the layout of crossing lines, for example, as illustrated in Fig. 9a. Fortunately, the Ratchet elements can be used to avoid such erroneous crossing occurs (see Fig. 9b), and thus any pair of orthogonal lines can be arranged to cross each other properly.

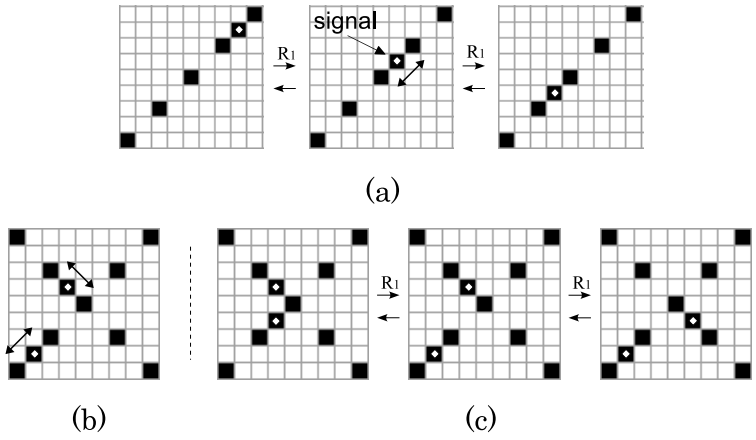

FIGURE 6
(a) Sequence of configurations where, due to rule R_1, a signal fluctuates between both directions of the line, resembling a particle under Brownian motion. The 1-state cell representing a signal is labeled by a white diamond-shaped mark, in order to distinguish it from other cells (in state 1) on a line. Moreover, each time a transition rule (or one of its rotational and reflectional equivalences) is used, its label in Fig. 5 is denoted above the corresponding arrow. (b) Configuration of two orthogonal lines intersecting at a common cell in state 1 on both lines. (c) A typical sequence of configurations where one signal propagates on each of two crossing lines. In this case, whenever collision of the signals at the crosspoint occurs (the leftmost configuration), the Brownian motion of signals will eventually move one signal backward away from the crosspoint, so as to allow another signal to cross first. Thus, signals on different lines can propagate independently without interference with each other.

In the following section, we show that even without the use of the transition rule R_3 as well as the local configuration of CJoin, all conservative DI-circuits still can be constructed, which enables a further decrease in the number of transition rules of our ACA.

4 FURTHER REDUCING THE TRANSITION RULES OF THE ASYNCHRONOUS CELLULAR AUTOMATON

For simplicity, we say that the line K of a CJoin in Fig. 2 is equivalent to the line K', so as the line J being equivalent to J'. When we construct the conservative DI primitives (Fig. 3), each CJoin element is subject to the place-

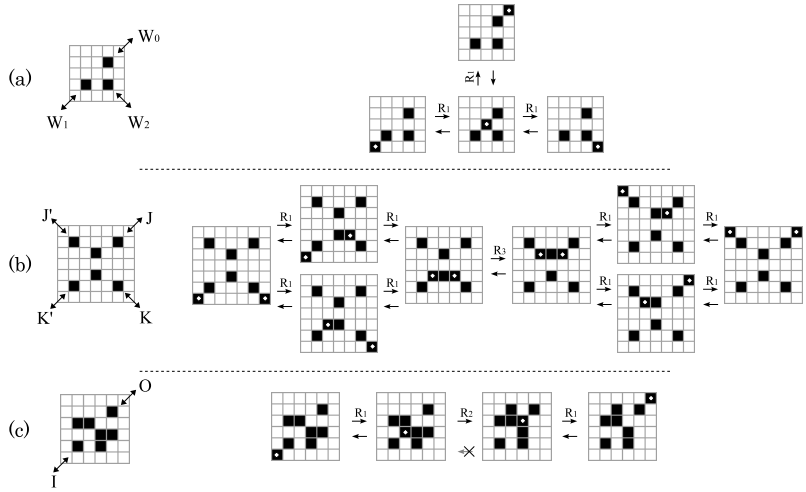

FIGURE 7
(a) Local configuration representing a Hub, and sequence of configurations where the Hub receives one signal on one of its lines, after which it transfers the signal to either of other lines, and vice versa. (b) Local configuration representing a CJoin, and sequence of configurations where the CJoin receives two signals one on each of its lines K' (or J') and K (resp. J), after which it produces two signals one to each of the lines J (resp. K) and J' (resp. K'), respectively. Due to asynchronicity of the CA, several different sequences of configurations are possible. (c) Local configuration representing a Ratchet, and sequence of configurations where the Ratchet receives one signal on line I, after which it transfers the signal to line O. Though a signal may fluctuate on lines I and O before and after being processed by the Ratchet, it can not go back from line O to line I.

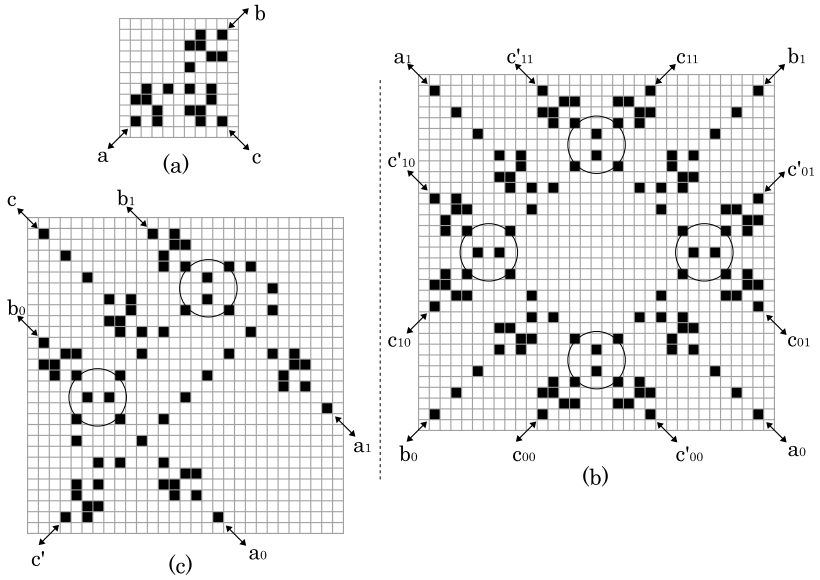

FIGURE 8
Constructions of (a) Merge, (b) Conservative 2×2-Join and (c) CSequencer elements in the cellular space. Within each construction, configurations of the Hub, CJoin and Ratchet are laid out at appropriate regions in accordance with the circuit scheme in Fig. 3, and then connecting them by lines. For clarity, each local configuration of a CJoin element is surrounded by a circle.

ment of Ratchets on one pair of its equivalent lines. Thus, one can consider the sub-circuit composed by a CJoin along with one Ratchet being placed on each of two equivalent lines as a new element, as defined in Fig. 10a, which obviously has a partial functionality of the CJoin. Furthermore, constructions in Fig. 3 show that this element (sub-circuit) along with the Hub and Ratchet can be used to construct any arbitrary conservative DI-circuit, i.e., they form a universal set for DI-circuits.

More importantly, instead of extending the configuration of a CJoin, the sub-circuit in Fig. 10a can be implemented straightforward by a configuration (see Fig. 10b) which closely resembles the configuration of a Ratchet in Fig. 7c. In this case, Fig. 10c shows that application of the transition rule R_2 along with rule R_1 can fully accomplish the sub-circuit's functionality. Re-

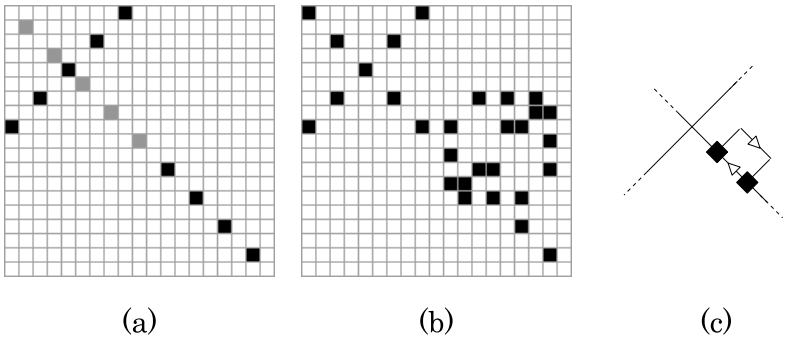

FIGURE 9
(a) Invalid crossing occurring between a pair of orthogonal lines, in which the crosspoint occupies a cell in state 1 on one line, but a state-0 cell on another line. Since the crosspoint must be a common cell in state 1 shared by both lines (see Fig. 6), two orthogonal lines may not be able to intersect properly. (b) Following the scheme in (c), installation of the Ratchet elements in the line that will intersect with the other line at a 0-state cell, thereby two orthogonal lines can cross each other properly. (c) Circuit composed by Ratchet and Hub elements, of which the functionality is no more than a simple (bi-directional) line.

moval of the rule R_3 from the transition function, therefore, does not affect the universality of our ACA for DI-circuits. For example, Fig. 11 illustrates the constructions of a Conservative 2×2-Join and a CSequencer that are almost as similar as those constructions in Fig. 8, except that each local configuration of the CJoin in latter has been replaced by a configuration of the sub-circuit in the former.

In conclusion, our asynchronous cellular automaton requires only two cell states $\{0,1\}$ as well as two kinds of transition rules $\{R_1, R_2\}$ (Fig. 5) for constructing the whole set of DI-circuits, i.e., it is computationally universal.

5 CONCLUSIONS

Random fluctuation provides an effective and powerful resource for biological systems [25]. The Brownian cellular automaton designed in [12] demonstrated that it can be employed efficiently in the searching processes associated with computation. In particular, the fluctuation allows a natural realization of arbitration and choice, a functionality that is essential for asyn-

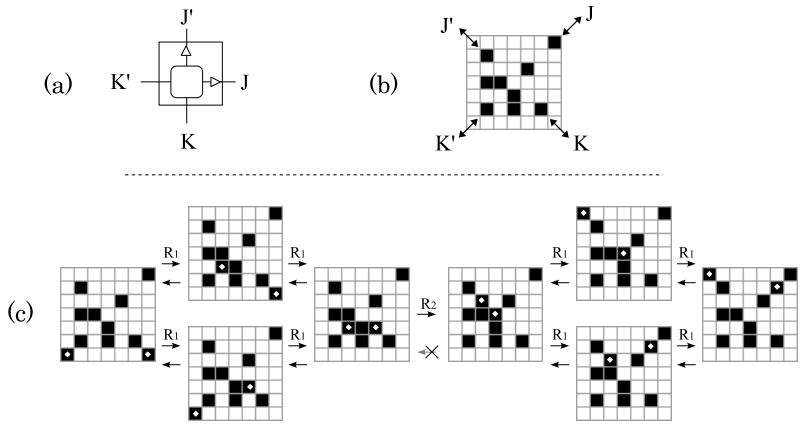

FIGURE 10
(a) Sub-circuit composed by CJoin and Ratchet elements. Unlike the CJoin, this circuit possibly has two input lines K and K', and two output lines J and J'. (b) Local configuration representing the sub-circuit. (c) Sequence of configurations where the sub-circuit receives two signals one on each of the lines K and K', after which it produces one signal on each of the lines J and J'. In this case, a signal on line J or J' can not go back to either of lines K and K'.

chronous systems but usually hard to implement without the use of Brownian motion (e.g. [8, 10]). As a result, the BCA requires merely three cell states and three transition rules, which is far less than comparable ACA models with computational universality achieved thus far.

This paper took the fluctuation-based computing approach one step further, by introducing a new ACA model that can actively exploit the Brownian motion in a similar way as the BCA. This model employs two cell states and two kinds of transition rules, and is capable of universal computation. The fewer number of cell states is due to some degree to the use of a larger neighborhood by our ACA as compared to the BCA. The decrease in the number of transition rules, however, has little relevance to the neighborhood size, but is achieved via the implementation of some partial functionalities of the Brownian circuit elements [12]. The reduced complexity of our ACA may contribute to the development of feasible computer architecture based on (asynchronous) cellular automata [2, 3, 4, 6, 18].

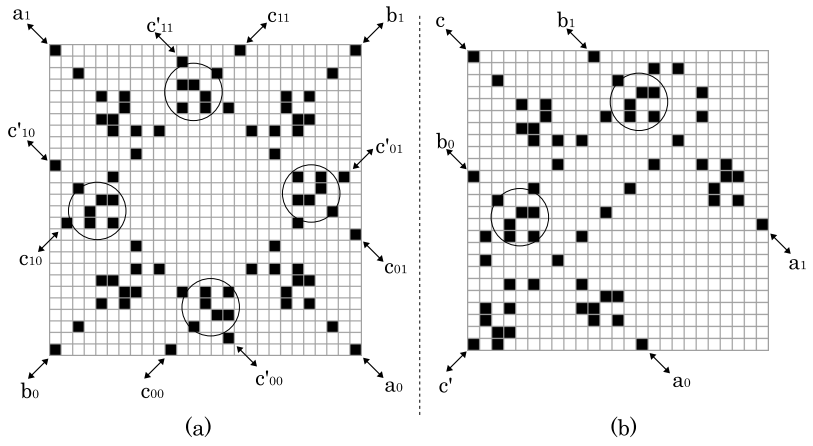

FIGURE 11
New constructions of (a) Conservative 2×2-Join and (b) CSequencer based on the sub-circuit in Fig. 10. Every configuration of the circuit is indicated by a circle.

We verified the correctness of our ACA via a Java-based simulator, by applying a general updating scheme called *uniform choice* [23] to iterate cells. By uniform choice, at each time step the cell to be updated is selected randomly from the entire space with uniform probability. Finally, we say that the 2 states and 2 rules seem the minimal numbers of cell states and transition rules required by an ACA to be capable of universal computation. Further investigations based on various ACA models, of course, are needed to confirm this conjecture.

ACKNOWLEDGMENTS

The author especially thanks Dr. Ferdinand Peper at National Institute of Information and Communications Technology (NICT), Japan, for the fruitful discussions on fluctuation-based computing models. This research work was partially supported by the Chinese Program for New Century Excellent Talents in University (No. NCET-09-0840), and the Fundamental Research Funds for the Central University (No. CDJRC10180008).

REFERENCES

[1] Adachi S., Lee J., Peper F. (2004). On signals in asynchronous cellular spaces, *IEICE Trans. Inform. Systems*, *E87-D(3)*, pp.657–668.

[2] Bandyopadhyay A., Fujita D., Pati R. (2009). Architecture of a massively parallel processing nano-brain operating 100 billon molecular neurons simultaneously, *Int. J. Nanotechnology and Molecular Computation*, *1*, pp.50–80.

[3] Biafore M. (1995). Cellular automata for nanometer-scale computation, *Physica D*, *70*, pp.415–433.

[4] Durbeck L.J.K., Macias N.J. (2001). The cell matrix: an architecture for nanocomputing, *Nanotechnology*, *12*, pp.217–230.

[5] Hauck S. (1995). Asynchronous design methodologies: an overview, *Proc. IEEE*, *83(1)*, pp.69–93.

[6] Henrich A.J., Lutz C.P., Gupta J.A., Eigler D.M. (2002). Molecule cascades, *Science*, *298(5597)*, pp.1381–1387.

[7] Keller R.M. (1974). Towards a theory of universal speed-independent modules, *IEEE Trans. Comput.*, *C-23(1)*, pp.21–33.

[8] Lee J., Adachi S., Peper F., Morita K. (2003). Embedding universal delay-insensitive circuits in asynchronous cellular spaces, *Fund. Inform.*, *58(3-4)*, pp.295–320.

[9] Lee J., Adachi S., Peper F., Morita K. (2004). Asynchronous Game of Life, *Physica D*, *194*, pp.369–384.

[10] Lee J., Peper F., Adachi S.,Morita K. (2004). Universal delay-insensitive circuits with bi-directional and buffering lines, *IEEE Trans. Comput.*, *53(8)*, pp.1034–1046.

[11] Lee J., Adachi S., Peper F., Mashiko S. (2005). Delay-insensitive computation in asynchronous cellular automata, *J. Comput. System Sci.*, *70*, pp.201–220.

[12] Lee J., Peper F. (2008). On Brownian cellular automata, In: A. Adamatzky, R. Alonso-Sanz, A. Lawniczak, G.J. Martinez, K. Morita, T. Worsch (Eds.), *Theory and Application of Cellular Automata*, Luniver Press, pp.278–291.

[13] Lee J., Peper F., et al. (2010). Brownian circuits—part II, In preparation.

[14] Margolus N. (1984). Physics-like models of computation, *Physica D*, *10*, pp.81–95.

[15] Nakamura K. (1981). Synchronous to asynchronous transformation of polyautomata, *J. Comput. System Sci.*, *23*, pp.22–37.

[16] Patra P. (1995). Approaches to the Design of Circuits for Low-Power Computation, *Ph.D. Thesis*, University of Texas at Austin.

[17] Patra P., Fussell D.S. (1996). Conservative delay-insensitive circuits, *Workshop on Physics and Computation*, pp.248–259.

[18] Peper F., Lee J., Adachi S., Mashiko S. (2003). Laying out circuits on asynchronous cellular arrays: a step towards feasible nanocomputers? *Nanotechnology*, *14(4)*, pp.469–485.

[19] Peper F., Lee J., Abo F., Isokawa T., Adachi S., Matsui N., Mashiko S. (2004). Fault-tolerance in nanocomputers: a cellular array approach, *IEEE Trans. Nanotechnol.*, *3(1)*, pp.187–201.

[20] Peper F., Lee J., et al. (2010). Brownian circuits—Part I, In preparation.

[21] Peper F., Lee J., Isokawa T. (2010). Brownian cellular automata, *J. Cellular Automata*, *5(3)*, pp.185–206.

[22] Priese L. (1983). Automata and concurrency, *Theor. Comp. Sci.*, *25*, pp.221–265.

[23] Schönfisch B., de Roos A. (1999). Synchronous and asynchronous updating in cellular automata, *BioSystems*, *51*, pp.123–143.

[24] Wegrzyn A., Karatkevich A., Bieganowski J. (2004). Detection of deadlocks and traps in Petri nets by means of Thelen's prime implicant method, *Int. J. Appl. Math. Comput. Sci.*, *14*, pp.113–121.

[25] Yanagida T., Ueda M., Murata T., Esaki S., Ishii Y. (2007). Brownian motion, fluctuation and life, *Biosystems*, *88(3)*, pp.228–242.