On weak universality of three-dimensional Larger than Life cellular automaton

Katsunobu Imai, Kyosuke Oroji, and Tomohiro Kubota
Hiroshima University, Hiroshima, Japan
imai@hiroshima-u.ac.jp

Abstract. Larger than Life cellular automaton (LtL) is a class of cellular automata and is a generalization of the game of Life by extending its neighborhood radius. We have studied the three-dimensional extension of LtL. In this paper, we show a radius-4 three-dimensional LtL rule is a candidate for weakly universal one.

Keywords: cellular automata, Larger than Life, three-dimension, weak universality

1 Introduction

A class of two-dimensional cellular automata, Larger than Life (LtL) [6], is a natural generalization of the game of Life [4] by extending the radius of its neighborhood. A lot of interesting patterns such as bugs are found in LtL [7]. A bug is a kind of glider or spaceship patterns found in the game of Life [9]. An radius-5 LtL rule, Bosco [6] has high computing efficiency employing the function of bugs and the other coherent structures [8].

As far as the game of Life, there are several studies about three-dimensional extensions [12]. Although there are many combination of the ranges of birth and survival, Bays found some better candidates as the three-dimensional “Life” and found several glider patterns [13]. Although the computational universality of (two-dimensional) Life was shown more than 35 years ago [4], there is no related study of universality of three-dimensional Life, to the best of our knowledge.

We have studied LtL rules in three-dimension. We showed a methodology to design period-1 bugs and constructed several period-1 bugs in the cases of radii from 3 to 7 and we also found several stable patterns and blinker patterns [10]. At the time, we also tried to find some functional patterns such as turning a bug or copying a bug by colliding a bug to blinkers or stable blocks. Because they are thought to be inevitable for embedding a computing function to a cellular space. But the trial was not fruitful. Because there was two reasons: (1) the number of found bugs was too small, (2) period-1 bugs are too ‘solid’ to recover the shapes after some reactions casedy by the collisions of another bug, stable block or blinker.

In the process of constructing these bugs and the simulation of collisions of them, we also found a period-10 bug in the case of radius 4 [10]. We noticed that
the bug has a kind of stabilizing property to recover the original shape of the bug even starting from a pattern which is not any pattern of the bug. Unfortunately the bug did not have any useful function, we have been trying to find bugs with bit longer periods.

We have finally found a radius-4 three-dimensional LtL rule as a weakly universal candidate, i.e, starting from a periodic infinite initial configuration, it is possible to simulate a universal machine (the rule 110 cellular automaton can be simulated [5]).

2 Larger than Life cellular automata

Larger than Life (LtL) [6] is a natural generalization of the game of Life and defined by $L = (r, \beta_1, \beta_2, \delta_1, \delta_2)$, where $r$ is the radius of its neighborhood, $[\beta_1, \beta_2]$ is the range of the total number of live neighborhood cells for giving birth to the central focus cell ($0 \rightarrow 1$), $[\delta_1, \delta_2]$ is the range for survival of the central focus cell ($1 \rightarrow 1$). The shape of its neighborhood is the square of the length $2r + 1$, thus the neighborhood includes $(2r + 1)^2$ cells. Note that the game of Life can be denoted by $(1, 3, 3, 3, 4)$.

In the case of LtL, Evans [6] found a lot of special glider like patterns called a bug. A bug is a pattern which crawls the cellular space and the same pattern repeatedly appears at a certain displaced position $d = (d_1, d_2)$ after a certain period $\tau$.

![Fig. 1. Family of bugs $\tau; (d_1, d_2)$ supported by Bosco’s rule (5, 34, 45, 34, 58) [6].](image)

LtL can be also defined in three-dimension in the same way, except the number of neighborhood cells is $(2r + 1)^3$. It is extremely large, for example, 729 in the case of radius-4. A displacement vector of a bug needs to be also modified to three-dimension.

3 Three-dimensional LtL rule (4, 102, 133, 102, 142)

3.1 Bug, blinkers and basic modules

In this section we show that the three-dimensional LtL rule, $L = (4, 102, 133, 102, 142)$, has interesting patterns such as a bug, blinkers, and basic modules which are useful for executing computations by embedding a logical circuit.
There is a period-13 bug $B$ which advances two cells in 13 time steps, i.e., $\tau_B = 13, d_B = (0, 2, 0)$. Fig. 2 shows the evolution of the patterns in a period. Fig. 3 shows the detailed cross-sectional pattern of $B_0$ of the bug $B$. In the following, the coordinate of the position of the black circle is used to identify the position of $B_0$.

Fig. 2. The evolution of a bug $B$ in a period.

Fig. 3. The cross-sectional view of a bug $B_0$.

We also found two blinkers $P$ and $Q$. $P$ is a period-4 blinker. Fig. 4 shows the evolution of the patterns in a period. Fig. 5 shows the cross-sectional pattern $P_2$ of the blinker $P$ in Fig. 4.

$Q$ is a period-14 blinker. Fig. 6 shows the evolution of the patterns in a period. In contrast to the blinker $P$, the blinker $Q$ is asymmetric in the structure. $Q_i$ and $Q_i + 7$ ($0 < i < 6$) are symmetric and both sequences are alternatingly appeared. Fig. 7 shows the cross-sectional pattern of $Q_1$ of the blinker $Q$ in Fig. 6.

A collision of a bug $B$ to a blinker $Q$, with a special offset and phase of a period, realizes a turning module of the bug $B$. Fig. 8 shows a process of the right turning. An input bug $B_0$ changes the direction. Place $B_0$ at $(0, 0, 0)$ and $Q_1$ at $(4, 14, 6)$ as an initial configuration, after 34 steps, a right-turned $B_0$ is generated at $14, 12, 6$ as illustrated in Fig. 9. At time 17 in Fig. 8, the shape of
**Fig. 4.** The evolution of period-4 brinker P in a period.

**Fig. 5.** The cross-sectional view of P2.

**Fig. 6.** The evolution of period-14 brinker Q in a period.
Fig. 7. The cross-sectional view of Q1.

The output pattern is almost recovered but it takes several more steps to converge the exact shape of the bug B.

Fig. 8. A process of turning a bug.

Fig. 9. A layout of turning configuration in $xy$-plane and the relative positions of input and output bugs.

A *duplicator* of a bug B can be realized by a collision of B to two blinkers, Q, with an offset and a phase. Fig. 10 shows a duplicating process of a bug B. A bug B0 as an input, two B0 patterns are generated as outputs. As in Fig. 11 B0 at (0, 0, 0) and two Q1 are placed at $(\pm 9, 13, 1)$ as an initial configuration, two B0 patterns finally appear at the positions $(\pm 8, 13, 1)$ after 44 time steps.

Note that these collisions are destructive and each function is only used once.
Fig. 10. A process of duplicating a bug.

Fig. 11. A layout of duplicate configuration in $xy$-plane and the relative positions of input and output bugs.

3.2 Logical gates

We shows that a NOT gate module and an AND gate module can be realized by combining a duplicator and a turning module.

Fig. 12 is a NOT gate module. A duplicator is used as a kind of “clocking” module and generate a bug. An input signal to the NOT gate is encoded by the existence of a bug. If the input is 0 then the generated bug is used as the output. If the input is 1 then the associated input bug $B_0$ causes a collision to the generated bug. Almost all types of collisions of two bugs finally annihilate all live cells produced by the collision. Thus there is no output, i.e., 0.

Fig. 13 shows an AND gate module. It can be realized by the similar way. Only in the case that two bugs are placed as inputs 1 and 1, the input2 bug survives as its output.

3.3 Synchronizing issue

The position of input and output bugs of a duplicator (a turning module) differs in $z$-coordinate. The value of the offset of a duplicator and turning module are 1 and 6 respectively. Each module flipped upside down is also effective, i.e., a
flipped duplicator is also a duplicator, and a flipped (right) turning module can be used as a (left) turning module.

The positions of duplicated bugs by the pattern in Fig. 11 is \((\pm 8, 13, 1)\) and the difference of their phase of period from the input bug is \(-5\). Because another B0 pattern appear at 44-step later, but the original bug must be B0 pattern again in 39-step later. The change of the phase of period causes a serious problem for combining the pattern of gate modules. Because we need to change the phase of blinkers P and Q depending on the number of cascading modules. But the difference of the phase of period between input and output bugs of turning module pattern (Fig. 9) is “fortunately” \(5\) \(= 39 - 34\). Thus we can reset the phase difference by combining the same number of both modules. Fig. 14 illustrates the positions of input and output bugs of a phase synchronized duplicator.

Next, we consider the logical gate patterns. In general, two inputs of an AND gate in Fig. 13 should be strictly synchronized both in their positions and phases. But in our case, only the bug of input 2 is used as its output signal and the bug of input 1 is just used as an inhibitor of the bug generated by the duplicator. Because the annihilation of bugs by collisions occurs in almost all combination

Fig. 12. NOT gate modules (input 0 and 1).

Fig. 13. An AND gate module (input 1 and 1).
of phases, we do not need to synchronize two inputs. Thus an AND gate never changes the phase of input signal. We also do not need to control the position strictly. Of course, a bug with cross mark needs to be eliminated. But almost all types of collisions with a bug and a blinker P finally cause annihilation. But to avoid confliction of their transitional evolution, one need to leave enough space between each module.

A NOT gate in Fig. 12 needs to be carefully considered because the output signal uses the bug generated from a clocking signal by a duplicator and the phase of the output bug is not synchronized. So we replace the previous NOT gate pattern with a synchronized version. Fig. 15 illustrates the input and output relation of bugs for an improved NOT gate. This layout also usable as a delay or changing the positions of bugs for wiring a circuit.

4 Conclusion

In this paper, we show that an LtL, (4, 102, 133, 102, 142), is a candidate for weak universality. Because universal logic elements can be embedded and a necessary wiring method are shown, it should be possible to build a cell module for the rule 110 and placing it infinitely but periodically to simulate the evolution of the rule 110 cellular automaton. To the best of our knowledge, this rule is the first universal candidate among three-dimensional life-like and LtL rules.

As in the case of two-dimensional LtL rule, Bosoco, a bug gun should be the most important module. It seems inevitable to achieve strong universality, i.e., universality with finite initial configuration. We strongly believe the existence of Bosco like rules in three-dimensional LtL. But we can’t find any bug gun so far in three-dimension.
Acknowledgments. This work is supported by the JSPS KAKENHI Grant Numbers JP26330016, JP17K00015.

References

1. Bays, C., Candidates for the game of Life in three dimensions, *Complex Systems*, 1, 373-400 (1987).
2. Bays, C., A new game of three-dimensional Life, *Complex Systems*, 5, 15-18 (1991).
3. Bays, C., Further notes on the game of three-dimensional Life, *Complex Systems*, 8, 67-73 (1994).
4. Berlekamp, E.R., Conway, J.H., Guy, R.K., Winning ways for your mathematical plays, vol. 2, Academic Press (1982).
5. Cook, M.: Universality in elementary cellular automata, *Complex Systems*, 15 (1) 1–40 (2004).
6. Evans, K.M.: Larger than Life: Digital creatures in a family of two-dimensional cellular automata, *Discrete Mathematics and Theoretical Computer Science Proceedings* AA (DM-CCG) 177–192 (2001).
7. Evans, K.M.: Larger than Life: threshold-range scaling of Life’s coherent structure, *Physica D* 183, 43–67 (2001).
8. Evans, K.M.: Is Bosco’s rule universal?, *Proceedings of Machines, Computations, and Universality*, Saint Petersburg, *Lecture Notes in Computer Science* 3354, Springer, 188–199 (2004).
9. Gardner, M.: Mathematical games – the fantastic combinations of John Conway’s new solitaire game, Life, *Scientific American* 223, 120–123 (1970).
10. Imai, K., Imai, K., Masamori, Y., Iwamoto, C., and Morita, K.: On designing gliders in three-dimensional Larger than Life cellular automata, *Natural Computing: 4th International Workshop on Natural Computing*, Himeji, *PICT* 2, Springer, 184-190 (2010).