# Self-Stabilizing Distributed Algorithms by Gellular Automata

**Taiga Hongu**
**Masami Hagiya**
*The University of Tokyo*
*Tokyo, Japan*
*hongu314@g.ecc.u-tokyo.ac.jp*
*hagiya@is.s.u-tokyo.ac.jp*

Gellular automata are cellular automata with the properties of asynchrony, Boolean totality and noncamouflage. In distributed computing, it is essential to determine whether problems can be solved by self-stable gellular automata. From any initial configuration, self-stable gellular automata converge to desired configurations, as self-stability implies the ability to recover from temporary malfunctions in transitions or states. This paper shows that three typical problems in distributed computing, namely, solving a maze, distance-2 coloring and spanning tree construction, can be solved with self-stable gellular automata.

*Keywords*: gellular automata; solving a maze; distance-2 coloring; spanning tree construction; self-stability

## 1. Introduction

Many studies have been conducted to implement cellular automata using physical or chemical materials, such as [1–3]. These include recent efforts to implement cellular automata by reaction-diffusion systems in porous gels [4]. One motivation for implementing cellular automata using gels is to develop smart materials that can autonomously respond to external environments.

The term *gellular automata* was coined in [5], where the diffusion of DNA molecules is controlled by opening and closing holes between cells. Gellular automata were later formalized as cellular automata with the features of asynchrony, Boolean totality and noncamouflage in [6, 7], where two types of DNA molecules were assumed, one for states of cells and the other for signals transmitting states.

In the research in the latter direction, the computational universality of gellular automata was shown [6], and the computational power of gellular automata as distributed systems was investigated in [8]. Self-stability is a crucial factor in distributed computing. According to [9], self-stability is the ability of a system to converge to states with desired conditions from any initial state. If gellular automata are self-stable, they recover desired conditions even if temporary malfunctions occur in transitions or states. Smart materials are expected to have this property.

In our previous study, we developed self-stable gellular automata that solved a maze [10] using a distributed algorithm similar to Lee's algorithm [11], under the restrictions that the number of states is finite and state transitions are asynchronous. However, this system takes time to detect undesired situations, such as loops.

In this paper, we reconsider the transition rules and target configurations of the gellular automata and present new transition rules that can solve a maze in a relatively short time. Moreover, we examine two other typical problems in distributed computing: distance-2 coloring and spanning tree construction. Like solving a maze, we confirm that these problems can be solved with self-stable gellular automata and explain how to design suitable systems for this purpose.

There are many studies on cellular automata solving maze problems, such as [12, 13], but we could not find self-stable ones except ours. Self-stable cellular automata for $k$-coloring are proposed by a very recent study [14]. They examined $k$-coloring by deterministic cellular automata and probabilistic ones, and suggest a different self-stabilizing algorithm for the number of colors $k$. However, typical distributed problems such as mazes and spanning trees are not dealt with. Also, self-stability is defined from the viewpoint of the probabilistic model in their paper, while it is defined from the viewpoint of the nondeterministic model in ours.

The gellular automaton for solving a maze and others are demonstrated by the simulator available at cell-sim-ca0d6.firebaseapp.com. Select "New Maze" in "Simulation Target."

## 2. Solving a Maze

### 2.1 Definitions

In this paper, a two-dimensional square lattice and a von Neumann neighborhood are assumed. Each cell in the square lattice has a state from the following set:

$$\{W, B, S, T_0, T^*, T^\dagger, R\} \cup \{P_i', P_i'', P_i^*, P_i^\dagger \mid i = 0, 1, 2, \ldots, n-1\}.$$

The states $T_0$, $T^*$ and $T^\dagger$ are denoted $T$. The states $P_i'$, $P_i''$, $P_i^*$ and $P_i^\dagger$ are collectively denoted $P_i$. If $i$ is arbitrary, $P_i$ is simply denoted $P$. The parameter $n$ is the number of states in $P_i$ and is equal to five in this section.

The state $W$ denotes a *wall* of a maze, which does not make any transitions. The state $B$ denotes a *blank*, which may make a transition to $P$ or $R$. The states $S$ and $T$ are the *starting point* and the *terminal point*, respectively, and they do not make any transitions. The state $R$ indicates that it is reachable from the terminal point, and a path consisting of $P$ stretches over cells in $R$.

The superscripts $'$, $''$, $*$ and $\dagger$ are used for detecting junctions by rules (9–21), explained later, and the subscripts $i$ are used for directing paths.

**Definition 1** (transition rule). A transition rule consists of three components: the current state of a cell that makes a transition, a condition to be satisfied by the neighboring cells and the next state that the cell will take.

**Definition 2** (asynchrony, Boolean totality, noncamouflage). Cellular automata are asynchronous if cells make transitions asynchronously; that is, each cell may either make a transition by following a transition rule or do nothing at each step. Cellular automata are Boolean totalistic if the conditions of transition rules depend only on neighboring cells being in a particular state, not on the direction or number of cells. Cellular automata are noncamouflage if no conditions of transition rules contain the current state of the cell that makes a transition.

A transition rule of asynchronous Boolean-totalistic noncamouflage cellular automata is defined as follows:

$$s_1(t_1 \wedge \ldots \wedge t_m \wedge \neg\, t_{m+1} \wedge \ldots \wedge \neg\, t_{m+n}) \rightarrow s_2.$$

In this rule, $s_1$ is the current state, $t_1 \wedge \ldots \wedge t_m \wedge \neg\, t_{m+1} \wedge \ldots \wedge \neg\, t_{m+n}$ is the condition, and $s_2$ is the next state. This means that a cell in state $s_1$ whose neighborhood contains cells in states $t_1, \ldots, t_n$ and does not contain cells in states $t_{m+1}, \ldots, t_{m+n}$ can make a transition to state $s_2$. By the noncamouflage property, $s_1$ does not appear among $t_1, \ldots, t_{m+n}$.

**Definition 3** (configuration, run, step). A configuration is a mapping from cells at lattice points in a square lattice to states, and a run is an infinite sequence of configurations, each of which, except for the first one, is obtained by applying the transition rules to the previous configuration. A transition step is the process of transforming from configuration $C_1$ to configuration $C_2$, which is obtained by having each cell in $C_1$ make a single transition or do nothing. Due to the asynchrony and possibility that several rules can be applied to a state, a configuration sometimes has more than one next possible configuration. In this case, one of them is chosen nondeterministically.

**Definition 4** (passage, path, loop, junction). A passage is a sequence of neighboring cells, each of which is in state $B$, $R$ or $P$. A maze is connected if there is a passage from the starting point $S$ to the terminal point $T$ in the maze.

A path is a sequence of neighboring cells in states $\ldots, P_0, P_1, \ldots, P_{n-1}, P_0, \ldots$, where the indices are incremented in $(\mathbb{Z}/n\mathbb{Z})$.

If a path has both ends, that is, a head that is not adjacent to $P_{i-1}$ and a tail that is not adjacent to $P_{i+1}$, we say that the path is maximal. If the head of a maximal path is adjacent to $T$ and its tail is adjacent to $S$, the maximal path is called a solving path.

If a path has no ends, it is called a loop. In particular, if a loop has no junctions (described later), we call that loop pure.

We say that a path has a junction if a cell in state $P_i$ in the path is adjacent to two or more different cells in $P_{i-1}$ (or $S$ if $i = 0$) or two or more in $P_{i+1}$ (or $T$). Such a cell in state $P_i$ is called an entrance in the former case and an egress in the latter case.

**Definition 5** (solution). A solution of a maze is a configuration in which there is only one maximal path, and its head (tail) is adjacent to the starting point $S$ (the terminal point $T$, respectively). If the maze is connected, there are solutions, and if it is not connected, there are no solutions. Figure 1 shows an example of the solutions of the maze.

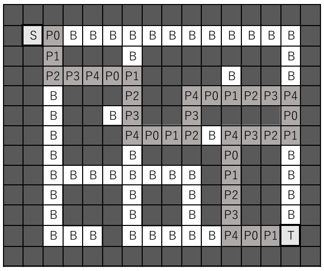


**Figure 1.** An example of solutions of a maze (black cells denote $W$).

**Definition 6** (fair run). A run $R$ is fair if a certain configuration $C$ appears in $R$ infinitely often, and any configuration $C'$ that can be obtained from $C$ by a transition step also appears in $R$ infinitely often.

Throughout this paper, we assume nondeterministic models of computation. In probabilistic models such as Markov processes, probabilities of unfair runs are zero; that is, runs are fair with probability 1.

**Definition 7** (target configuration, self-stability). Some configurations that are desirable (for a specific purpose) are defined as target configurations. Cellular automata are self-stable if, in any fair run from any configuration, a target configuration appears in finite steps, and only target configurations appear after that.

Definition 7 is generally adopted in the field of distributed computing [9]. Even when a perturbation occurs in a target configuration, a new target configuration eventually appears if cellular automata are self-stable, because we can start a fair run from the resulting nontarget configuration.

## 2.2 Procedure for Solving a Maze

### 2.2.1 Transition Rules

We introduce 21 transition rules of gellular automata for solving a maze:

$$
\begin{array}{rcll@{\qquad}rcll}
B(T_0) & \to & R & (1) & P^*_i(\neg P''_{i-1}) & \to & P'_i & (12) \\
B(R) & \to & R & (2) & P_i(P'_{i+1} \wedge P''_{i+1}) & \to & P^\dagger_i & (13) \\
R(S) & \to & P'_0 & (3) & P_i(T \wedge P''_{i+1}) & \to & P^\dagger_i & (14) \\
R(P_i \wedge \neg P_{i+2}) & \to & P'_{i+1} & (4) & P''_i(P^\dagger_{i-1}) & \to & B & (15) \\
P_i(\neg T \wedge \neg R \wedge \neg P_{i+1}) & \to & B & (5) & P^\dagger_i(\neg P''_{i+1}) & \to & P'_i & (16) \\
P_i(\neg S \wedge \neg P_{i-1}) & \to & B & (6) & T_0(P_i) & \to & T^* & (17) \\
P'_i() & \to & P''_i & (7) & T^*(P'_i \wedge P''_j) & \to & T^\dagger & (18) \\
P''_i() & \to & P'_i & (8) & P''_i(T^\dagger) & \to & B & (19) \\
P_i(P'_{i-1} \wedge P''_{i-1}) & \to & P^*_i & (9) & T^\dagger(\neg P'') & \to & T_0 & (20) \\
P_0(S \wedge P''_{n-1}) & \to & P^*_0 & (10) & T^*(\neg P) & \to & T_0 & (21) \\
P''_i(P^*_{i+1}) & \to & B & (11) & & & &
\end{array}
$$

Each rule is actually a schema of rules and represents a number of definite rules. For example, rule (4) represents $R(P'_i \wedge \neg P'_{i+2} \wedge \neg P''_{i+2}) \to P'_{i+1}$ and $R(P''_i \wedge \neg P'_{i+2} \wedge \neg P''_{i+2}) \to P'_{i+1}$ for each $i$, because $P'_i$ and $P''_i$ are collectively denoted by $P_i$.

If a cell in state $B$ is adjacent to $T_0$ or $R$, rules (1–2) change its state to $R$. In this way, we can detect all reachable cells from the terminal point $T$. Once a cell in state $B$ adjacent to the starting point $S$ makes a transition to $R$, rules (3–4) generate a path from $S$ and extend it while making as few loops as possible by preventing the path from joining an existing path.

Rules (5–6) are intended to reduce dead ends of paths. If the head of a path is not adjacent to $T$ and cannot stretch any more because there are no neighboring cells in state $R$, it changes back to state $B$. Similarly, if the tail of a path is not adjacent to $S$, it changes back to state $B$.

Entrances are reduced with rules (7–12). First, cells in state $P_i'$ or $P_i''$ switch their states from $P_i'$ to $P_i''$ or $P_i''$ to $P_i'$. Next, a cell adjacent to both $P_i'$ and $P_i''$ finds itself being an entrance and makes a transition to state $P_i^*$. Then, one (or more) of the paths joining at the entrance cell disappears gradually, and the entrance changes back to state $P_i'$.

Figure 2 shows the procedure for reducing entrances. Egresses are also reduced with rules (13–16).

Rules (17–21) restrict the number of paths reaching the terminal point $T$ to fewer than one. If the terminal point $T_0$ is adjacent to cells in state $P$, it makes a transition to state $T^*$, and no more cells in $R$ are generated from it. As in the case of junctions, if the terminal point $T$ is adjacent to several cells in $P$, one (or more) of the paths joining at $T$ disappears gradually.

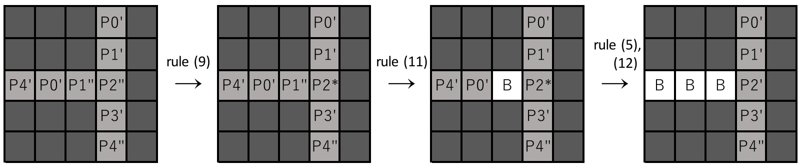


**Figure 2.** Reduction of an entrance by rules (7–12).

## 2.2.2 Self-Stability of Solving a Maze

An initial configuration is a configuration that satisfies all of the following conditions:

(I-1) There is just one starting point $S$ and one terminal point $T$, and these are not adjacent.

(I-2) The number of cells not in state $W$ is finite.

A target configuration is a configuration that does not satisfy conditions (I-1) and (I-2) for an initial configuration or that satisfies all of the following conditions:

(T-1) If the maze is connected, there is only one solving path from $S$ to $T^*$.

(T-2) There are no maximal paths except solving paths.

(T-3) There are no cells in $R$ adjacent to $S$, $P$ or $B$.

(T-4) If the maze is not connected, there is a cell in $T_0$ not adjacent to $B$ or $P$.

(T-5) There are no junctions; that is, there are no cells in $P_i$ adjacent to two or more cells in $P_{i-1}$ (or $S$ if $i = 0$), or two or more in $P_{i+1}$ (or $T$).

(T-6) There are no cells in $P_i^*$ or $P_i^\dagger$.

We now prove that these gellular automata are self-stable.

**Theorem 1.** (Self-Stability of Solving a Maze) Gellular automata with the given states, transition rules and conditions of target configurations are self-stable.

First, we show that from any initial configuration, a target configuration appears after a finite number of steps. Second, we show that once a target configuration appears, only target configurations appear afterward.

**Lemma 1.** Assume that from any initial configuration, a target configuration can be obtained by some transition steps. Then a target configuration appears in any fair run.

*Proof.* Assume that no target configuration appears in a fair run. As the cellular space is finite, the number of possible configurations is also finite. Therefore, there exists a configuration $C$ that appears an infinite number of times in the run, and because of fairness, any configurations that can be obtained from $C$, including a target configuration, also appear in the run. This is a contradiction. □

To prove the theorem, we first show that we can obtain a target configuration from any initial configuration by the following operations (i)–(iv) in order.

(i) We spread $R$ by rules (1–2) until they can no longer be applied, then spread $P$ by rules (3–4) until they can no longer be applied.

(ii) By applying rules (5–16), we remove all maximal paths except solving paths. Then there are only solving paths without junctions and pure loops. If there remain cells in $P_i^*$ or $P_i^\dagger$, we get rid of them by applying rules (8, 12, 16).

(iii) If the maze is connected and there are solving paths, we move to (iv).

If the maze is connected, but there are no solving paths because the terminal point $T$ is not adjacent to $P$, we change $T$ to $T_0$ by applying rules (20–21). We then spread $R$ from $T_0$ only on the passage that will be a solving path without junctions. When $R$ is adjacent to $P$ in a pure loop, we change all $R$ on the passage to $P$ by applying rules (3–4) and remove the loop and the resulting path using rules (5–16). By repeating this process, a single solving path is obtained.

If there remain cells in $P_i^*$ or $P_i^\dagger$, we get rid of them using rules (8, 12, 16). There should be no cells in $R$ adjacent to $B$, $S$ and $P$ because we spread $R$ only on the passage of a solving path, and there should be no junctions.

If the maze is not connected, we also change $T$ to $T_0$, as above. We then spread $R$ to all cells reachable from $T$ while removing pure loops by rules (3–4), (7–16). Then there should be no cells in $R$ adjacent to $B$, $S$ and $P$ and no junctions. Moreover, $T_0$ should not be adjacent to $B$, $P$.

(iv) If there are two or more solving paths, we keep one and remove the others by applying rules (17–21). Because there are no junctions in the paths, we can remove them by applying rule (5) until just one cell is adjacent to $S$. Finally, we change $T$ to $T^*$ by rule (17).

Figure 3 shows the operations (i)–(iv).

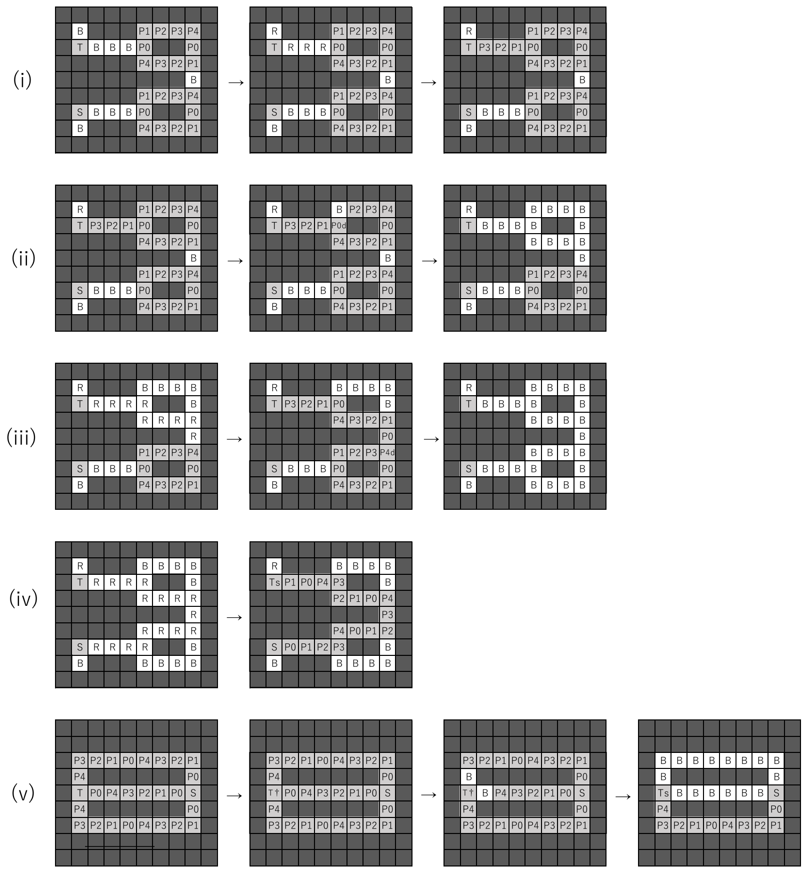


**Figure 3.** Procedure of the operation (i–iv).

We now show that only target configurations appear after a target configuration is obtained. Table 1 shows the conditions satisfied after each transition step.

| | (I) | (II) | (III) | (IV) | (V) | (VI) | (VII) | (VIII) |
|---|---|---|---|---|---|---|---|---|
| Initial State | x | x | x | x | x | x | x | x |
| operation (i) | o | o | x | x | o | x | x | x |
| operation (ii) | o | o | o | o | x | x | x | x |
| operation (iii) | o | o | o | o | o | o | x | x |
| operation (iv) | o | o | o | o | o | o | o | o |

**Table 1**. Conditions of mazes satisfied after each operation.

(I) There are no cells in $R$ adjacent to $S$, $P$.

(II) There are no cells in $B$ adjacent to $R$.

(III) There are no junctions and no cells in $P_i^*$ or $P_i^\dagger$.

(IV) There are no maximal paths except solving paths.

(V) If the maze is unconnected, there is a cell in $T_0$ not adjacent to $B$, $P$.

(VI) If the maze is connected, there are solving paths.

(VII) If the maze is connected, each of the cells in $S$ or $T$ has just one neighboring cell in $P$.

(VIII) If the maze is connected, there is a cell in $T^*$.

After (iv), all of the conditions (I–VIII) are satisfied. We can see that the conditions of the target configurations are satisfied. (T-1) holds because of (III), (VI), (VII), (VIII). (T-2), (T-3), (T-4), (T-5) and (T-6) hold because of (IV), (II), (V), (III) and (III). (In fact, if and only if all of the conditions (I–VIII) are satisfied, all of the conditions of target configurations (T-1–T-6) are satisfied.) Then only rules (7–8) can be applied to cells under target configurations, and they continue to satisfy (T-1–T-6). Therefore, after the first, only target configurations appear.

## 3. Distance-2 Coloring

### 3.1 Definitions

The space of cellular automata for solving the distance-2 coloring problem is the same as that for solving a maze. A state of a cell is either $W$, which represents a wall, or a pair $(c, \text{conf})$ of a *color state* $c$ and a *conflict state* conf. Cells whose state is a pair $(c, \text{conf})$ are called *colored*.

A color state $c$ is either $c_i'$ or $c_i''$ $(i = 1, 2, \ldots, n)$, and cells whose color state is $c_i'$ or $c_i''$ are considered to have the same color $i$. They are sometimes collectively denoted by $c_i$ or $c$, as in the case of automata for solving a maze. The parameter $n$ is the number of colors used, which is 13 in this section. A conflict state conf is a list of 0 or 1, such as $[0, 1, 0, \ldots, 1]$, whose length is $n$. The $i^{\text{th}}$ element is 1 if there are two or more neighboring cells in the color $i$ and is 0 otherwise. A cell in the color $i$ may change its color if conf$[i]$ of any neighboring cell is 1 or it has a neighboring cell in the same color $i$.

**Definition 8.** (distance-2 coloring) A colored cell is called unsafe if there is a neighboring cell in the same color as the cell or a pair of neighboring cells in the same color, and safe otherwise. A configuration is called distance-2 colored if there are no unsafe cells.

The color of any cell in a distance-2 colored configuration is different from those of the cells within a distance of two cells from it.

Figure 4 shows distance-2 coloring with five colors. Cells with different letters have different colors. Panel (a) shows a distance-2 colored configuration, but panel (b) does not because there are colored cells adjacent to two cells in R. (Note that cells $W$ adjacent to more than two cells in the same color are allowed.)

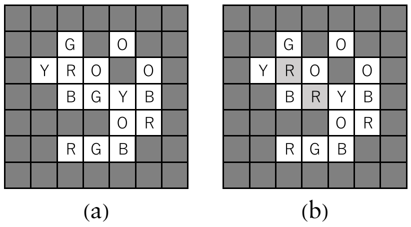


**Figure 4.** An example of distance-2 coloring of cells.

## 3.2 Procedure for Distance-2 Coloring

In this section, we introduce the transition rules and the proof of self-stability of the automata for distance-2 coloring.

### 3.2.1 Transition Rules

The automata for distance-2 coloring have the following seven transition rules. The symbol $*$ expresses an arbitrary color state or conflict state. The symbol conf$|_{[i]=j}$ is a conflict state such that conf$[i] = j$, that is, $[*, \ldots, *, j, *, \ldots, *]$, where the $i^{\text{th}}$ element is replaced by $j$:

$$\left(c_i', \text{conf}\right)() \to \left(c_i'', \text{conf}\right) \tag{1}$$

$$\left(c_i'', \text{conf}\right)() \to \left(c_i', \text{conf}\right) \tag{2}$$

$$\left(c_i, \text{conf}\ |_{[j]=0}\right)\left(\left(c_j', *\right) \wedge \left(c_j'', *\right)\right) \to \left(c_i, \text{conf}\ |_{[j]=1}\right) \tag{3}$$

$$\left(c_i, \text{conf}\ |_{[j]=1}\right)\left(\left(\neg\ c_j'', *\right)\right) \to \left(c_i, \text{conf}\ |_{[j]=0}\right) \tag{4}$$

$$\left(c_i', \text{conf}\right)((c_i'', *)) \to \left(c_j', \text{conf}\right) \tag{5}$$

$$\left(c_i'', \text{conf}\right)((c_i', *)) \to \left(c_j'', \text{conf}\right) \tag{6}$$

$$\left(c_i'', \text{conf}_1\right)\left(\left(*, \text{conf}_2\ |_{[i]=1}\right)\right) \to \left(c_j', \text{conf}_1\right). \tag{7}$$

These transition rules work as follows.

- (1–2): They switch the color states of cells from $c_i'$ to $c_i''$ or $c_i''$ to $c_i'$ to enable cells to recognize whether there are two or more neighboring cells in the same color.
- (3–4): If a cell has two or more neighboring cells in the same color $i$, they change conf$[i]$ of the cell from 0 to 1. If not, they change it from 1 to 0.
- (5–6): If a cell is adjacent to cells in the same color, they change the color of the cell to an arbitrary one.
- (7): If a cell is in a color state $c_i''$ and there is a neighboring cell whose conf$[i]$ is 1, they change its color to an arbitrary one.

### 3.2.2 Self-Stability of Distance-2 Coloring

An initial configuration is a configuration in which the number of cells not in state $W$ is finite. A target configuration is a configuration that does not satisfy the above condition for initial configurations or that satisfies all of the following conditions:

(T-1) There are no colored cells whose colors are the same as one of their neighboring cells.

(T-2) There are no colored cells that are adjacent to two or more neighboring cells with the same color.

(T-3) There are no colored cells whose conflict states are not $[0, 0, \ldots, 0]$.

Now we show that these gellular automata are self-stable.

**Theorem 2** (Self-Stability of Distance-2 Coloring). Gellular automata with the above states, transition rules and conditions of target configurations are self-stable.

As in the case of solving a maze, we consider the following operations:

(i) If there is a cell such that conf[$i$] = 1 for some $i$ and there is at most one neighboring cell of color $i$, we change its conf[$i$] from 1 to 0 by applying rule (4).

(ii) If a cell and one of its neighboring cells are the same color, we change its color according to rules (5–6). Also, if two or more neighboring cells of a cell are the same color $i$, we first make both $c_i'$ and $c_i''$ appear in the neighboring cells by rules (1–2). Then its conf[$i$] is changed to 1 by rule (3) and finally, we change $c_i''$ by rule (7). In both cases, we choose the color to which the cell changes, except the color of neighboring cells and that of the cells to which they are adjacent. As the number of colors we cannot choose is at most 12 (as in Figure 5), which is less than the number of colors, 13, we can always choose one, and the number of unsafe cells decreases.

(iii) By repeating (i–ii), we can change all of the unsafe cells to safe ones and their conflict states to [0, 0, … , 0].

A target configuration is obtained through the operations (i–iii). Then only rules (1–2) can be applied to cells in target configurations, which does not change conditions (T-1–T-3). Therefore, after the first, only target configurations appear.

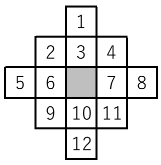


**Figure 5.** 12-cells.

## 4. Minimum Coloring

### 4.1 5-Coloring

Self-stability is proved in the case of $n = 13$ in Section 3. On the other hand, the cellular space can be colored according to distance-2 coloring within five colors as shown in Figure 6. Each number corresponds to a respective color.

In fact, in the case of $n = 5$, gellular automata with the above states, transition rules and conditions of target configurations are self-stable. At first, as in Figure 6, we determine the allocation of colors according to distance-2 coloring (called *ideal allocation* hereafter). Then, a target configuration can be obtained by operations (i–iii). However, we should revise process (ii) as follows.

(ii') As with (ii), we can determine the cell to be colored. Here, we choose the color just corresponding to the color of the cell in the ideal allocation. Then the coloring allocation gradually approaches the ideal allocation; that is, the number of the cells that are at the same coordinate in both allocations but have different colors decreases. (Note that the operation may end before the coloring allocation accords with the ideal allocation, but it satisfies the conditions of target configurations.)

Of course, we can see that only target configurations can be obtained after one appears similar to $n = 13$.

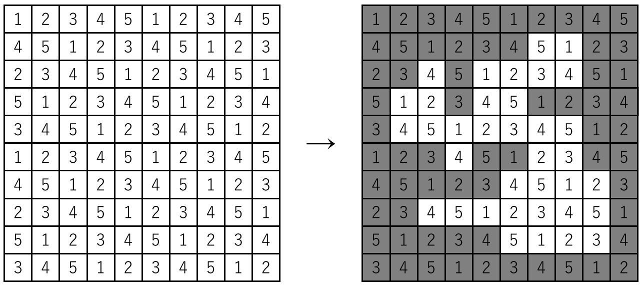


**Figure 6.** An example of 5-coloring.

### 4.2 4-Coloring

We need at least five colors for distance-2 coloring because if there is a cell in state $C$ that is surrounded by cells in state $C$, all of these five cells should have different colors. However, according to [15], 2-distance coloring of graph $G$ is achieved within $k$ colors such that $k = \text{degree}(G) + 1$. That implies that initial configurations in which there are no cells surrounded by cells in the state $C$ can be colored with four colors. Then with a similar method to 5-coloring, we can color such a configuration.

## 5. Spanning Tree

### 5.1 Definitions

The space of cellular automata for spanning tree construction is the same as that for solving a maze. A state of a cell is either $W$, which represents a wall, or a pair $(P, C)$, which should belong to a spanning tree. Here $P$ is called a *tree state* and $C$ is called a *color-conflict state*. A tree state $P$ is one of the following set:

$$\{r_i, t_i[c_j], l_i[c_j] \mid i = 0, 1, \ldots, m-1, \ j = 1, 2, \ldots, n\}.$$

A cell whose tree state is $r_i$, $t_i[c_j]$ or $l_i[c_j]$ is called a *root*, an *inner node* or a *leaf* of a spanning tree. The index $i$ is called a *wave index*, and $c_j$ is called a *parent color*. If $i$ or $j$ is arbitrary, $r_i$ is denoted $r$, and $t_i[c_j]$ is denoted by $t_i$, $t[c_j]$ and $t$ (and similarly for $l_i[c_j]$). A color-conflict state $C$ is a pair of a *color state* and a *conflict state*, similar to the case of distance-2 coloring. The parameter $m$ is the number of $r_i$, which is six in this section.

**Definition 9** (parent, child). In the following explanation, cells in $T$ are denoted as $T_i$ in order to distinguish different cells.

If a parent color of a cell $T_0$ is the same as a color of one of its neighboring cells $T_1$, then $T_1$ is a parent of $T_0$, and $T_0$ is a child of $T_1$. A cell in state $T_2$ is called an ascendant of $T_0$ only when $T_2$ is a parent of $T_0$ or (respectively) an ascendant of $T_0$. Then $T_0$ is called a descendant of $T_2$.

If there is a sequence of neighboring cells $T$ and it contains $T_0$ and $T_3$, $T_3$ is reachable from $T_0$. (Note that $T_0$ itself is also reachable from $T_0$.)

**Definition 10** (path, loop). A path is a sequence of neighboring cells whose states are aligned regularly like this:

$$\left(t_{i_0}\left[c_{j_0}\right], \left(c_{j_1}, \mathrm{conf}_0\right)\right), \left(t_{i_1}\left[c_{j_1}\right]\left(c_{j_2}, \mathrm{conf}_1\right)\right), \left(t_{i_2}\left[c_{j_2}\right]\left(c_{j_3}, \mathrm{conf}_2\right)\right),$$
$$\ldots, \left(t_{i_{k-2}}\left[c_{j_{k-2}}\right]\left(c_{j_{k-1}}, \mathrm{conf}_{k-2}\right)\right), \left(t_{i_{k-1}}\left[c_{j_{k-1}}\right]\left(c_{j_k}, \mathrm{conf}_{k-1}\right)\right).$$

The beginning of the sequence can be a root, and the end of the sequence can be a leaf. That is, a path is a sequence of neighboring cells such that:

- The tree state of each cell, except both ends of the sequence, is an inner node.
- Cells except the beginning of the sequence satisfy the following condition:
  (1) The parent color of the cell is equivalent to the color of the next cell.
- Cells except for the end of the sequence satisfy the following condition:
  (2) The color of each cell is equivalent to the parent color of the previous cell.

Assume that the sequence $T_0, \ldots, T_n$ is a path. If $T_0$ and $T_n$ are inner nodes and a color of $T_n$ is the same as a parent color of $T_0$, the path is called a *loop*. If the path is not a loop and the beginning (end) of the path $T_0$ ($T_n$) has no neighboring cell that can be a parent (child), the cell $T_1$ ($T_n$) is called a *head* (a *tail*, respectively). If a path has both *ends*, a head and a tail, we say a path is *maximal*. Notably, if there is a maximal path whose head is a root and whose tail is a leaf, the maximal path is a *branch path*.

A path has a *junction* when a cell in $T$ in the path has two or more parents or two or more children. Such a cell in $T$ is called an *entrance* in the former and an *egress* in the latter case, respectively.

If a tree state of a cell is $t_i[c_j]$ or $l_i[c_j]$, it points to a neighboring cell whose color is $j$ as its parent. In this manner, we can define a parent-child relation in the cellular space if the parent of each cell is uniquely determined. Figure 7 shows an example of the construction of a spanning tree.

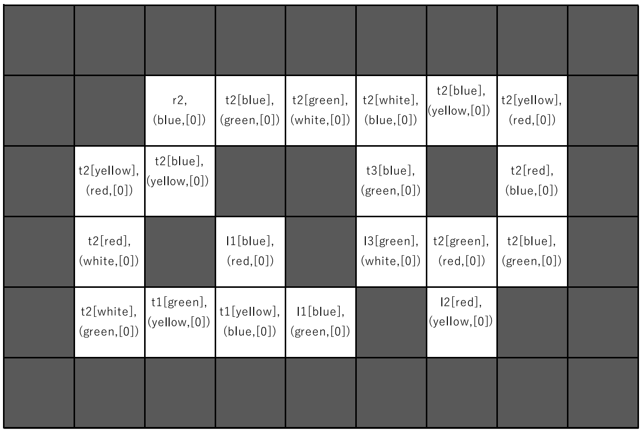


**Figure 7**. An example of a spanning tree construction.

## 5.2 Procedure for Spanning Tree Construction

### 5.2.1 Transition Rules

In order to construct a spanning tree by the following rules, distance-2 coloring is required to be achieved. If the rules (1–7) for distance-2 coloring are written in such a form:

$$s_1(t_1 \wedge \ldots \wedge t_m \wedge \neg\, t_{m+1} \wedge \ldots \wedge \neg\, t_{m+n}) \rightarrow s_2$$

then the rules (1′–7′) for spanning tree construction are needed:

$$(P, s_1)((*, t_1) \wedge \ldots \wedge (*, t_m) \wedge \neg\, (*, t_{m+1}) \wedge \ldots \wedge \neg\, (*, t_{m+n})) \rightarrow (P, s_2$$

In addition to these seven rules, there are 24 more rules, that is, 31 rules in all for spanning tree construction:

$$\left(t_i\left[c_j\right], \left(c_k, \text{conf}\right)\right)\left(\neg\, (t[c_k],\ *) \wedge \neg \left(l[c_k],\ *\right)\right) \rightarrow \left(l_i\left[c_j\right], \left(c_k, \text{conf}\right)\right) \tag{8}$$

$$\left(l_i\left[c_j\right], \left(c_k, \text{conf}\right)\right)((t[c_k],\ *)) \rightarrow \left(t_i\left[c_j\right], \left(c_k, \text{conf}\right)\right) \tag{9}$$

$$\left(l_i\left[c_j\right], \left(c_k, \text{conf}\right)\right)\left(\left(l[c_k],\ *\right)\right) \rightarrow \left(t_i\left[c_j\right], \left(c_k, \text{conf}\right)\right) \tag{10}$$

These rules mean that an inner node that does not have any child makes a transition to a leaf, and a leaf that has children makes a transition to an inner node:

$$\left(t_i\left[c_j\right], C\right)\left((\,*\,, (c_k,\ *)) \wedge \neg \left(\,*\,, \left(c_j,\ *\right)\right)\right) \rightarrow \left(t_i[c_k], C\right) \quad \left(j \neq k\right) \tag{11}$$

$$\left(l_i\left[c_j\right], C\right)\left((\,*\,, (c_k,\ *)) \wedge \neg \left(\,*\,, \left(c_j,\ *\right)\right)\right) \rightarrow \left(l_i[c_k], C\right) \quad \left(j \neq k\right). \tag{12}$$

If a cell has no neighbors whose colors are the same as its parent color, the parent color is changed to one of the colors of the neighbors. These are used to let all the reachable cells from a root merge with the spanning tree:

$$\left(t_{i-1}\left[c_j\right], C\right)\left(\left(r_i, \left(c_j,\ *\right)\right)\right) \rightarrow \left(t_i\left[c_j\right], C\right) \quad \left(i = 0, 2, 4\right) \tag{13}$$

$$\left(t_{i-1}\left[c_j\right], C\right)\left(\left(t_i[c_k], \left(c_j,\ *\right)\right)\right) \rightarrow \left(t_i\left[c_j\right], C\right) \quad \left(i = 0, 2, 4\right) \tag{14}$$

$$\left(l_{i-1}\left[c_j\right], C\right)\left(\left(r_i, \left(c_j,\ *\right)\right)\right) \rightarrow \left(l_i\left[c_j\right], C\right) \quad \left(i = 0, 2, 4\right) \tag{15}$$

$$\left(l_{i-1}\left[c_j\right], C\right)\left(\left(t_i[c_k], \left(c_j,\ *\right)\right)\right) \rightarrow \left(l_i\left[c_j\right], C\right) \quad \left(i = 0, 2, 4\right) \tag{16}$$

$$\left(l_i\left[c_j\right], C\right)() \rightarrow \left(l_{i+1}\left[c_j\right], C\right) \quad \left(i = 0, 2, 4\right) \tag{17}$$

$$\left(t_{i-1}[c_h], \left(c_j,\right)\right)\left(t_i\left[c_j\right], (\,*\,,\ *)\right) \wedge \neg \left(l_k\left[c_j\right], (\,*\,,\ *)\right) \wedge \neg \left(t_k\left[c_j\right], (\,*\,,\ *)\right)) \rightarrow \left(t_i[c_h], \left(c_j,\right)\right) \left(i = 1, 3, 5;\, i \neq k\right) \tag{18}$$

$$\left(t_{i-1}[c_h], \left(c_j,\right)\right)\left(\left(l_i\left[c_j\right], (\,*\,,\ *)\right) \wedge \neg \left(l_k\left[c_j\right], (\,*\,,\ *)\right) \wedge \neg \left(t_k\left[c_j\right], (\,*\,,\ *)\right)\right) \rightarrow \left(t_i[c_h], \left(c_j,\right)\right) \left(i = 1, 3, 5;\, i \neq k\right) \tag{19}$$

$$\left(r_{i-1}, \left(c_j,\right)\right)\left(\left(t_i\left[c_j\right], (\,*\,,\ *)\right) \wedge \neg \left(l_k\left[c_j\right], (\,*\,,\ *)\right) \wedge \neg \left(t_k\left[c_j\right], (\,*\,,\ *)\right)\right) \rightarrow \left(r_i, \left(c_j,\right)\right) \quad \left(i = 1, 3, 5;\, i \neq k\right) \tag{20}$$

$$\left(r_{i-1}, \left(c_j,\right)\right)\left(l_i\left[c_j\right], (\,*\,,\ *)\right) \wedge \neg \left(l_k\left[c_j\right], (\,*\,,\ *)\right) \wedge \neg \left(t_k\left[c_j\right], (\,*\,,\ *)\right)) \rightarrow \left(r_i, \left(c_j,\right)\right) \quad \left(i = 1, 3, 5;\, i \neq k\right) \tag{21}$$

$$\left(r_i, C\right)() \rightarrow \left(r_{i+1}, C\right) \quad \left(i = 1, 3, 5\right) \tag{22}$$

To confirm whether a spanning tree is constructed, waves are generated alternately from a root to leaves (from parents to children) and from leaves to a root (from children to parents). Wave indices are incremented in $(\mathbb{Z} / m\mathbb{Z})$, such as $\ldots, 4, 5, 0, 1, 2, \ldots$. In rules (13–22), the index $i$ is incremented. Wave index 0 is propagated from a root by changing the wave indices of cells from 5 to 0. When it reaches each leaf, its wave index is changed from 5 to 0, and instantly 0 to 1, then wave index 1 is propagated from a leaf by changing the wave indices of cells from 0 to 1. (If a cell has several children, it can be changed if all children have been changed.) When it reaches the root, its wave index is changed 0 to 1, and instantly 1 to 2, Wave indices 2, 3, 4 and 5 are propagated as well. Therefore, the wave from a root has an even number index, and the wave from leaves has an odd number index. If a spanning tree is constructed, it does not have a loop, and all the cells reachable from a root are in branch paths.

Figure 8 shows the transitions of the wave index. The root of the spanning tree, which is denoted by a gray cell, changes its wave index from 0 to 2 by applying rules (13–22) to each cell.

If a spanning tree is constructed correctly, there are no pairs of adjacent cells in $T$ the difference of whose wave indices is three, such as 0 and 3, or 2 and 5. We call such a pair an *unsafe pair*. If the difference is two, such as 0 and 2, or 3 and 5, we call a pair an *unstable pair*:

$$\left(r_i, \left(c_j, *\right)\right)\left(\neg \left(l\left[c_j\right], C\right) \wedge \neg \left(t\left[c_j\right], C\right)\right) \rightarrow \left(r_{i+1}, \left(c_j, *\right)\right). \tag{23}$$

Note that the smaller difference is considered. For instance, the difference of a pair (0,5) is one, since this is not an unstable or unsafe pair.

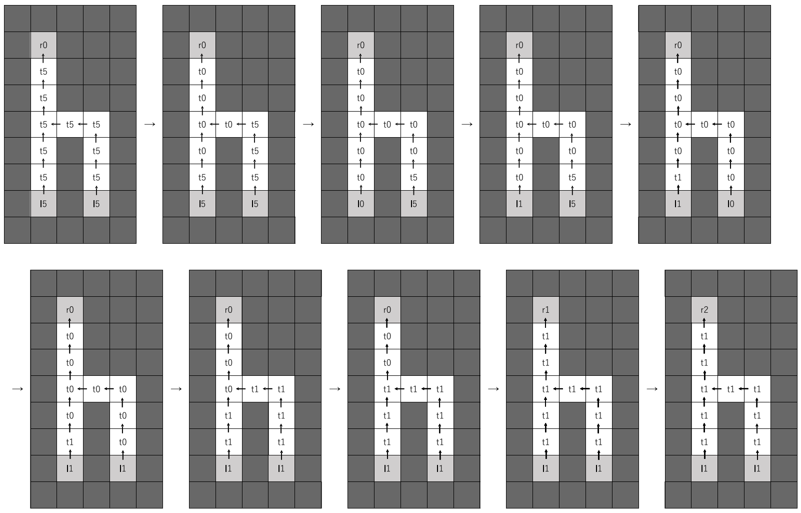

r0 t5 t5 t5 t5 t5 t5 t5 t5 t5 l5 l5
r0 t0 t0 t0 t0 t5 t0 t5 t5 t5 l5 l5
r0 t0 t0 t0 t0 t0 t0 t0 t0 t5 l0 l5
r0 t0 t0 t0 t0 t0 t0 t0 t0 t5 l1 l5
r0 t0 t0 t0 t0 t0 t0 t0 t1 t0 l1 l0
r0 t0 t0 t0 t0 t0 t0 t0 t1 t0 l1 l1
r0 t0 t0 t0 t1 t1 t1 t1 t1 t1 l1 l1
r0 t0 t0 t1 t1 t1 t1 t1 t1 t1 l1 l1
r1 t1 t1 t1 t1 t1 t1 t1 t1 t1 l1 l1
r2 t1 t1 t1 t1 t1 t1 t1 t1 t1 l1 l1

**Figure 8.** Transitions of wave index of the root from 0 to 2.

If a root does not have any child, sometimes waves are not propagated. Then the root doubles as a leaf, that is, increments its own wave index:

$$\left(t_i[c_k],\, C\right)\left(\left(r_j,\, (c_k,\, *)\right)\right) \to \left(t_j[c_k],\, C\right) \quad \left(|i-j| \geq 2\right) \tag{24}$$

$$\left(t_i[c_k],\, C\right)\left(\left(t_j[c_h],\, (c_k,\, *)\right)\right) \to \left(t_j[c_k],\, C\right) \quad \left(|i-j| \geq 2\right) \tag{25}$$

$$\left(l_i[c_k],\, C\right)\left(\left(t_j[c_h],\, (c_k,\, *)\right)\right) \to \left(l_j[c_k],\, C\right) \quad \left(|i-j| \geq 2\right). \tag{26}$$

If there is an unstable or unsafe pair of cells that are in a parent-child relation, a wave index of one of the pair is changed to be the same number as that of the parent.

Figure 9 shows the application of rules (24–26) to the descendant cells of the root. They are applied so that the root and its descendant cells have the same wave index:

$$\left(t_i[c_k],\, C\right)\left(\left(r_j,\, (c_h,\, *)\right)\right) \to \left(t_j[c_h],\, C\right) \quad \left(|i-j| \geq 3;\, k \neq h\right) \tag{27}$$

$$\left(t_i[c_k],\, C\right)\left(\left(t_j\left[c_g\right],\, (c_h,\, *)\right)\right) \to \left(t_j[c_h],\, C\right) \quad \left(|i-j| \geq 3;\, k \neq h\right) \tag{28}$$

$$\left(l_i[c_k],\, C\right)\left(\left(t_j\left[c_g\right],\, (c_h,\, *)\right)\right) \to \left(l_j[c_h],\, C\right) \quad \left(|i-j| \geq 3;\, k \neq h\right) \tag{29}$$

$$\left(t_i[c_k],\, C\right)\left(\left(l_j\left[c_g\right],\, (c_h,\, *)\right)\right) \to \left(t_j[c_h],\, C\right) \quad \left(|i-j| \geq 3;\, k \neq h\right) \tag{30}$$

$$\left(l_i[c_k],\, C\right)\left(\left(l_j\left[c_g\right],\, (c_h,\, *)\right)\right) \to \left(l_j[c_h],\, C\right) \quad \left(|i-j| \geq 3;\, k \neq h\right). \tag{31}$$

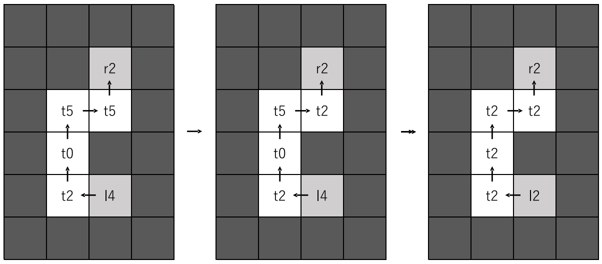


**Figure 9**. Application of the rules (24–26).

Moreover, if there is an unsafe pair of cells, not only the wave index but also the parent color of one of them is changed. The aim is to join them in branch paths.

Figure 10 shows how to take external cells into the tree. At first, wave indices of cells in the tree are changed by rules (13–22) to make an unsafe pair, and then external cells in this pair are taken into the tree by rules (27–31). These details are described in Section 5.2.2.

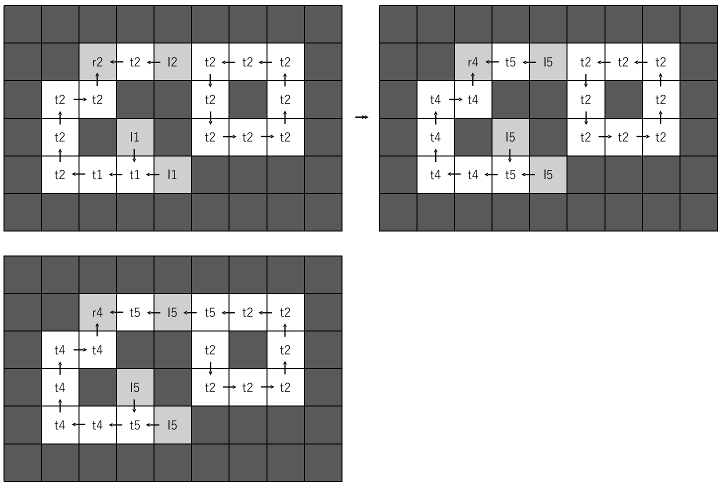


**Figure 10.** Application of rules (27–31).

### 5.2.2 Self-Stability of Spanning Tree Construction

An initial configuration is a configuration that satisfies all of the following conditions:

(I-1) There is just one root.

(I-2) The number of cells not in state *W* is finite.

A target configuration is a configuration that satisfies all of the following conditions:

(T-1) There are no cells in state *T* that are reachable from a root and in a loop.

(T-2) There are no cells in state *T* that are reachable from a root and not in any branch path.

(T-3) All the branch paths have at most one *interference pair*: a pair of adjacent cells in a branch path that are in a parent-child relation and whose difference of the wave indices is 1 in $(\mathbb{Z} / m\mathbb{Z})$, such as 5 and 0, 0 and 1.

(T-4) If the wave index of a root is 1 (3, 5), those of all its descendant cells are also 1 (3, 5).

(T-5) All cells in state *T* reachable from a root are in *valid tree states*; that is, a cell that does not have parents is a root, cells that have both parents and children are inner nodes, and cells that do not have children are leaves.

We now prove that, if distance-2 coloring is achieved, (T-1) is satisfied whenever (T-2) is satisfied.

**Lemma 2.** If distance-2 coloring is achieved, there are no entrances.

*Proof.* Given the conditions of distance-2 coloring, there are no cells whose two or more neighboring cells are the same color. If a cell is an entrance; that is, it has two or more parents, these are the same color, which contradicts the above. □

Assume that distance-2 coloring is achieved and there is a loop in one-half of a branch path; the path is in the form of $(r, *), \ldots, T_k, \ldots, T_k, \ldots, (l, *)$. It has an entrance, which contradicts the above. Thus we can show that (T-1) is satisfied whenever (T-2) is satisfied.

We also show there are no unsafe pairs of cells reachable from a root in the target configuration.

**Lemma 3.** There are no unsafe pairs of cells reachable from a root in target configurations.

*Proof.* Because of (T-2) and (T-3), the difference between the wave indices of cells that are reachable from a root and a root is at most one. Thus the wave indices of cells appearing in a configuration are one more than that of the root, the same as that of the root or one less than that of the root. Therefore, the difference of the wave index of all the reachable cells is at most two. □

Then we show that for gellular automata under these transition rules, the initial configurations and the target configurations are self-stable. At first, we obtain a target configuration from any initial configuration. Then we show no configurations except target configurations appear after that. Now the following operations (i–iii) are performed.

(i) First, distance-2 coloring is achieved by rule (1′–7′) in the same manner as Section 3.2.2.

(ii) Next, we apply rules (11–12) so that all the cells except a root have a parent. After that, we also apply rules (8–10), as these are no longer applied now, to set leaves in a correct position of paths. Then, we make the root and all of its descendant cells have the same wave index without changing the parent-child relation by the following operations: Suppose cells in a path from the root to the first interference pair have the same wave index. If a difference between wave indices of a parent and a child of the first interference pair is

(a) one (the parent is one greater than the child), if the index of the parent is 0, 2 or 4, the wave can be propagated from the root to the leaves by rules (13–16). If not, the index of the root is also 1, 3 or 5, so indices of the parent cell and all its ascendant cells can be changed to 2, 4 or 0 by rules (13–14), (22), then rules (24–26) can be applied to the child without changing the parent-child relation.

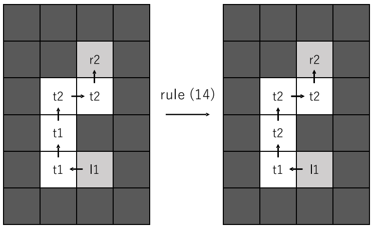
(b) one (the parent is one less than the child), if the index of the parent is 0, 2 or 4, the wave can be propagated from the leaves to the root by rules (18–21). If not, the index of the root is also 1, 3 or 5, so indices of the parent cell and all its ascendant cells can be changed to 2, 4 or 0 by rules (13–14), (22).

(c) two or more, the wave index of the child can be changed to the same number as that of the parent by rules (24–26).

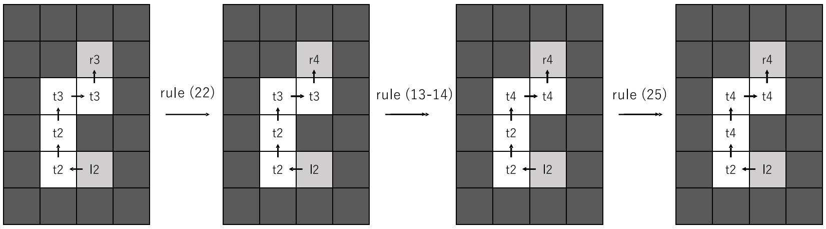
This is repeated until the root and its descendant cells have the same wave index. In such a case, we call the wave index the *tree wave index*.

Figure 11 shows the procedure of operation (ii)-(a), (b). The procedure in the case (c) is described in Figure 10.

(a) The index of the parent is 0, 2 or 4.



(a) The index of the parent is 1, 3 or 5.



(b) The index of the parent is 0, 2 or 4.

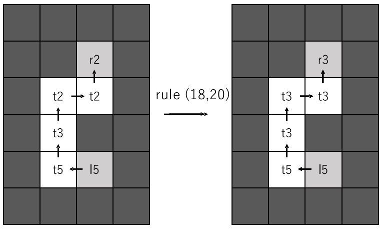


**Figure 11.** (*continues*)

(b) The index of the parent is 1, 3 or 5.

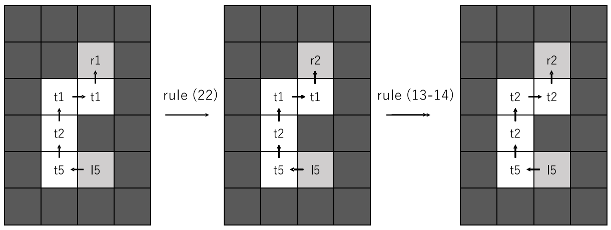


**Figure 11.** Procedure of operation (ii).

(iii) Then, we can optionally change the tree wave index by applying rules (13–22). If there is a pair of cells in which one is in a branch path and the other is not, we change the tree wave index by rules (13–22) (or rule (23) if the root has no children) so that the difference of the wave indices of them is three, change the parent color and the wave index of the latter cell to those of the former cell by rules (27–31), and repeat (ii) so that a root and all the descendants have the same wave index. This is repeated until the root and all of the cells reachable from the root are in branch paths and have the same index.

Table 2 shows the conditions satisfied after each operation.

| | (I) | (II) | (III) | (IV) | (V) | (VI) |
|---|---|---|---|---|---|---|
| Initial State | x | x | x | x | x | x |
| operation (i) | o | x | x | x | x | x |
| operation (ii) | o | o | o | x | x | x |
| operation (iii) | o | o | o | o | o | o |

**Table 2.** Conditions of a spanning tree satisfied after each operation.

(I) No cell is adjacent to more than two cells that have the same color (distance-2 colored).

(II) Root and all its descendant cells have the same wave index.

(III) All descendant cells of a root are in valid tree states.

(IV) No cells in $T$ are reachable from a root and not in any branch path.

(V) Root and all cells in $T$ reachable from the root have the same wave index.

(VI) All cells in $T$ reachable from a root are in valid tree states.

After (iii), all of the conditions (I–VI) are satisfied. (T-2) is satisfied if and only if (IV) is satisfied, and according to Lemma 2, (T-1) is also satisfied. Moreover, (T-3) and (T-4) are satisfied if (V) is satisfied, and (T-5) is satisfied if (VI) is satisfied. Thus a target configuration appears by these operations.

Then we consider which rules can be applied in the target configurations, and whether they can make the other configurations appear. Since rules (3′–7′) are not applied if a configuration is distance-2 colored, and rules (8–28) do not change the color-conflict states of cells, (3′–7′) cannot be applied to target configurations. Also (1′–2′) do not affect the conditions of target configurations, thus we do not need to consider these rules. Besides, only cells reachable from a root affect all the conditions of target configurations, so we do not consider the application of rules to unreachable cells from a root.

- (8–10) They cannot be applied because of (T-5).
- (11–12) Since all cells reachable from a root are in branch paths because of (T-2), they cannot be applied.
- (13–22) They can be applied.
- (23) Since all the cells reachable from a root are in branch paths because of (T-2), it can be applied only when the root is not adjacent to any cells in state *T*. In this case, other rules we consider, (8–21), (24–33), cannot be applied, and (23) does not change the conditions of target configurations in that case. (Rule (22) can be applied, but it is included in (23) when there is no *T* adjacent to the root.) Thus only target configurations appear, so we do not consider this rule after that.
- (24–26) Because of (T-3), the difference of wave indices of reachable cells in parent-child relation is one at most, then they cannot be applied.
- (27–31) Because of (T-3) and Lemma 3, they cannot be applied.

Then, except (1′–2′), only rules (13–22) can be applied to cells in target configurations. They only change the wave indices, so (T-1), (T-2) and (T-5) are satisfied.

When (T-4) is considered, the wave index of a root can be changed to 1 (3, 5) by rules (20–21). This can be applied only when the wave indices of all the adjacent cells are 0 (2, 4). Because of (T-3), then, those of all of the descendant cells of the root are 0 (2, 4), and rules (13–19), (22) cannot be applied at the same time. Thus (T-4) is also satisfied.

Then let us consider (T-3). There are seven patterns of two branch paths satisfying the conditions of target configurations, like Figure 12, and we show no patterns make transitions to configurations except target configurations.

(a) The index of a root is 1, 3 or 5, and all the reachable cells have the same index.

(b) The index of a root is 0, 2 or 4, and all the reachable cells have the same index.

(c) The index of a root is 0, 2 or 4, and each branch path has one interference pair such that the index of the parent is one more than that of the child.

(d) The index of a root is 0, 2 or 4, and each branch path has one interference pair such that the index of the parent is one less than that of the child.

(e) The index of a root is 0, 2 or 4, and only one of the branch paths has one interference pair such that the index of the parent is one more than that of the child.

(f) The index of a root is 0, 2 or 4, and only one of the branch paths has one interference pair such that the index of the parent is one less than that of the child.

(g) The index of a root is 0, 2 or 4, and each branch path has one interference pair such that the index of one parent is one more than that of the child, and the index of the other parent is one less than that of the child.

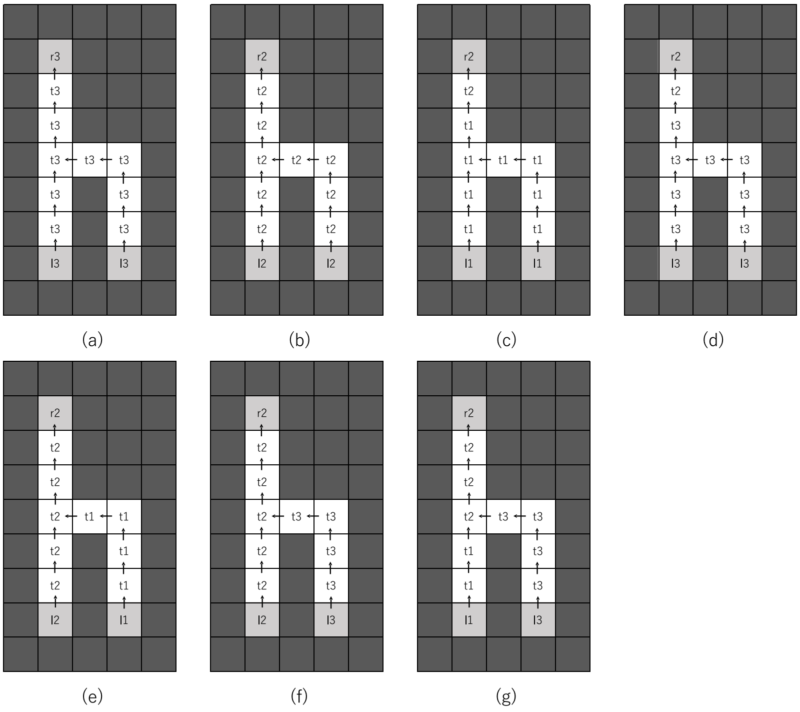


**Figure 12.** All the patterns of two branch paths.

Then we consider whether these patterns make transitions to configurations except for target configurations.

(a) In this case, only rule (22) can be applied, which does not change (T-3).

(b) In this case, only rule (17) can be applied, which does not change (T-3).

(c) In this case, rules (13–16) can be applied only to the children of the interference pairs, so it does not change (T-3).

(d) In this case, rules (13–16) can be applied only to the parents of the interference pairs. If the parents of these pairs have several children, they can be applied only when all of the children have the same index that is one more than that of the parent, so two or more interference pairs are not generated in one branch path. Thus it does not change (T-3).

(e-f) Consider when rules (13–16) (or (18–21)) and rule (17) are applied at the same time. Junction cells cannot be leaves since junctions have children, and not all the children of the junction cells have the same index in this case. Then the junction does not make a transition, there is a pair of cells the difference of which is two or more, therefore interference pairs are not generated. Thus it does not change (T-3).

(g) Similar to the case (e-f), the junction does not make a transition until all of the children have the same index. Thus it does not change (T-3).

Therefore (T-3) is also satisfied. By the above, only target configurations appear after it appears.

## 6. Conclusion

In this paper, we showed how to construct gellular automata that solve three problems: solving a maze, distance-2 coloring and spanning tree construction. As self-stable gellular automata can recover from malfunctions of states and transitions, materials that contain them are able to form structures like blood vessels or neural networks that can repair themselves following external damage or environmental changes.

By adding and changing some states and transition rules, we can also design gellular automata for solving other problems. For instance, gellular automata that solve the Hamiltonian circuit problem can be constructed by modifying those for a maze. The actual construction of these gellular automata remains for future studies. We also plan to improve the automata by decreasing the number of states and transition rules and reducing the number of steps required for them to converge to a target configuration.

## Acknowledgments

We thank Akira Yagawa for valuable discussions and implementing the simulator. We also thank the anonymous reviewers for improving the paper. This work was partially supported by Grant-in-Aid for challenging Exploratory Research 17K19961.

## References

[1] F. Peper, J. Lee, S. Adachi and T. Isokawa, “Cellular Nanocomputers: A Focused Review,” *International Journal of Nanotechnology and Molecular Computation*, **1**(1), 2009 pp. 33–49. doi:10.4018/jnmc.2009010103.

[2] D. Scalise and R. Schulman, “Emulating Cellular Automata in Chemical Reaction-Diffusion Networks,” *Natural Computing*, **15**(2), 2016 pp. 197–214. doi:10.1007/s11047-015-9503-8.

[3] P. Yin, S. Sahu, A. J. Turberfield and J. H. Reif, “Design of Autonomous DNA Cellular Automata,” in *International Workshop on DNA-based Computers (DNA 2005)*, London, ON, Canada (A. Carbone and N. A. Pierce, eds), Berlin, Heidelberg: Springer, 2005 pp. 399–416. doi:10.1007/11753681_32.

[4] T. Hosoya, I. Kawamata, S.-I. M. Nomura and S. Murata, “Pattern Formation on Discrete Gel Matrix Based on DNA Computing,” *New Generation Computing*, **37**(1), 2019 pp. 97–111. doi:10.1007/s00354-018-0047-1.

[5] M. Hagiya, S. Wang, I. Kawamata, S. Murata, T. Isokawa, F. Peper and K. Imai, “On DNA-based Gellular Automata,” in *Unconventional Computation and Natural Computation (UCNC 2014)*, London, ON, Canada (O. Ibarra, L. Kari and S. Kopecki, eds.), Cham, Switzerland: Springer, 2014 pp. 177–189. doi:10.1007/978-3-319-08123-6_15.

[6] T. Yamashita, T. Isokawa, F. Peper, I. Kawamata and M. Hagiya, “Turing-Completeness of Asynchronous Non-camouflage Cellular Automata,” in *Cellular Automata and Discrete Complex Systems (AUTOMATA 2017)*, Milan, Italy (A. Dennunzio, E. Formenti, L. Manzoni and A. Porreca, eds.), Cham, Switzerland: Springer, 2017 pp. 187–199. doi:10.1007/978-3-319-58631-1_15.

[7] T. Yamashita, T. Isokawa, F. Peper, I. Kawamata and M. Hagiya, “Turing-Completeness of Asynchronous Non-camouflage Cellular Automata,” *Information and Computation*, **274**, 2020 04539. doi:10.1016/j.ic.2020.104539.

[8] T. Yamashita and M. Hagiya, “Simulating Population Protocols by Gellular Automata,” in *57th Annual Conference of the Society of Instrument and Control Engineers of Japan (SICE 2018)*, Nara, Japan, Piscataway, NJ: IEEE, 2018 pp. 1579–1585. doi:10.23919/SICE.2018.8492697.

[9] S. Dolev, *Self-Stabilization*, Cambridge, MA: MIT Press, 2000.

[10] T. Yamashita, A. Yagawa and M. Hagiya, "Self-Stabilizing Gellular Automata," *Unconventional Computation and Natural Computation (UCNC 2019)*, Tokyo, (I. McQuillan and S. Seki, eds.), Cham, Switzerland: Springer, 2019 pp. 272–285. doi:10.1007/978-3-030-19311-9_21.

[11] C. Y. Lee, "An Algorithm for Path Connections and Its Applications," *IRE Transactions on Electronic Computers*, **EC-10**(3), 1961 pp. 346–365. doi:10.1109/TEC.1961.5219222.

[12] S. Golzari and M. R. Meybodi, "A Maze Routing Algorithm Based on Two Dimensional Cellular Automata," *Cellular Automata (ACRI 2006)*, Perpignan, France, (S. El Yacoubi, B. Chopard and S. Bandini, eds.), Berlin, Heidelberg: Springer, 2006 pp. 564–570. doi:10.1007/11861201_65.

[13] M. A. I. Tsompanas, G. C. Sirakoulis and A. Adamatzky, "Cellular Automata Models Simulating Slime Mould Computing," *Advances in Physarum Machines: Sensing and Computing with Slime Mould* (A. Adamatzky, ed.), Cham, Switzerland: Springer, 2016 pp. 563–594. doi:10.1007/978-3-319-26662-6_27.

[14] N. Fatès, I. Marcovici and S. Taati, "Cellular Automata for the Self-Stabilisation of Colourings and Tilings," *Reachability Problems (RP 2019)*, Brussels, Belgium, (E. Filiot R. Jungers and I. Potapov, eds.), Cham, Switzerland: Springer, 2019 pp. 121–136. doi:10.1007/978-3-030-30806-3_10.

[15] B. Benmedjdoub, E. Sopena and I. Bouchemakh, "2-Distance Colorings of Integer Distance Graphs," HAL Archives. (Mar 17, 2021) hal.archives-ouvertes.fr/hal-01279943.