Cellular Automata Preimages
Count and List Algorithm

Iztok Jeras[1] and Andrej Dobnikar[2]

[1] `http://www.rattus.info/al/al.html`
`iztok.jeras@rattus.info`
[2] University of Ljubljana, Faculty of Computer and Information Science,
Trzaska cesta 25, SI-1001 Ljubljana, Slovenia
`andrej.dobnikar@fri.uni-lj.si`

Abstract. Preimages of cellular automata are observed. Their number is computed using simple matrix operations. Three algorithms for making a list of preimages are graphically presented using the de Bruijn diagram and its concatenated form: the preimage network.

1 Introduction

Counting of preimages of one-dimensional cellular automata (CA) has been studied by Jen [1] in around 1989, Woorhees [2] in 1993, and McIntosh [7, 8] in 1993. This paper is build on slightly modified methods defined by McIntosh.

The de Bruijn diagram describes the overlapping of strings. It is the main tool for observing CA preimages, but it can graphically represent only preimages of a single cell. In this paper the de Bruijn diagram is firstly redrawn into the preimage diagram, which is then concatenated into the preimage network. This network graphically represents preimages of whole configuration strings.

The listing of preimages is less studied than their counting. In 1992 Wuensche [3–5] informally described the first algorithm; in this paper it is called the Trace and Backtrack (TB) algorithm. In 2004 Mora, Juárez and McIntosh [6] described a different algorithm that uses the subset diagram (SD) of the de Bruijn diagram. This paper introduces a third method; the Count and List (CL) algorithm. All algorithms are described graphically using the preimage network.

This paper provides only a few specific examples. Documentation and software for a more general and detailed approach can be found in [9].

2 Formal CA Definition

All examples in this paper are based on the rule 110 elementary (2,1) CA (Wolfram notation for closest neighbor boolean CA), so the formal definition focuses on this special case. Only finite CA are observed, since infinite CA may have an infinite number of preimages.

One-dimensional CA are arrays of cells. The value of each cell c_x at position x is either 0 or 1. The state of an array of cells is represented by a *string*

$\alpha = \ldots c_{x-1}c_xc_{x+1}\ldots$. The string of states of all cells of a finite CA of length N is a *configuration* $C = c_0c_1\ldots c_{N-1}$. The state of the CA at time t is represented by the configuration C^t. In all examples in this paper the so called *ether configuration* $\alpha = 00010011011111_2$ is used (the most common background pattern in rule 110). The subscribed number 2 is used to denote binary strings.

The *neighborhood* $n_x = c_{x-1}c_xc_{x+1}$ of the observed cell is composed of the observed cell and its two closest neighbors (Fig. 1). Neighborhoods of any pair of adjacent cells $c_{x-1}c_x$ overlap for the length of 2 cells. The *overlap* $o_x = c_{x-1}c_x$ uses the position index x from the right cell in the adjacent pair (Fig. 1). Another way to think about overlaps is to observe a cell c_x; then the *left overlap* is $o_x = c_{x-1}c_x$ and the *right overlap* is $o_{x+1} = c_xc_{x+1}$. Both the left and the right overlap are parts of the neighborhood $n_x = o_xc_{x+1} = c_{x-1}o_{x+1}$. Overlaps are used to represent the boundaries between adjacent segments in cell strings.

At each CA *evolution step* all cells from the current configuration C^t synchronously evolve into the future configuration C^{t+1}. The evolution of every single cell c_x into its future value c_x^{t+1} is defined as the output of the *local transition function* (f), that takes as input the present neighborhood n_x^t of the observed cell at position x. The local transition function is commonly called the *rule*.

$$c_x^{t+1} = \mathrm{f}(n_x^t) = \mathrm{f}(c_{x-1}^t c_x^t c_{x+1}^t)$$

The evolution step of the whole CA is defined as the *global transition function* $C^{t+1} = \mathrm{F}(C^t)$. It is the application of the local transition function to all cells in the configuration simultaneously (Fig. 2).

$$\mathrm{F}(c_0c_1\ldots c_{N-1}) = \mathrm{f}(n_0)\mathrm{f}(n_1)\ldots\mathrm{f}(n_{N-1})$$

For rule 110 (01101110_2) the local transition function is defined by the next table.

$$\underbrace{111_2}_{0}\quad\underbrace{110_2}_{1}\quad\underbrace{101_2}_{1}\quad\underbrace{100_2}_{0}\quad\underbrace{011_2}_{1}\quad\underbrace{010_2}_{1}\quad\underbrace{001_2}_{1}\quad\underbrace{000_2}_{0}$$

A *cyclic configuration* is created by joining the left and the right boundary of a finite configuration length N. The position index $x_{\circlearrowleft} = x \mod N$ becomes a cyclic group of length N. The *cyclic boundary condition* is commonly used since it avoids the explicit definition of the boundary.

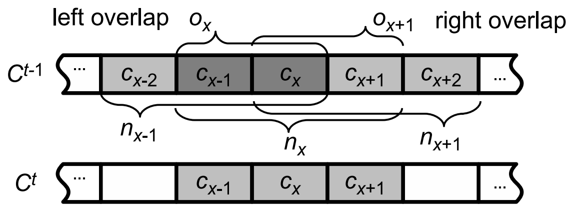

Fig. 1. Neighborhood and overlaps of the observed cell

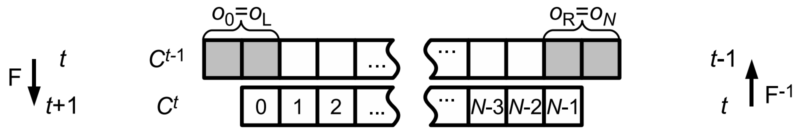

Fig. 2. Configuration boundaries

A *bounded configuration* is created by cutting a finite configuration length N from an infinite CA. At the left and the right boundary neighborhoods overstep the configuration by one cell (Fig. 2). To calculate a future configuration C^{t+1} this overstepping cells at both boundaries must be defined. The same problem occurs when calculating preimages, so the preimage C^{t-1} is 2 cells longer than the present configuration C^t. Boundaries are defined by overlaps $o_0 = o_\mathrm{L}$ for the left boundary and $o_N = o_\mathrm{R}$ for the right boundary.

3 Observing the Past

The *preimages* of a single cell c_x^t are *locally valid neighborhoods* n_x^{t-1} that are mapped into the observed cell value by the local transition function (Fig. 1). The inverse of the local transition function is defined as:

$$\mathrm{f}^{-1}(c_x^t) = \{n_x^{t-1} \in \{0,1\}^3 \mid \mathrm{f}(n_x^{t-1}) = c_x^t\} \ .$$

Preimages C^{t-1} of the present configuration C^t are past configurations that are mapped into the present configuration by the global transition function (Fig. 2). The inverse of the global transition function is:

$$\mathrm{F}^{-1}(C^t) = \{C^{t-1} \in \{0,1\}^N \mid \mathrm{F}(C^{t-1}) = C^t\} \quad \text{, for cyclic configurations, and}$$

$$\mathrm{F}^{-1}(C^t) = \{C^{t-1} \in \{0,1\}^{N+2} \mid \mathrm{F}(C^{t-1}) = C^t\} \quad \text{, for bounded configurations.}$$

Local preimages must overlap correctly to form global preimages. The de Bruijn diagram and the preimage network are graphical representations of overlapping neighborhoods. The first represents preimages of single cells the second preimages of cell strings.

3.1 The de Bruijn Diagram

The *de Bruijn diagram* represents preimages of a single present cell c^t, which are past neighborhoods n^{t-1} (Fig. 3). It is composed of 4 nodes, one for each of the distinct overlaps, and 8 directed links, one for each of the distinct neighborhoods. Nodes are drawn twice and arranged into two identical columns (from overlap 00_2 at the top to overlap 11_2 at the bottom). The two columns can be seen as overlaps (boundaries) at the left and right side of an observed cell (Fig. 1). Directed links connect source nodes o_s^{t-1} (left overlaps) to drain nodes o_d^{t-1} (right

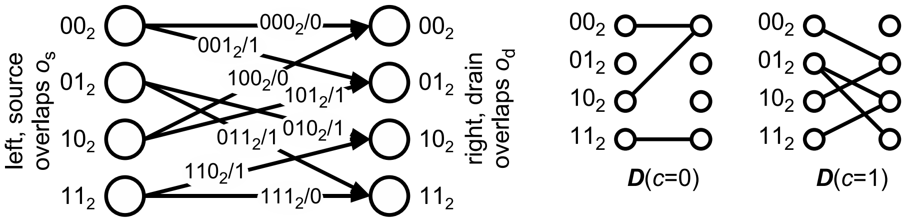

Fig. 3. The de Bruijn diagram (left) and two derived preimage diagrams (right)

overlaps). Links represent neighborhoods $n_{\mathrm{sd}}^{t-1} = o_{\mathrm{s}}^{t-1} c_{\mathrm{d}}^{t-1} = c_{\mathrm{s}}^{t-1} o_{\mathrm{d}}^{t-1}$ that can be decomposed into the source or the drain overlap. Links are labeled with the pair n^{t-1}/c^t, where $c^t = \mathrm{f}(n^{t-1})$ is defined by the local transition function.

The de Bruijn diagram is decomposed into two *preimage diagrams*, one for each of the available cell values. Only *locally valid neighborhoods* (those that are mapped into the observed cell by the local transition function $c^t = \mathrm{f}(n^{t-1})$) are allowed to appear in the preimage diagram. All invalid links are removed.

Example: The neighborhood $n^{t-1} = 101_2$ (link) is decomposed into the left overlap 10_2 (source node) and right overlap 01_2 (drain node). Since the neighborhood is translated into $c^t = 0$ its link is labeled $101_2/0$. After the decomposition into preimage diagrams this link becomes part of the diagram for $c = 0$.

The topological matrix of preimage diagrams is used for counting preimages.

Definition 1. *The* single cell preimage matrix $\boldsymbol{D}(c)$ *represents preimages of an observed single present cell with value c. It is a square of* 4×4 *elements, one for each source-drain overlap pair* $(o_{\mathrm{s}}, o_{\mathrm{d}})$*. The matrix element* $d_{o_{\mathrm{s}},o_{\mathrm{d}}}$ *is* 1 *if first: a past neighborhood* $n = o_{\mathrm{s}} c_{\mathrm{d}} = c_{\mathrm{s}} o_{\mathrm{d}}$ *exists that can be constructed as a link from the source overlap* o_{s} *to the drain overlap* o_{d}*; and second: n is a locally valid neighborhood* $\mathrm{f}(n) = c$*. Else the matrix element is* 0.

$$\boldsymbol{D}(0) = \begin{bmatrix} 1\,0\,0\,0 \\ 0\,0\,0\,0 \\ 1\,0\,0\,0 \\ 0\,0\,0\,1 \end{bmatrix} \qquad \boldsymbol{D}(1) = \begin{bmatrix} 0\,1\,0\,0 \\ 0\,0\,1\,1 \\ 0\,1\,0\,0 \\ 0\,0\,1\,0 \end{bmatrix}$$

The definition of the *single cell preimage matrix* is a special case of the definition of the *cell string preimage matrix*, where $|\alpha| = 1$.

3.2 Preimage Network

Present cells are aligned into a string $\alpha = c_0 c_1 \ldots c_{N-1}$ (Fig. 4 bottom). The preimage network of α (Fig. 4 top) is constructed by aligning preimage diagrams for each of the cells in the string side by side in the same order as in the string. Touching node pairs at the same height from two adjacent diagrams are merged into single nodes combining the diagrams into a network. Nodes at the extreme left and extreme right represent the boundaries of the preimage network.

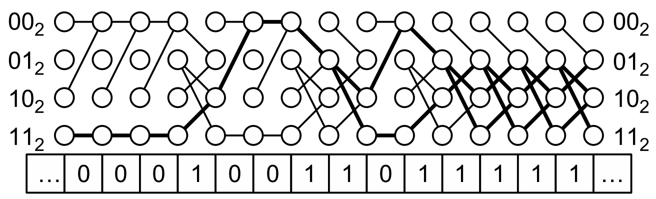

Fig. 4. Preimage network for the bounded ether conf. (globally valid paths are bold)

Distinct preimages have distinct representations as paths in the network. Each preimage C^{t-1} of a present configuration C^t is a *globally valid path* that must begin at the left boundary, pass a single link for each present cell, and end at the right boundary. For cyclic configurations the boundaries are connected, so paths must begin and end at the same overlap $o_\mathrm{L} = o_\mathrm{R}$. A preimage matrix can be defined to describe the preimage network.

Definition 2. $\boldsymbol{D}(\alpha)$ *is the* cell string preimage matrix *of the observed present string α of length $|\alpha| = N \geq 0$. Elements d_{o_L,o_R} in the preimage matrix represent the number of preimages (distinct paths in the network) that begin at an overlap o_L at the left and end at an overlap o_R at the right boundary (Fig. 2 and 4).*

The matrix of an empty string $\alpha = \varepsilon$, $|\alpha| = 0$, is an identity matrix $\boldsymbol{D}(\varepsilon) = I$.

Theorem 1. *The cell string preimage matrix $\boldsymbol{D}(\alpha)$ of the string $\alpha = c_0c_1 \dots c_{N-1}$ is the product of the chain of single cell preimage matrices $\boldsymbol{D}(c_x)$.*

$$\boldsymbol{D}(\alpha) = \prod_{x=0}^{N-1} \boldsymbol{D}(c_x) = \boldsymbol{D}(c_0)\boldsymbol{D}(c_1)\cdots\boldsymbol{D}(c_{N-1})$$

The theorem's proof is a simple induction on the length of the string. The following preimage matrix describes the ether configuration string.

$$\boldsymbol{D}(\alpha = 00010011011111_2) = \begin{bmatrix} 0\,0\,0\,0 \\ 0\,0\,0\,0 \\ 0\,0\,0\,0 \\ 0\,2\,3\,2 \end{bmatrix}$$

4 Counting Preimages

The number of preimages for bounded configurations $p_{\mathrm{L}\leftrightarrow\mathrm{R}}$ is computed by applying boundary vectors $\boldsymbol{b}_\mathrm{L}$ and $\boldsymbol{b}_\mathrm{R}$ (unrestricted boundaries in the example) to the preimage matrix of the observed string $\boldsymbol{D}(\alpha)$.

$$p_{\mathrm{L}\leftrightarrow\mathrm{R}}(\alpha = 00010011011111_2) = \boldsymbol{b}_\mathrm{L}\,\boldsymbol{D}(\alpha)\,\boldsymbol{b}_\mathrm{R}^\mathrm{T} = \begin{bmatrix}1\;1\;1\;1\end{bmatrix} D(\alpha) \begin{bmatrix}1\;1\;1\;1\end{bmatrix}^\mathrm{T} = 7$$

The number of preimages for cyclic configurations $p_\circlearrowleft$ is the sum of diagonal elements in $\boldsymbol{D}(\alpha)$. This are preimages that begin and end with the same overlap.

$$p_\circlearrowleft(\alpha = 00010011011111_2) = 2$$

5 Listing Preimages

5.1 Trace and Backtrack Algorithm

The TB algorithm traces the preimage network in the same way as the wall follower algorithm [10] for solving mazes (Fig. 5). The algorithm starts tracing at left boundary nodes (start-points). At each fork the leftmost (uppermost) path is taken first. If the algorithm reaches a dead-end, it backtracks the traced path to the last fork. Each time the right boundary is reached, the traced path is written into the preimage stack. When all the start-points are exhausted, the algorithm ends, and the stack contains the list of all globally valid preimages.

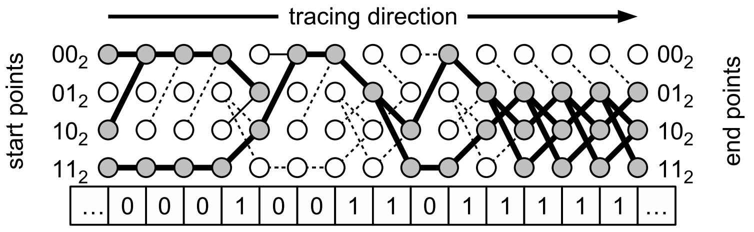

Fig. 5. Tracing and backtracking path on the network of the bounded ether conf.

5.2 Subset Diagram Algorithm

The idea is to trace only paths that do not lead to dead-ends, this are exactly the paths that can be reached from the right boundary (Fig. 6). The network is analyzed (searched for traceable paths) from the right and then traced from the left. Network analysis methods are the same as in the CL algorithm (described next), with the difference, that boolean multiplication (AND operator) is used.

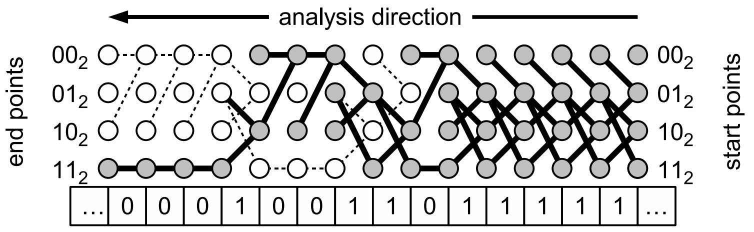

Fig. 6. Paths reachable from the right boundary

Globally valid paths connect both boundaries (Fig. 4), they are composed of links that can be reached from both the left and the right boundary (links bold on both Fig. 5 and Fig. 6).

5.3 Count and List Algorithm

A further modification to the algorithm is to count paths that can be traced from the right boundary instead of only checking if there are any. Path counters (Table 1) are computed iteratively for all position indexes, starting from counters at the right boundary. For bounded configurations counters are vectors $\boldsymbol{b}_x$, starting with the right boundary vector $\boldsymbol{b}_\mathrm{R}$ (unrestricted boundary in the example). For cyclic configurations counters must be matrices $\boldsymbol{D}_x$, starting with an identity matrix.

Bounded configurations:	Cyclic configurations:
$\boldsymbol{b}_N = \boldsymbol{b}_\mathrm{R} = [1,1,1,1]$	$\boldsymbol{D}_N = \boldsymbol{I}$
$\boldsymbol{b}_x = \boldsymbol{D}(c_x \dots c_{N-1})\,\boldsymbol{b}_\mathrm{R} = \boldsymbol{D}(c_x)\,\boldsymbol{b}_{x+1}$	$\boldsymbol{D}_x = \boldsymbol{D}(c_x \dots c_{N-1}) = \boldsymbol{D}(c_x)\,\boldsymbol{D}_{x+1}$
$\boldsymbol{b}_0 = \boldsymbol{D}(\alpha)\,\boldsymbol{b}_\mathrm{R}$	$\boldsymbol{D}_0 = \boldsymbol{D}(\alpha)$

Table 1. Counter vectors for the bounded ether configuration

o	counter vectors $\boldsymbol{b}_x$														
00	0	0	0	0	7	7	7	0	4	4	3	2	2	1	1
01	0	0	0	7	0	0	4	7	0	5	4	3	2	2	1
10	0	0	0	0	7	7	7	0	4	4	3	2	2	1	1
11	7	7	7	7	0	0	0	4	3	3	2	2	1	1	1
α	0	0	0	1	0	0	1	1	0	1	1	1	1	1	
x	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14

The computed counters can be presented on the preimage network as link widths (Fig. 7). Logarithmic widths can be used if counters become large.

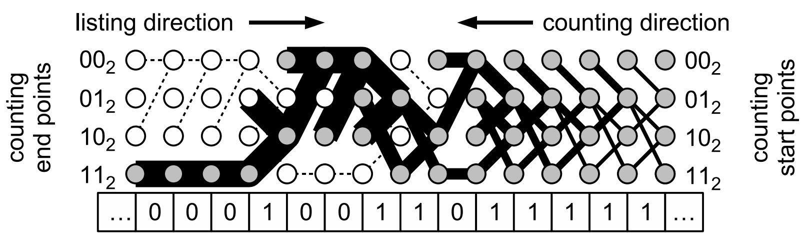

Fig. 7. CL algorithm on the preimage network of the bounded ether conf.

In the CL algorithm all preimages can be listed quasi simultaneously. The preimage path starts at the left boundary as a wide root, it forks into thiner branches (Fig. 8) and ends as single preimage leaves at the right boundary.

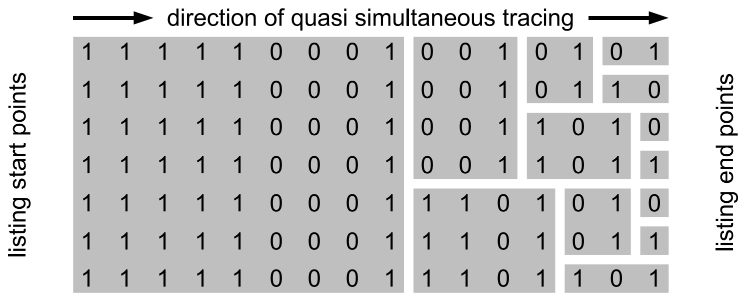

Fig. 8. The list of preimages of the bounded ether conf. as produced by the CL alg.

6 Conclusion

The purpose of the research described in the paper was to find a quantitative and qualitative measurement for information looses in irreversible CA. The research is not finished yet, but the logarithm of the number of preimages could be used as a quantitative measurement, and the preimage network could be used to observe how are looses spread in space. The source code for the TB an CL algorithms can be found at `http://www.rattus.info/al/al.html`.

References

1. Jen, E.: Enumeration of Preimages in Cellular Automata. Complex Systems **3** (1989) 421-456
2. Voorhees, B.: Predecessors of cellular automata states II. Pre-images of finite sequences. Physica D **73** (1993) 136-151
3. Wuensche, A., Lesser, M.: The Global Dynamics of Cellular Automata. Addison-Wesley (1992) `http://www.cogs.susx.ac.uk/users/andywu/gdca.html`
4. Wuensche, A.: Attractor Basins of Discrete Networks. Cognitive Science Research Paper 461. Univ. of Sussex. D.Phil thesis (1997) `ftp://ftp.cogs.susx.ac.uk/pub/users/andywu/papers/aw_thesis.pdf`
5. Wuensche, A.: Classifying Cellular Automata Automatically: Finding gliders, filtering, and relating space-time patterns, ... COMPLEXITY **4** (1999) 47-66 `ftp://ftp.cogs.susx.ac.uk/pub/users/andywu/papers/cplex.pdf`
6. Mora, J. C. S. T., Juárez, G., McIntosh, H. V.: Calculating ancestors in one-dimensional cellular automata. International Journal of Modern Physics C **15** (2004) 1151-1169
7. McIntosh, H. V.: Linear Cellular Automata Via de Bruijn Diagrams (1994) `http://delta.cs.cinvestav.mx/~mcintosh/newweb/marcodebruijn.html`
8. McIntosh, H. V.: Ancestors: Commentaries on The Global Dynamics of Cellular Automata by Andrew Wuensche and Mike Lesser (1993) `http://delta.cs.cinvestav.mx/~mcintosh/oldweb/wandl/wandl.html`
9. Jeras, I., Dobnikar, A.: Algorithms for Computing Preimages of Cellular Automata Configurations. Submited for publication to Physica D (2005) `http://www.rattus.info/al/al.html`
10. Pullen W. D.: Maze Algorithms. `www.astrolog.org/labyrnth/algrithm.htm`