Formal Logic of Cellular Automata

Sukanta Das
Department of Information Technology
Indian Institute of Engineering Science and Technology
Shibpur, 711103, India
sukanta@it.iiests.ac.in

Mihir K. Chakraborty
School of Cognitive Science
Jadavpur University, Kolkata, 700032, India
mihirc4@gmail.com

This paper develops a formal logic, named L_{CA}, targeting modeling of one-dimensional binary cellular automata. We first develop the syntax of L_{CA}, then give semantics to L_{CA} in the domain of all binary strings. Then the elementary cellular automata and four-neighborhood binary cellular automata are shown as models of the logic. These instances point out that there are other models of L_{CA}. Finally it is proved that any one-dimensional binary cellular automaton is a model of the proposed logic.

Keywords: cellular automata; formal logic; spatial rule; temporal rules; evolution; derivation

1. Introduction

Although histories of cellular automata (CAs) and formal logic are quite old, these two fields of mathematics and computer science did not meet much during their journey. Few works are found in the literature that relate formal logic and other kinds of automata; see for example [1] and a collected volume [2]. But such a relation between CAs and formal logic is yet to be adequately studied. In particular, whether the structure and dynamics of CAs can give birth to a new formal logic has not been previously investigated. This paper attempts to fulfill this need.

An effort has been taken in the recent past to study CAs using propositional logic [3, 4]. The authors of these works have studied one-dimensional binary CAs as models of propositional logic. On the contrary, this paper develops a different formal logic that we name L_{CA} and shows that the one-dimensional binary CAs are models of the proposed logic L_{CA}.

Section 2 introduces some definitions and terminologies related to CAs. The logic L_{CA} is developed in Section 3. Here we first develop

https://doi.org/10.25088/ComplexSystems.30.2.187

the syntax of L_{CA} and then give semantics of the logic L_{CA} in the domain of all binary strings. Section 4 proves that an elementary cellular automaton (ECA) is a model of the proposed logic, and evolution of the ECA is nothing but a derivation of the logic in the domain of binary strings. Another binary cellular automaton (CA), having four-neighborhood dependency, is also shown as a model of the logic (Section 4). Finally, we prove that any one-dimensional binary CA is a model of the logic.

2. Definitions and Terminology

A CA is a quadruple $(\mathcal{L}, \mathcal{S}, \mathcal{N}, f)$, where $\mathcal{L} \subseteq \mathbb{Z}^D$ is the D-dimensional cellular space, $\mathcal{S}$ is the finite set of states, $\mathcal{N} = (\overrightarrow{v_1}, \overrightarrow{v_2}, \ldots, \overrightarrow{v_m})$ is the neighborhood vector of m distinct elements of $\mathcal{L}$ that associates one cell to its *neighbors*, and $f : \mathcal{S}^m \to \mathcal{S}$ is called the local rule of the automaton [5]. During evolution, all the cells of a CA are updated simultaneously. Generally, the neighbors of a cell are the nearest cells surrounding the cell. However, when the neighborhood vector $\mathcal{N}$ is given, then the neighbors of a cell at location $\vec{v} \in \mathcal{L}$ are at locations $(\vec{v} + \overrightarrow{v_i}) \in \mathcal{L}$ for all $i \in \{1, 2, \ldots, m\}$. In this paper, nevertheless, we consider the following:

$$D = 1,\ \mathcal{L} = \mathbb{Z}/n\mathbb{Z},\ \mathcal{S} = \{0, 1\},$$
$$\mathcal{N} = \{-l_r, \ldots, -1, 0, 1, \ldots, r_r\}.$$

That is, the CAs under consideration are finite, having two states per cell, and each cell depends on itself and consecutive l_r number of left neighbors and consecutive r_r number of right neighbors. They are sometimes called left radius and right radius, respectively. Obviously, $m = l_r + r_r + 1$. A widely studied class of these CAs is the elementary cellular automata (ECAs) where $l_r = r_r = 1$.

A configuration of a binary CA is a mapping $c : \mathcal{L} \to \{0, 1\}$. Let us denote a set of all possible configurations of a CA of size n as $C_n = \{0, 1\}^{\mathcal{L}}$. So, $|C_n| = 2^n$, and for a configuration $x \in C_n$, $x = (x_0 x_1 \ldots x_{n-1})$. Local rules of the CA induce the global transition function $G : C_n \to C_n$, which satisfies the following condition: $y = G(x)$, $y \in C_n$, where $y = (y_i)_{i \in \mathcal{L}}$ and $y_i = f(x_{i-l_r}, \ldots, x_i, \ldots, x_{i+r_r})$. The local rules, especially for ECAs, are sometimes presented in tabular form (see Table 1), and the ECA rules are traditionally recognized by the decimal equivalent of the outputs of the eight (2^3) arguments of f. However, a CA rule can also be represented by a de Bruijn graph [6].

Arguments of f	111 (7)	110 (6)	101 (5)	100 (4)	011 (3)	010 (2)	001 (1)	000 (0)	Rule #
(i) f	0	0	0	0	1	0	1	0	10
(ii) f	0	1	0	1	1	0	1	0	90

Table 1. ECA rules 10 and 90.

Definition 1. Let Σ be a set of symbols and $s \geq 1$ be a number. Then, the de Bruijn graph is $B(s, \Sigma) = (V, E)$, where $V = \Sigma^s$ is the set of vertices, and $E = \{(ax, xb) \mid a, b \in \Sigma, x \in \Sigma^{s-1}\}$ is the set of edges.

Figure 1(a) shows $B(2, \{0, 1\})$, an example of a de Bruijn graph. This graph (i.e., $B(2, \{0, 1\})$) can be used to represent ECAs: the edges (ax, xb) represent the domain of the rule. Now, we need to label each edge by $f(a, x, b)$ to represent a CA with rule f. For clarity, however, we label each edge by "$axb \,/\, f(a, x, b)$." Figure 1(b) shows the de Bruijn graph for ECA rule 90 (see Table 1).

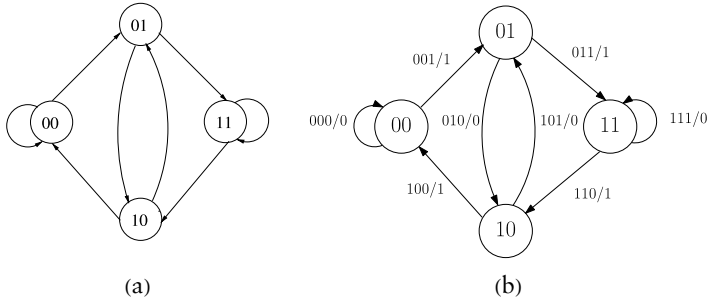

Figure 1. The de Bruijn graph of CA with rule 90. (a) The de Bruijn graph $B(2, \{0, 1\})$. (b) De Bruijn graph for rule 90.

Definition 2. A cycle of length n in a de Bruijn graph is a sequence of vertices $(v_1, v_2, \ldots, v_n, v_{n+1})$, where $v_{n+1} = v_1$ and $(v_i, v_{i+1}) = e_i$, $i \in \{1, 2, \ldots, n\}$. We generally represent this sequence as $(e_1, e_2, \ldots, e_n)$.

A cycle of length n in a de Bruijn graph corresponds to a configuration of an n-cell CA. For example, the configuration 001 111 corresponds to a cycle of length six in Figure 1. In fact, cycles in a de Bruijn graph and configurations are synonymous in this context. The successor of a configuration can be obtained from the de Bruijn graph by replacing an edge by its label (i.e., $f(a, x, b)$).

https://doi.org/10.25088/ComplexSystems.30.2.187

There are six elementary circuits in the de Bruijn graph of an ECA. Two of them are of length 1: (000) and (111), one is of length 2: (010, 101), two are of length 3: (001, 010, 100) and (011, 110, 101), and the rest are of length 4: (001, 011, 110, 100). Since cycles in a de Bruijn graph represent configurations of some automata, these elementary circuits also correspond to some configurations.

Definition 3. A configuration is called homogeneous if it corresponds to an elementary circuit of a de Bruijn graph.

Hence, there are six homogeneous configurations in ECAs. We generally represent these configurations by their minimal representations. That is, the length of such a representative homogeneous configuration is the length of its corresponding cycle in a de Bruijn graph. For ECAs, the homogeneous configurations are: **0**, **1**, **01**, **001**, **011** and **0011**. Obviously, all configurations of a CA are not homogeneous. An arbitrary configuration of an n-cell CA, which corresponds to a cycle of length n in the de Bruijn graph, is composed of some homogeneous configurations of smaller sizes. For example, the configuration 001 111 of a six-cell ECA is composed of two homogeneous configurations: 1 and 0011 (see Figure 1).

Definition 4. Two configurations are called shift equivalent to each other if and only if one can be obtained by left shifting the other.

For example, configurations 001 111 and 100 111 are shift equivalent to each other, because 001 111 is the one-bit left shift of 100 111. Obviously, shift equivalent configurations are composed of the same atomic configurations.

3. The Logic L_{CA}

A *logic* is mathematically defined as a pair $(\mathcal{L}, \vdash)$, where $\mathcal{L}$ is a set of sentences of the logic, also called *language*, and $\vdash$ is a *consequence relation*. To get the language $\mathcal{L}$, we need an alphabet and *formation rules* that form sentences using symbols of the alphabet. Members of $\mathcal{L}$ are called well-formed formulas (wffs). One method of getting the consequence relation ($\vdash$) is *natural deduction*, which uses a set of inference rules to reach a conclusion [7]. We adopt the natural deduction method for *derivation* of a wff from a given wff in the proposed L_{CA} logic. We next develop the syntax of L_{CA}.

3.1 Syntax

Let us consider a finite set of symbols: $P = \{p_0, p_1, \ldots, p_{N-1}\}$, where N is a natural number. These symbols are the members of the alphabet of our logic. Apart from them, there are four more symbols: σ, $*$,

) and (in the alphabet. Hence, the alphabet consists of

$$\sigma \quad * \quad) \quad (\quad p_0 \; p_1 \cdots p_{N-1}.$$

In the proposed logic, the members of P are *atomic* wffs. Since P is finite, the number of atomic wffs is also finite. Other (non-atomic) wffs are formed from the atomic wffs. We use other symbols of the alphabet to get the non-atomic wffs. Following are the formation rules of formulas.

1. All atomic wffs are wffs.
2. If α is a wff, then (α) and $\sigma(\alpha)$ are wffs.
3. If α and β are two wffs, then $\alpha * \beta$ and $\alpha\beta$ are wffs.
4. Nothing else is a well-formed formula.

Example 1. Suppose there are only two symbols in P. Then, p_0 and p_1 are the atomic wffs. The non-atomic wffs are (p_0), $\sigma(p_0)$, $\sigma(\sigma(p_0))$, $p_0 * p_1$, $(p_0 * p_1)$, $p_0 p_1$, $\sigma(p_0) * p_1$, $\sigma(p_0 * \sigma(p_1))$, $\sigma(p_0 \sigma(p_1))$ and so on.

The language of the logic $\mathcal{L}$ consists of the formulas formed by the given formation rules. If there is no ambiguity, we can drop brackets from a formula. We assume an abstract binary relation (R) over the set of all wffs (i.e., $R \subseteq \mathcal{L} \times \mathcal{L}$) as reflexive, symmetric and transitive. That is, R is an equivalence relation. In the language, as per definition, there are some formulas of the form $\underbrace{\sigma(\sigma(\cdots(\sigma}_{k \text{ times}}(\alpha)\cdots)$. We will represent this formula as $\sigma^k(\alpha)$. Similarly, a formula of the form $\underbrace{\alpha\alpha\cdots\alpha}_{k \text{ times}}$ is represented as α^k.

Definition 5. We say a formula is in normal form if it is in the form

$$\sigma^{k_1}\left(p'_1 * \sigma^{k_2}\left(p'_2 * \cdots \sigma^{k_l}(p'_l)\cdots\right)\right.$$

where $p'_1, p'_2, \ldots, p'_l \in P$ and $k_1, k_2, \ldots, k_l \geq 0$.

Let us now state the rules of the logic. In our logic, there are two classes of rules: one class is called *spatial* rules, and the other is called *temporal* rules, on account of their intended interpretation.

Spatial Rules

Following are the spatial rules of our logic.

- SR_1: $\frac{(\alpha)}{(\sigma^{k-i}(\sigma^i(\alpha)))}$

  where k and i are two natural numbers with $k \geq i$.

- SR_2: $\frac{(\alpha\beta)}{(\alpha)*(\beta)}$

  provided $R(\alpha, \beta)$ holds.

https://doi.org/10.25088/ComplexSystems.30.2.187

- SR_3: $\frac{(\alpha * \beta)}{(\alpha\beta)}$
- SR_4: $(1)\frac{(\alpha)}{\alpha}$, $(2)\frac{\alpha}{(\alpha)}$

The first rule introduces σ in a wff α by replacing α with $\sigma^{k-i}\big(\sigma^i(\alpha)\big)$. The second rule introduces $*$ in a wff. Similarly, $SR_4(2)$ introduces brackets in a wff. These rules, namely SR_1, SR_2 and $SR_4(2)$, are called *introduction rules*. On the other hand, SR_3 and $SR_4(1)$ eliminate $*$ and brackets, respectively. These are called *elimination rules*.

Temporal Rules

There is a set of n temporal rules to transform n atomic wffs.

- TR_i: For each $i \in \{0, 1, \ldots, N-1\}$
  $\frac{(p_i)}{(\alpha_i)}$
  where α_i is an arbitrary but fixed wff.

We can test whether a wff is *derivable* in the logic from a given wff using the spatial and temporal rules.

Definition 6. Derivation in the logic: For two wffs α and β, we say $\alpha \vdash \beta$ if and only if there is a sequence of wffs $\alpha_1(=\alpha)$, $\alpha_2, \ldots, \alpha_k(=\beta)$ such that α_i $(1 < i \leq k)$ is obtained from α_{i-1} by the application of any of the rules.

Example 2. Let us assume that α and β are two wffs. The following illustrates a derivation of β from α in this logic. The left column indicates the rule(s) applied on a previous wff to get the current one in the right column:

given wff	α
$SR_4(2)$	(α)
SR_2, assuming $\alpha \equiv \beta\gamma$ and $R(\beta, \gamma)$ holds	$(\beta) * (\gamma)$
SR_2, assuming $\beta \equiv p'_1 p'_1$ and $R(p'_1, p'_1)$ holds where $p'_1 \in P$, and SR_1	$(p'_1) * (p'_1) * \big(\sigma^3(\sigma(\gamma))\big)$
$SR_4(2)$	$(p'_1) * (p'_1) * \big(\sigma^3((\sigma(\gamma)))\big)$
SR_2, assuming $\sigma(\gamma) \equiv p'_2\gamma'$ and $R(p'_2, \gamma')$ holds where $p'_2 \in P$	$(p'_1) * (p'_1) * \big(\sigma^3((p'_2) * (\gamma'))\big)$
SR_1	$(p'_1) * (p'_1) * \big(\sigma^3\big((p'_2) * \big(\sigma(\sigma^2(\gamma'))\big)\big)\big)$ (1)
assuming $\sigma^2(\gamma') \equiv p'_3 \in P$	$(p'_1) * (p'_1) * \big(\sigma^3((p'_2) * (\sigma(p'_3)))\big)$
temporal rules	$(\alpha_1) * (\alpha_1) * \big(\sigma^3((\alpha_2) * (\sigma(\alpha_3)))\big)$

SR_3 and assuming $\alpha_2 \equiv p'_1$	$(\alpha_1) * (\alpha_1) * \left(\sigma^3((p'_1)(\sigma(\alpha_3)))\right)$
$SR_4(2)$	$((\alpha_1) * (\alpha_1)) * \left(\sigma^3((p'_1)(\sigma(\alpha_3)))\right)$
SR_3	$((\alpha_1)(\alpha_1))\left(\sigma^3((p'_1)(\sigma(\alpha_3)))\right)$
temporal rule	$((\alpha_1)(\alpha_1))\left(\sigma^3((\alpha_1)(\sigma(\alpha_3)))\right)$
$SR_4(1)$	$(\alpha_1)(\alpha_1)\sigma^3((\alpha_1)(\sigma(\alpha_3)))$
$SR_4(1)$	$\alpha_1\alpha_1\sigma^3(\alpha_1\sigma(\alpha_3))$
β	assuming $\beta \equiv \alpha_1\alpha_1\sigma^3(\alpha_1\sigma(\alpha_3))$

However, there may exist more than one derivation of a formula β from another formula α. Some of these derivations are in the *standard form* as defined next.

Definition 7. In a derivation, if the rules are applied in the following sequence and the normal form of the given wff is obtained at some steps, then the derivation is called a derivation in standard form.

- First use introduction rules.
- Obtain normal form.
- Use temporal rules.
- Use elimination rules.

Observe that the derivation of Example 2 is not a derivation in the standard form.

3.2 Semantics

Let us now give semantics to the wffs of the logic in a domain D of all binary strings except the empty string. Elements of D will be denoted as a, b and so forth where $a = (a_0a_1\ldots a_{j-1})$, $b = (b_0b_1\ldots b_{k-1})$ and $j, k \geq 1$ are the lengths of the strings. Members of P are some special binary strings. We interpret σ as a unary operator and $*$ as a binary operator in the domain D. We also define a relation, corresponding to R, in D. However, we use the same symbols for the operators and relations in the semantic domain.

Definition 8. Let $b = (b_0b_1\ldots b_{l-1})$ be a string of length $l \geq 1$. Then, we define $\sigma^0(b) = b$, and $\sigma^k(b) = \underbrace{\sigma\sigma\cdots\sigma}_{k \text{ number}}(b) = (b_ib_{i+1}\ldots b_{l-1}b_0\ldots b_{i-1})$ where $k \equiv i(\text{mod } l)$.

For example, if $b = 0\,110\,101\,011$, then $\sigma^2(b) = 1\,010\,101\,101$. Here the length of b is 10. Hence, $\sigma^{20}(b) = \sigma^0(b) = b$. Similarly, $\sigma^{21}(b) = \sigma^1(b) = 1\,101\,010\,110$.

Definition 9. Let us consider two binary strings $a = (a_0a_1\ldots a_{j-1})$ and $b = (b_0b_1\ldots b_{k-1})$ with $j, k \geq 1$. For two non-negative integers r and l,

https://doi.org/10.25088/ComplexSystems.30.2.187

the relation $R(a, b)$ holds if and only if the r-prefixes and the l-suffixes of the two sequences are equal. That is, $(a_0 a_1 \ldots a_{r-1}) = (b_0 b_1 \ldots b_{r-1})$ and $(a_{j-l} a_{j-l+1} \ldots a_j) = (b_{k-l} b_{k-l+1} \ldots b_k)$. Here the indices of a and b are the $(\mathrm{mod}\ j)$ and $(\mathrm{mod}\ k)$ indices, respectively.

Let us consider that $r = l = 1$. When $a = b = 0$ (or 1) then $R(a, b)$ holds. In fact, for any binary string a, $R(a, a)$ holds. If $a = 01\,011$ and $b = 0111$, then also $R(a, b)$ holds. But, if $a = 0$ and $b = 0111$, $R(a, b)$ does not hold. This relation shows the following properties.

- Reflexive: For any $a \in D$, $R(a, a)$ holds.
- Symmetric: For any $a, b \in D$, if $R(a, b)$ holds, then $R(b, a)$ also holds.
- Transitive: For any $a, b, c \in D$, if $R(a, b)$ and $R(b, c)$ hold, then $R(a, c)$ holds.

Hence, R is an equivalence relation. This relation is used to define the other operator $*$.

Definition 10. Let $a = (a_0 a_1 \ldots a_{j-1})$ and $b = (b_0 b_1 \ldots b_{k-1})$ be two strings. If $R(a, b)$ holds, then $(a * b) = (a_0 a_1 \ldots a_{j-1} b_0 b_1 \ldots b_{k-1})$. That is, $*$ is a partial operator.

To illustrate the functioning of this operator, assume that $l = r = 1$. Since $R(0, 0)$ holds, $0 * 0 = 00$. As another example, consider that $a = 0101$ and $b = 0011$. Then, $a * b = 01\,010\,011$ as $R(a, b)$ holds. However, we cannot use this $*$ operator for two arbitrary strings. For example, if $a = 1011$ and $b = 11\,000$, then $(a * b)$ is undefined, because $R(a, b)$ does not hold. This makes the $*$ a conditional concatenation operator.

Thus, the interpretation of the language of the logic is given in a structure $(D, \sigma, *)$ called the *semantic domain* of the language. Let $v : \mathcal{L} \mapsto D$ be a valuation function that interprets the wffs of the logic. For the atomic wffs, the following conditions are to be satisfied:

1. For any $p, p' \in P$, $v(p) \neq v(p')$ when $p \neq p'$.
2. For any $p, p' \in P$, $v(p) \neq \sigma^k(v(p'))$ for any $k \in \mathbb{N}$.
3. For any $p, p', p'' \in P$, $v(p) \neq v(p') * v(p'')$.

Example 3. Let $P = \{p_0, p_1, p_2, p_3\}$. The valuation to the atomic wffs as 0, 01, 001 and 010 cannot work, because $010 = \sigma(001)$. Another set of strings 0, 00, 01 and 001 also cannot be a valuation to the atomic wffs, because $00 = 0 * 0$. On the other hand, the set of strings $\{0, 01, 001, 011\}$ can be a valuation to P, because the above conditions are satisfied by $v(P)$.

The valuation function uses the following definition to interpret non-atomic wffs of the logic in D.

1. $\nu((\alpha)) = (\nu(\alpha))$
2. $\nu(\sigma(\alpha)) = \sigma(\nu(\alpha))$
3. $\nu(\alpha * \beta) = \nu(\alpha) * \nu(\beta)$
4. $\nu(\alpha\beta) = \nu(\alpha)\nu(\beta)$

Example 4. Let us consider $\alpha = ((p_1) * (p_1)) * \big(\sigma^3((p_2) * (\sigma(p_3)))\big)$. Assume that $\nu(p_1) = 01$, $\nu(p_2) = 0$ and $\nu(p_3) = 001$. Now, $\nu(\alpha)$ can be obtained in the following way:

$$\begin{aligned}
\nu(\alpha) &= \nu\Big(((p_1) * (p_1)) * \big(\sigma^3((p_2) * (\sigma(p_3)))\big)\Big) \\
&= \nu((p_1) * (p_1)) * \nu\big(\sigma^3((p_2) * (\sigma(p_3)))\big) \\
&= (\nu(p_1) * \nu(p_1)) * \Big(\nu\big(\sigma^3((p_2) * (\sigma(p_3)))\big)\Big) \\
&= (\nu(p_1) * \nu(p_1)) * \big(\sigma^3(\nu(p_2) * \nu(\sigma(p_3)))\big) \\
&= (\nu(p_1) * \nu(p_1)) * \big(\sigma^3(\nu(p_2) * \sigma(\nu(p_3)))\big) \\
&= \big(01 * 01\big) * \big(\sigma^3\big(0 * \sigma\big(001\big)\big)\big) \\
&= \big(0101\big) * \big(\sigma^3\big(0 * \sigma\big(001\big)\big)\big) \\
&= \big(0101\big) * \big(\sigma^3\big(0 * 010\big)\big) \\
&= \big(0101\big) * \big(\sigma^3\big(0010\big)\big) \\
&= \big(0101\big) * \big(0001\big) \\
&= 01\,010\,001
\end{aligned}$$

Depending upon the number of atomic wffs and the valuation function ν, there may be some strings that are not $\nu(\alpha)$ for any wff α. Let us now interpret the rules of the logic in D.

- SR_1: From the string $\nu(\alpha)$, the string $\nu\big(\sigma^{k-i}(\sigma^i(\alpha))\big)$ can be obtained, where k is the length of $\nu(\alpha)$ and $i \le k$ is a non-negative integer.
- SR_2: When $R(\nu(\alpha), \nu(\beta))$ holds, then $\nu(\alpha\beta)$ can be replaced by $\nu(\alpha) * \nu(\beta)$.
- SR_3: From the string $\nu(\alpha * \beta)$, the string $\nu(\alpha)\nu(\beta)$ can be obtained.
- SR_4: $\nu((\alpha))$ [resp. $\nu(\alpha)$] can be replaced by $\nu(\alpha)$ [resp. $\nu((\alpha))$].
- TR_i: For each $i \in \{0, 1, \ldots, N-1\}$, $\nu(p_i)$ can be replaced by $\nu(\alpha_i)$, where α_i is an arbitrary but fixed wff of the logic.

Similarly, we can interpret a derivation of the logic (see Definition 6) in D. A derivation of the logic in the domain D is the sequence $\nu(\alpha_1), \nu(\alpha_2), \ldots, \nu(\alpha_k)$ of strings such that the string $\nu(\alpha_i)$ $(1 < i \le k)$

https://doi.org/10.25088/ComplexSystems.30.2.187

is obtained from the string $\nu(\alpha_{i-1})$ by application of any of the given rules. In such a case, we say that the string $\nu(\alpha_k)$ is *semantically derived* from $\nu(\alpha_1)$.

Definition 11. For two wffs α and β, if $\nu(\alpha) \vdash \nu(\beta)$ for two semantic derivations, then the derivations are called equivalent to each other.

Example 5. Let us consider the derivation of Example 2. We interpret the formulas of the example in the semantic domain. Assume that $\nu(p_1) = 0$, $\nu(p_2) = 01$ and $\nu(p_3) = 001$ and let α_1, α_2 and α_3 be given values 01, 00 and 0010, respectively. That is, the temporal rules in the semantic domain are

$$\mathrm{TR}_1 : \frac{0}{01} \quad \mathrm{TR}_2 : \frac{01}{00} \quad \mathrm{TR}_3 : \frac{001}{0010}.$$

Consider that $\nu(\alpha) = 01\,010\,001$ and $\nu(\beta) = 0\,000\,100\,000$. Derivation of $\nu(\beta)$ is the following:

given wff	01010001
$\mathrm{SR}_4(2)$	(01010001)
SR_2	$(0101)*(0001)$
SR_2 and SR_1	$(01)*(01)*(\sigma^3(\sigma(0001)))$
definition of σ	$(01)*(01)*(\sigma^3(0010)))$
$\mathrm{SR}_4(2)$	$(01)*(01)*(\sigma^3((0010)))$
SR_2	$(01)*(01)*(\sigma^3((0)*(010)))$
SR_1	$(01)*(01)*(\sigma^3((0)*(\sigma(\sigma^2(010)))))$
definition of σ	$(01)*(01)*(\sigma^3((0)*(\sigma(001))))$ (2)
TR_2, TR_1 and TR_3	$(00)*(00)*(\sigma^3((01)*(\sigma(0010))))$
SR_3	$((00)*(00))*(\sigma^3((01)(\sigma(0010))))$
$\mathrm{SR}_4(2)$	$((00)*(00))*(\sigma^3((01)(\sigma(0010))))$
SR_3	$((00)(00))(\sigma^3((01)(\sigma(0010))))$
TR_2	$((00)(00))(\sigma^3((00)(\sigma(0010))))$
$\mathrm{SR}_4(1)$ and definition of σ	$0000\sigma^3((00)(0100))$
$\mathrm{SR}_4(1)$	$0000\sigma^3(000\,100)$
definition of σ	0000100000

Hence $01\,010\,001 \vdash 0\,000\,100\,000$. That is, $0\,000\,100\,000$ is derivable from $01\,010\,001$ in this logic. Observe that equation (2) is nothing but the interpretation of the derivation of Example 2 in D. There may be other examples following the same semantic derivation.

However, the derivation in equation (2) is not in the standard form. The following theorem states that it is possible to reduce $\nu(\alpha)$ into $\nu(\beta)$ where β is in normal form.

Theorem 1. For any wff α, $\nu(\alpha)$ is reducible to $\nu(\beta)$ where β is a wff in normal form.

Proof. If $\alpha = p$ or $\alpha = \sigma^k(p)$ for some $p \in P$ and $k \geq 1$, then the proof trivially follows.

Otherwise, we can identify $\nu(p)$ as a substring of $\nu(\alpha)$ for some $p \in P$. Then by suitably applying SR_1 and Definition 8, we can get $\nu(\alpha) = \sigma^{k_1}(\nu(\alpha'))$, where α' has $\nu(p)$ as its prefix. Since α' is obtained by using only formation rules and atomic wffs, we can write $\alpha = p * \alpha''$, where α'' is a wff. Hence, $\nu(\alpha) = \sigma^{k_1}(\nu(p) * \nu(\alpha''))$.

If $\alpha'' = p'$, then the proof directly follows. If $\alpha'' = \sigma^{k_2}(p')$ for some $p' \in P$ and $k_2 \geq 1$, then $\alpha = \sigma^{k_1}\big(\nu(p) * \nu\big(\sigma^{k_2}(p')\big)\big)$. Hence the proof follows. However, if α'' is another non-atomic wff, then using the given rationale, we can identify $\nu(p'')$ for some $p'' \in P$ in $\nu(\alpha'')$. Hence, we get

$$\nu(\alpha) = \sigma^{k_1}\big(\nu(p) * \nu\big(\sigma^{k_2}\big(\nu(p') * \sigma^{k_2}(\nu(p'') * \cdots)\big)\big)\big).$$

Now let us assume

$$\beta = \sigma^{k_1}\big(p * \sigma^{k_2}\big(p' * \sigma^{k_2}(p'' * \cdots)\big)\big).$$

Obviously $\nu(\alpha) = \nu(\beta)$. Here, β is in normal form. □

In the derivation of Example 5, we see that 01 010 001 is reducible to $(01) * (01) * \big(\sigma^3\big((0) * (\sigma(001))\big)\big)$. Considering $\gamma \equiv (p_1) * (p_1) * \big(\sigma^3((p_2) * (\sigma(p_3)))\big)$, we get from Example 5 that $\nu(\gamma) = \nu\big((01) * (01) * \big(\sigma^3\big((0) * (\sigma(001))\big)\big)\big)$. Here γ is in normal form. Although the derivation of Example 5 is not in standard form, we can slightly modify it to get a derivation in standard form. If we apply the temporal rules consecutively and use $SR_4(2)$ before them, then the resultant derivation becomes a derivation in standard form. In fact, for every derivation, there exists an equivalent derivation in standard form.

Proposition 1. For every semantic derivation in D, there exists an equivalent semantic derivation in the standard form.

4. Cellular Automata Are Models of the Logic

We begin this section with the following definition.

https://doi.org/10.25088/ComplexSystems.30.2.187

Definition 12. A derivation is called evolution if the derivation uses at least one temporal rule.

A *model* is an interpretation of a formal logic. This section shows that one-dimensional binary CAs are models of the logic L_{CA}. We first establish that ECAs are models of our logic, then extend the argument to show that four-neighborhood binary CAs are also models of the logic. Finally, we conclude that any binary CA is a model of the logic.

4.1 Elementary Cellular Automaton as a Model of the Logic

An ECA is a model of the logic L_{CA} via some valuation in the semantic domain $(D, \sigma, *)$. To show this, we have to interpret the semantic domain in the domain of ECAs. Table 2 is for that purpose.

The Logic L_{CA}	Semantic Domain	ECAs
—	0 and 1	states of a cell
—	l and r	left and right radii (where $l_r = r_r = 1$)
α	$\nu(\alpha) \in \{0, 1\}^n$	a configuration of size n
p_i, where $i = 0, 1, \ldots 5$	$\nu(p_i) = \{0, 1, 01, 001, 011, 0011\}$	elementary circuits of a de Bruijn graph
p_i^k	$\nu(p_i^k)$	a homogeneous configuration of size n where $n = k \times \lvert p_i \rvert$
$\sigma^k(\alpha)$	$\sigma^k(\nu(\alpha))$	configuration $\sigma^k(\nu(\alpha))$ is shift equivalent to α
$\alpha * \beta$ (when $R(\alpha, \beta)$ holds)	$\nu(\alpha) * \nu(\beta)$ (when $R(\nu(\alpha), \nu(\beta))$ holds)	$\nu(\alpha)\nu(\beta)$
$\alpha\beta$	$\nu(\alpha)\nu(\beta)$	$\nu(\alpha)\nu(\beta)$

Table 2. Interpretation of the semantic domain of L_{CA}.

Using the given interpretation, we get that the set of configurations C_n of an n-cell ECA is the valuation of some wffs of the logic. As the CAs are the dynamical systems that evolve with time, evolution of a CA can be understood as a derivation of the logic in the semantic domain. Such a derivation is an *evolution* in the logic.

Proposition 2. Evolution of an ECA is an evolution in the semantic domain of the logic L_{CA}.

Proof. To prove the theorem, we show that for a configuration $a \in C_n$, $a \vdash G(a)$ where $G(a) \in C_n$. Here a and $G(a)$ are two elements of the semantic domain $(D, \sigma, *)$. Observe that a configuration of

any length is composed with the elementary circuits of a de Bruijn graph (see Figure 1(a)). We also get a similar result from Theorem 1. So, we can write the following:

$$a = \sigma^{k_1}\big(\nu(p_i) * \sigma^{k_2}\big(\nu\big(p_j\big) * \cdots \sigma^{k_l}(\nu(p_k))\cdots\big)$$

where $\nu(p_i), \nu\big(p_j\big), \ldots, \nu(p_k) \in \{0, 1, 01, 001, 011, 0011\}$ and $k_1, k_2, \ldots, k_l \geq 0$. For a given ECA, let us define the temporal rules TR_i where

$$\nu(\alpha_i) = G(\nu(p_i))$$

for $i = 0, 1, \ldots, 5$. Now if we apply the temporal rules on the new format of a, we get the following string:

$$\sigma^{k_1}\big(\alpha_i * \sigma^{k_2}\big(\alpha_j * \cdots \sigma^{k_i}(\alpha_k)\cdots\big).$$

Next if we apply the elimination rules, we get a bit string, which is the next configuration of a. In our logic, this is a derivation, that is, $a \vdash G(a)$, where temporal rules are used. Hence, the next configuration of a configuration of an ECA is an evolution in the semantic domain of the logic. □

To illustrate the evolution of an ECA as an evolution in the semantic domain of the logic, we first need to fix the automaton. Let us consider ECA 90 from Table 1. For this rule, we need to fix TR_0 to TR_5 in the semantic domain, which are the following:

$$\mathrm{TR}_0: \frac{0}{0} \quad \mathrm{TR}_1: \frac{1}{0} \quad \mathrm{TR}_2: \frac{01}{00}$$

$$\mathrm{TR}_3: \frac{001}{110} \quad \mathrm{TR}_4: \frac{011}{011} \quad \mathrm{TR}_5: \frac{0011}{1111}$$

Now in the case of ECA 90 of size 16,

$$G(a) = 1\,110\,010\,001\,011\,010$$

when

$$a = 0\,011\,110\,101\,110\,001.$$

The following derivation shows that $G(a)$ can be obtained from a:

given wff	0011110101110001
$\mathrm{SR}_4(2)$	(0011110101110001)
SR_2	(00111101)*(01110001)
$\mathrm{SR}_4(2)$	((00111101))*((01110001))
SR_2	((001111)*(01))*((0111)*(0001))
SR_1	$((\sigma^3(\sigma^3(001\,111))) * (01)) * ((\sigma^2(\sigma^2(0111))) * (\sigma^3(\sigma(0001))))$
definition of σ	$((\sigma^3(111\,001)) * (01)) * ((\sigma^2(1101)) * (\sigma^3(0010)))$
SR_2	$((\sigma^3((11) * (1001))) * (01)) * ((\sigma^2((1) * (101))) * (\sigma^3((0) * (010))))$

https://doi.org/10.25088/ComplexSystems.30.2.187

SR_2	$((\sigma^3((1) * (1) * (1001))) * (01)) *$ $((\sigma^2((1) * (101)) * (\sigma^3((0) * (010))))$
SR_1	$\sigma^3(((1) * (1)) * \sigma^3(\sigma(1001))) * (01) * \sigma^2$ $((1) * (\sigma^2(\sigma(101)))) * \sigma^3((0) * (\sigma(\sigma^2(010))))$
definition of σ	$\sigma^3(((1) * (1)) * \sigma^3(0011)) * (01) *$ $\sigma^2((1) * (\sigma^2(011))) * \sigma^3((0) * (\sigma(001)))$
TR_0–TR_5	$\sigma^3(((0) * (0)) * \sigma^3(1111)) * (00) *$ $\sigma^2((0) * (\sigma^2(011))) * \sigma^3((0) * (\sigma(110)))$
SR_3	$\sigma^3(((0)(0))\sigma^3(1111))(00)\sigma^2((0)(\sigma^2(011)))\sigma^3((0)(\sigma(110)))$
$SR_4(1)$	$\sigma^3((00)\sigma^3(1111))(00)\sigma^2(0\sigma^2(011))\sigma^3(0\sigma(110))$
definition of σ	$\sigma^3(001\,111)(00)\sigma^2(0101))\sigma^3(0101)$
$SR_4(1)$, definition of σ	1 110 010 001 011 010

Therefore, $a \vdash G(a)$ in the semantic domain of our logic. This is true for any $a \in C_n$ and for any $n \geq 1$. Hence the proposed logic can express the evolution of the ECA.

4.2 Four-Neighborhood Binary Cellular Automaton: Another Model

We now show that another class of binary CAs where $l_r = 1$ and $r_r = 2$ are also models of the logic L_{CA}. In this class of CAs, each cell depends on the present states of its immediate left neighbor, two right neighbors and of itself to go to its next state. For such CAs, we consider a valuation function such that $v(P)$ is the following:

$$v(P) = \{0,\ 1,\ 01,\ 001,\ 011,\ 0001,\ 0011,\ 0111,\ 00\,011,\ 00\,111,\ 001\,101,\ 000\,111,\ 001\,011,\ 0\,001\,101,\ 0\,010\,111,\ 0\,001\,011,\ 0\,011\,101,\ 00\,011\,101,\ 00\,010\,111\}.$$

We give a meaning similar to the one we have given for ECAs to the symbols and others of the proposed logic in the domain of these CAs. The only exceptions here are $l = l_r = 1$, $r = r_r = 2$ and the number of elements of $v(P)$ is 19. We can see a similar result, noted below, for these CAs.

Proposition 3. Evolution of a one-dimensional four-neighborhood binary CA is an evolution in the semantic domain of the logic.

We omit the proof, as it is very similar to that of Proposition 2. We next show an example derivation in support of the result. For that, let us take a CA with the following rule:

1111	1110	1101	1100	1011	1010	1001	1000	0111	0110
1	0	1	1	1	1	0	1	1	0

0101	0100	0011	0010	0001	0000
1	0	0	1	1	0

For this CA, there are 19 temporal rules, as the cardinality of P is 19. We next fix the temporal rules TR_0 to TR_{18} for the CA:

$$\begin{array}{lllllllll}
\mathrm{TR}_1\colon & \frac{0}{0} & \mathrm{TR}_2\colon & \frac{1}{1} & \mathrm{TR}_3\colon & \frac{01}{11} & \mathrm{TR}_4\colon & \frac{001}{010} \\
\mathrm{TR}_5\colon & \frac{011}{101} & \mathrm{TR}_6\colon & \frac{0001}{1110} & \mathrm{TR}_7\colon & \frac{0011}{0001} & \mathrm{TR}_8\colon & \frac{0111}{1101} \\
\mathrm{TR}_9\colon & \frac{00\,011}{11\,001} & \mathrm{TR}_{10}\colon & \frac{00\,111}{00\,101} & \mathrm{TR}_{11}\colon & \frac{001\,101}{000\,110} & \mathrm{TR}_{12}\colon & \frac{000\,111}{110\,101} \\
\mathrm{TR}_{13}\colon & \frac{001\,011}{011\,101} & \mathrm{TR}_{14}\colon & \frac{0\,001\,101}{1\,100\,110} & \mathrm{TR}_{15}\colon & \frac{0\,010\,111}{0\,111\,101} & \mathrm{TR}_{16}\colon & \frac{0\,001\,011}{1\,111\,101} \\
\mathrm{TR}_{17}\colon & \frac{0\,011\,101}{0\,010\,110} & \mathrm{TR}_{18}\colon & \frac{00\,011\,101}{11\,010\,110} & \mathrm{TR}_{19}\colon & \frac{00\,010\,111}{11\,111\,101} & &
\end{array}$$

Now consider a configuration $a = 0\,011\,110\,101\,110\,001$ of size 16 (this was used in Derivation 2), and for the given CA, $G(a) = 0\,011\,011\,111\,011\,110$. The following steps show that $a \vdash G(a)$:

$$\begin{array}{ll}
\text{given wff} & 0011110101110001 \\
\mathrm{SR}_4(2) & (0011110101110001) \\
\mathrm{SR}_2 & (001111010111)*(0001) \\
\mathrm{SR}_1 & (\sigma^9(\sigma^7(001\,111\,010\,111))) * (0001) \\
\text{definition of } \sigma & (\sigma^9(111\,010\,111\,001)) * (0001) \\
\mathrm{SR}_4(2) & (\sigma^9((111\,010\,111\,001))) * (0001) \\
\mathrm{SR}_2 & (\sigma^9((111\,010) * (111\,001))) * (0001) \\
\mathrm{SR}_1, \mathrm{SR}_2 & (\sigma^9((\sigma^3(\sigma^3(111\,010))) * ((1) * (11\,001)))) * (0001) \\
\text{definition of } \sigma & (\sigma^9((\sigma^3(010\,111)) * ((1) * (11\,001)))) * (0001) \\
\mathrm{SR}_1 & (\sigma^9((\sigma^3((01) * (0111))) * ((1) * (\sigma^3(\sigma^2(11\,001)))))) * \\
 & \quad (0001) \\
\text{definition of } \sigma & (\sigma^9((\sigma^3((01) * (0111))) * ((1) * (\sigma^3(00\,111)))) * (0001) \\
\mathrm{TR}_3, \mathrm{TR}_8, \mathrm{TR}_2, & (\sigma^9((\sigma^3((11) * (1101))) * ((1) * (\sigma^3(00\,101)))) * (1110) \\
\quad \mathrm{TR}_{10}, \mathrm{TR}_6 & \\
\mathrm{SR}_3 & (\sigma^9((\sigma^3((11)(1101)))((1)(\sigma^3(00\,101))))(1110) \\
\mathrm{SR}_4(1) & \sigma^9(\sigma^3(111\,101)(1\sigma^3(00\,101)))1110 \\
\text{definition of } \sigma, & \sigma^9(101\,111\,101\,001)1110 \\
\quad \mathrm{SR}_4(1) & \\
\text{definition of } \sigma & 0011011111011110
\end{array} \tag{3}$$

Therefore, evolution of the CA is a derivation of the proposed logic.

https://doi.org/10.25088/ComplexSystems.30.2.187

4.3 A General Result

We have shown that two types of binary CAs, ECAs and four-neighborhood CAs, are the models in the semantic domain of the logic L_{CA}. Indeed, this logic can model any binary CA under the interpretation of symbols given in the previous two subsections. Following is an important result of our work.

Theorem 2. Any one-dimensional binary CA is a model in the semantic domain of L_{CA}.

Proof. Given a one-dimensional binary CA, consider the number of elements in P as the number of elementary circuits in the corresponding de Bruijn graph and choose $v(P)$ as the homogeneous configurations with minimal length of the CA. Next consider that $l = l_r$ and $r = r_r$. Now, a configuration of size n corresponds to a cycle in the de Bruijn graph of length n (see Definition 2), which is obviously composed of one or more elementary circuits. This configuration is also an element in the semantic domain of the logic. So, using Theorem 1, a configuration a can be represented as the following:

$$a = \sigma^{k_1}\big(v(p_i) * \sigma^{k_2}\big(v\big(p_j\big) * \cdots \sigma^{k_l}(v(p_k))\cdots\big)$$

where $p_i, p_j, \ldots, p_k \in P$ and $k_1, k_2, \ldots, k_l \geq 0$.

Let us now define $\alpha_0, \alpha_1, \ldots$ of temporal rules considering

$$\alpha_i = G(p_i).$$

Now apply the temporal rules to transform it, and then use the elimination rules (SR_3 and $\mathrm{SR}_4(1)$) and definition of σ. The transformed a is then $G(a)$. That is, after giving an interpretation to the symbols and others in the domain of an arbitrary one-dimensional binary CA, we can get that its evolution is nothing but an evolution in the logic. Hence the proof. □

5. Conclusion

We have developed a logical language for one-dimensional binary cellular automata (CAs), called L_{CA}. After developing the syntax of the logic, we have given an interpretation in the domain of all binary strings, and then we have shown that binary CAs are the models of the logic. Although this work concentrates on finite CAs, the stated construction can easily be extended to the infinite CAs. This work can be further extended to show that any one-dimensional CA (binary and nonbinary) can be a model of the logic L_{CA}.

References

[1] N. Bedon, "Logic and Branching Automata," *Logical Methods in Computer Science*, **11**(4), 2015 pp. 1–38. doi:10.2168/LMCS-11(4:2)2015.

[2] J. Flum, E. Gradel and T. Wilke (eds.), *Logic and Automata: History and Perspectives*, Amsterdam: Amsterdam University Press, 2008.

[3] T. Ishida, S. Inokuchi and Y. Kawahara, "Cellular Automata and Formulae on Monoids," *Cellular Automata (ACRI 2014)* (J. Wąs, G. C. Sirakoulis and D. Bandini, eds.), Cham, Switzerland: Springer, 2014 pp. 55–64. doi:10.1007/978-3-319-11520-7_7.

[4] T. Ishida, S. Inokuchi and Y. Kawahara, "Propositional Logic and Cellular Automata on Monoids," *Journal of Cellular Automata*, **12**(1–2), 2016 pp. 27–45.

[5] K. Bhattacharjee, N. Naskar, S. Roy and S. Das, "A Survey of Cellular Automata: Types, Dynamics, Non-uniformity and Applications," *Natural Computing*, **19**, 2018 pp. 433–461. doi:10.1007/s11047-018-9696-8.

[6] K. Sutner, "De Bruijn Graphs and Linear Cellular Automata," *Complex Systems*, **5**(1), 1991 pp. 19–30. content.wolfram.com/uploads/sites/13/2018/02/05-1-3.pdf.

[7] D. Prawitz, *Natural Deduction: A Proof-Theoretical Study*, Stockholm: Almqvist & Wiksell, 1965.