

Simulating Boolean Circuits with Tissue P Systems

Vaka Jaya PRAKASH, Kamala KRITHIVASAN

Department of Computer Science and Engineering
Indian Institute of Technology Madras
Chennai - 600036, India
prakash@cs.iitm.ernet.in
kamala@iitm.ernet.in

Abstract

We propose a model for simulation of Boolean circuits with tissue P systems. The simulation is done in two steps; first we simulate basic building blocks (i.e., AND, OR, and NOT gates) of Boolean circuits and next show the simulation when these gates can be combined to form actual circuits. We consider only *non-cooperative* rules (the case of using cooperative rules is obvious), two types of processing modes viz., *minimal* and *parallel*, and all types of transmitting modes, i.e., *one*, *replicative*, and *spread*.

1 Introduction

P systems [8], an area of active current research, is motivated from the structure of the cell and functioning of membranes. The three fundamental features of the cell which are used in membrane computing model are the *membrane structure*, the *objects*, and the *evolution rules*. In a cell, objects can be considered as being *atomic* as in the case of *P systems with symbol-objects* or they can be associated with a structure, as in the case of DNA molecules, which can be described by a string. This leads one to consider P systems with string-objects.

One common feature in all variants of P systems is the membrane structure, which can be represented as a tree like structure. A new way of looking at P systems was proposed in *tissue P systems (tP systems)* [6], where the membrane structure can be represented by a graph. Tissue P systems are based on the ideas of inter-cellular communication and the way the neurons cooperate and process impulses in a complex net established by *synapses* [1]. Tissue P systems consist of several *cells* which are related by protein channels. Each cell has a state from a given finite set and can process multisets of objects, represented by symbols from a given alphabet. The rules are of the form $sM \rightarrow s'M'$, where s, s' are states and M, M' are multisets of objects. A single rule can be applied to one occurrence of M (the *minimal* mode), to all possible occurrences of M (the *parallel* mode), or we can apply a maximal package of rules of the form $sM_i \rightarrow s'M'_i$, $1 \leq i \leq k$ (that is, involving the same states s and s'), which can be applied to the current multiset (the *maximal* mode). The processed objects can be communicated to one of the neighboring cells (the

one mode), or to all the neighboring cells (the *replicative* mode), or to a subset of neighboring cells (the *spread* mode).

Boolean circuits are well known computing devices which incorporate features of parallelism. A model for simulating Boolean circuits with DNA algorithm is proposed in [7]. A model for simulating Boolean circuits using peptide computing is considered in [3]. A simple model for simulating Boolean circuits with P systems is proposed in [4]. There the membrane structure is as given in [8]. In this paper we try to simulate Boolean circuits using tissue P system [6]. For simulating Boolean circuits we define tissue P system same as in [6] except the output. In our model the output is just an object sent out from the output cell. The formal definition is given in Section 4. Here we consider *min* and *par* modes of processing multisets of objects and all three modes for transmitting the processed multisets. Our model is efficient in a sense that using this model we can simulate 1-input NOT gate, 2-input logical gates AND, OR, as well as n -input logical gates AND, OR, XOR using non-cooperative rules. We also show how Boolean circuits can be simulated.

In the next three sections we recall some prerequisites needed for the paper. In Section 5 first we consider non-cooperative rules in each of the minimal, parallel, and maximal modes for simulating NOT (\neg) gate. Subsequently, we consider non-cooperative rules in the minimal mode for simulating AND (\wedge), OR (\vee) gates. In Section 6 we simulate n -input basic gates using non-cooperative rules in parallel mode for processing multiset of objects. In Section 7 we simulate n -input basic gates using non-cooperative rules in the minimal mode for processing multisets of objects. Finally, we show how these basic gates can be combined together to form a required circuit.

2 Some Mathematical Prerequisites

The set of natural numbers is denoted by N . A multiset over a set X is a mapping $M : X \rightarrow N$; for $a \in X$, we say that $M(a)$ is the multiplicity of a in M . The set $\text{supp}(M) = \{a \in X | M(a) > 0\}$ is called the support of M . Here we work only with multisets over finite sets X (implicitly, with multisets of a finite support). For a given alphabet V , we denote by V^* the language of all strings over V , including the empty string, denoted by λ . The length of $x \in V^*$ is denoted by $|x|$.

A multiset M over an alphabet V can be represented by a string $w \in V^*$ such that $\Psi_V(w)$ gives the multiplicities in M of the symbols from V ; obviously, all permutations of w are representations of the same multiset. This suggests to use strings as representations of multisets, thus, when we will say “the multiset $w \in V^*$ ” this means “the multiset represented by w ”. Clearly, the empty multiset, that with an empty support, is represented by the empty string, λ .

3 Boolean Circuits-Preliminaries

Boolean circuits are a formal model of the combinational logic circuit. The basic building blocks of Boolean circuits are logic gates such as AND, OR, NOT gates. The logic gates are interconnected together using wires (cables) in a specified manner to

obtain desired Boolean circuits. Consider the smallest Boolean algebra $B = \langle \{0, 1\}, \vee, \wedge, \oplus, \neg, 0, 1 \rangle$ with support $\{0, 1\}$, binary operations OR (\vee), AND (\wedge), XOR (\oplus), unary operator NOT (\neg), and null-ary operations (or constants) 0 and 1. Binary and unary operations on finite sets can be described by operation tables, depicting the result of the operation on each possible input. The operation tables for binary OR (\vee), AND (\wedge), XOR (\oplus) and unary NOT (\neg) are given in Figure 1.

\wedge	0	1
0	0	0
1	0	1

\vee	0	1
0	0	1
1	1	1

\neg	
0	1
1	0

\oplus	0	1
0	0	1
1	1	0

Figure 1: Value tables for AND (\wedge), OR (\vee), NOT (\neg), and XOR (\oplus).

Consider the set $B_k = \{f \mid f : \{0, 1\}^k \rightarrow \{0, 1\}\}$ of k -ary Boolean functions. Note that $0, 1 \in B_0$; $\neg \in B_1$; $\vee, \wedge, \oplus \in B_2$, and they are called the elementary Boolean functions. Note also that, because OR (\vee) operation is associative, one can define a ternary OR, $V^3 : \{0, 1\}^3 \rightarrow \{0, 1\}$ by $V^3(x_1, x_2, x_3) = ((x_1 \vee x_2) \vee x_3)$. This can be extended to an m -ary OR, $\vee^m : \{0, 1\}^m \rightarrow \{0, 1\}$, $\vee^m(x_1, x_2, \dots, x_m) = x_1 \vee x_2 \dots \vee x_m$. The same holds for the AND, XOR operations which can be extended to m -ary AND (\wedge^m), and XOR (\oplus^m) with $m \geq 2$.

Definition 3.1 A Boolean circuit $\alpha = (V, E, \lambda)$ is a finite directed acyclic graph (V, E) , having the set V of vertices, the set E of directed edges, and a vertex labeling function $\lambda : V \rightarrow \{I\} \cup \{\wedge, \vee, \oplus, \neg\}$, where I is a special symbol. A vertex $x \in V$ with $\lambda(x) = I$ has in-degree 0 and is called an input vertex. A vertex $y \in V$ with out-degree 0 is called an output vertex. Vertices with labels \wedge, \vee, \oplus have in-degree 2 and out-degree 1, while vertices with label \neg has in-degree 1 and out-degree 1.

The input of α is given by n -tuples $\langle x_1, x_2, \dots, x_n \rangle$ of distinct vertices, and the output by m -tuples $\langle y_1, y_2, \dots, y_m \rangle$. In circuit theory the vertices with labels other than I are called gates, the in-degree of a vertex is called the fan-in, and the out-degree the fan-out. Above, we have defined a circuit whose gates are labeled with the elementary Boolean functions $\{\wedge, \vee, \oplus, \neg\} \subseteq B_1 \cup B_2$. More generally, we can speak of circuits with vertex labeling $\lambda : V \rightarrow \{I\} \cup B_0 \cup \dots \cup B_k$. A vertex x with $\lambda(x) \in B_i$ has in-degree i .

Definition 3.2 A circuit α with input $\langle x_1, x_2, \dots, x_n \rangle$ and output $\langle y_1, y_2, \dots, y_m \rangle$ computes a function $f : \{0, 1\}^n \rightarrow \{0, 1\}^m$ in the following way: for every i , $1 \leq i \leq n$, to the input x_i is assigned a value $val(x_i) \in \{0, 1\}$ representing the i -th bit argument of the function. To every other vertex x is assigned the unique value $val(x) \in \{0, 1\}$ obtained by applying the operation $\lambda(x)$ to the values of the vertices incoming into x . The m -tuple $\langle val(y_1), \dots, val(y_m) \rangle$ is the value of function f , every output vertex y_j gives the j -th bit of the output.

For $m = 1$, the circuits will compute functions $f : \{0, 1\}^n \rightarrow \{0, 1\}$, and we will consider here only such circuits and such functions.

4 Tissue P System

We give below the definition from [6].

A tissue P system, of degree m , $m \geq 1$, is a construct

$$\Pi = (O, \sigma_1, \sigma_2, \dots, \sigma_m, syn, i_{out}),$$

where:

- O is a finite non-empty alphabet (objects);
- $syn \subseteq \{1, \dots, m\} \times \{1, \dots, m\}$ (synapses among cells);
- $i_{out} \in \{1, \dots, m\}$ indicates the *output cell*;
- σ_i , $1 \leq i \leq m$, are cells, where each $\sigma_i = (Q_i, s_i, w_i, P_i)$, with:
 - Q_i is a finite set of states;
 - $s_i \in Q_i$ is the initial state;
 - w_i is the initial multiset of objects;
 - P_i is a finite set of rules of the form $sw \rightarrow s'xy_{go}z_{out}$, where $s, s' \in Q_i, w, x \in O^*, y_{go} \in (O \times \{go\})^*$ and $z_{out} \in (O \times \{out\})^*$, with the restriction that $z_{out} = \lambda$ for all $i \in \{1, 2, \dots, m\}$ different from i_{out} .

A tP system as above is said to be cooperative if it contains atleast a rule $sw \rightarrow s'w'$ such that $|w| > 1$, and non-cooperative in the opposite case. The objects which appear in the left hand multiset w of a rule $sw \rightarrow s'w'$ are sometimes called impulses, while those in w' are called excitations.

According to Remark 1 in [6], rules of the forms $s \rightarrow s', s \rightarrow s'w'$ are considered to simulate various basic gates like AND, OR, XOR, and NOT. In this paper we use rules of the form $s\lambda \rightarrow s'w'$ for n input gates. This is the same as $s \rightarrow s'w'$ in [6].

Any m -tuple of the form (s_1w_1, \dots, s_mw_m) , with $s_i \in Q_i$ and $w_i \in O^*$, for all $1 \leq i \leq m$, is called a configuration of Π ; thus, $(s_{1,0}w_{1,0}, \dots, s_{m,0}w_{m,0})$ is the initial configuration of Π .

Using the rules from the sets $P_i, 1 \leq i \leq m$, we can define transitions among the configurations of the system. For this purpose we first consider three *modes of processing the impulse-objects* and three *modes of transmitting excitation-objects* from one cell to another cell.

Let us denote $O_{go} = \{(a, go) \mid a \in O\}, O_{out} = \{(a, out) \mid a \in O\}$, and $O_{tot} = O \cup O_{go} \cup O_{out}$. For $s, s' \in Q_i, x \in O^*, y \in O_{tot}^*$, we write

$$sx \Longrightarrow_{min} s'y \text{ iff } sw \rightarrow s'w' \in P_i, w \subseteq x, \text{ and } y = (x - w) \cup w';$$

$$sx \Longrightarrow_{par} s'y \text{ iff } sw \rightarrow s'w' \in P_i, w^k \subseteq x, w^{k+1} \not\subseteq x, \\ \text{for some } k \geq 1 \text{ and } y = (x - w^k) \cup w'^k;$$

$$sx \Longrightarrow_{max} s'y \text{ iff } sw_1 \rightarrow s'w'_1, \dots, sw_k \rightarrow s'w'_k \in P_i, k \geq 1, \text{ such that} \\ w_1 \dots w_k \subseteq x, y = (x - w_1 \dots w_k) \cup w'_1 \dots w'_k, \\ \text{and there is no rule } sw \rightarrow s'w' \in P_i \text{ such that} \\ w_1 \dots w_k w \subseteq x.$$

Three modes of processing multisets and three modes of transmitting the processed multisets from one cell to another cell are considered.

For processing multisets, *minimal* (*min*), parallel (*par*), and maximal (*max*) modes are used.

- In the *min* mode, only one occurrence of the left-hand side multiset of a rule is processed (replaced by the multiset from the right-hand side of the rule, at the same time changing the state of the cell).
- In the *par* mode, a maximal change is performed with respect to a chosen rule, in the sense that as many as possible copies of the multiset from the left hand of the rule are replaced by the corresponding number of copies of the multiset from the right hand side of the rule.
- In the *max* mode, change is performed with respect to all rules which use the current state of the cell and introduce the same new state after processing the multisets.

For transmitting the processed multisets, we use replicative (*repl*), *one*, and *spread* modes.

- In the *repl* mode, each processed multiset in a cell σ_i is sent to each of the cells σ_j such that $(i, j) \in syn$.
- In the *one* mode, each processed multiset in a cell σ_i is sent to one of the cells σ_j such that $(i, j) \in syn$, chosen nondeterministically.
- In the *spread* mode, each processed multiset in a cell is nondeterministically distributed among the cells σ_j such that $(i, j) \in syn$.

If we have at most two cells, then the three modes of transmitting the processed multisets from a cell to another cell coincide.

During any transition, some cells can do nothing: if no rule is applicable to the available multisets in the current state, a cell waits until new multisets are sent to it from its ancestor cells. It is also worth noting that each transition lasts one time unit, and that the work of the net is synchronized, the same clock marks the time for all cells.

A sequence of transitions among configurations of the system Π is called a computation of Π . A computation which ends in a configuration where no rule in no cell can be used is called an halting computation. The result of a halting computation is the object sent to the environment from the output cell i_0 .

We denote by $N_{(m,r)}\{\gamma, \alpha, \beta\}(\Pi)$, $\gamma \in \{Coo, nCoo\}$, $\alpha \in \{min, par, max\}$, $\beta \in \{repl, one, spread\}$, the object sent out by a *tP* system Π , in the mode (γ, α, β) , with m cells and r states.

5 Simulating NOT, AND, OR, and XOR Gates

In this section we simulate AND (\wedge), OR (\vee), and NOT (\neg) gates using non-cooperative rules. We take tissue P system with *min*, *max*, and *par* modes as

processing modes for multiset of objects, and *one*, *spread*, and *repl* modes as transmitting modes for processed multiset of objects. We consider that the input for a gate is a multiset and is given in a cell which is in initial state. The output is computed and sent to the outer region. Practically, the gate will act as a filter that receives two symbols synchronously, and after some computation steps, sends the result to the environment.

5.1 Simulation of NOT Gate

The NOT gate is also known as the “inverter”. It just switches the value that enters the gate into its complement value. We can do this with one cell and non-cooperative rules, as shown in Figure 4. Initially σ_1 contains any one the symbols from $\{0, 1\}$.

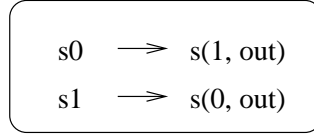


Figure 2: Simulation of NOT gate

The tP system is $\Pi = (O, \sigma_1, syn, i_{out})$, where:

- $O = \{0, 1\}$;
- $syn = \emptyset$;
- $i_{out} \in \{1\}$ indicates the *output cell*;
- $\sigma_1 = (Q, s, w, P)$, where
 - $Q = \{s\}$ is a finite set of states;
 - $s \in Q$ is the initial state;
 - $w \in \{0, 1\}$;
 - $P = \{s0 \rightarrow s(1, out), s1 \rightarrow s(0, out)\}$.

For a given input value from the set $\{0, 1\}$, $N_{(1,1)}\{ncoo, \alpha, \beta\}(\Pi)$ gives NOT value, for all $\alpha \in \{min, par, max\}, \beta \in \{repl, one, spread\}$.

5.2 Simulation of AND Gate

The following tissue P system with only one cell and non-cooperative rules gives the AND value of two (not necessarily synchronized) input values. Initially σ_1 contains any one of the multisets $\{00, 11, 10, 01\}$. The tP system is $\Pi = (O, \sigma_1, syn, i_{out})$, where:

- $O = \{0, 1\}$;
- $syn = \emptyset$;

- $i_{out} \in \{1\}$ indicates the *output cell*;
- $\sigma_1 = (Q, s, w, P)$, where
 - $Q = \{s, s', s''\}$ is a finite set of states;
 - $s \in Q$ is the initial state;
 - w is one of the multisets from the set $\{00, 11, 10, 01\}$;
 - P contains the following rules:
 1. $s0 \rightarrow s'\lambda$,
 2. $s1 \rightarrow s''\lambda$,
 3. $s'0 \rightarrow s(0, out)$,
 4. $s'1 \rightarrow s(0, out)$,
 5. $s''1 \rightarrow s(1, out)$,
 6. $s''0 \rightarrow s(0, out)$.

The system works as follows:

Initially σ_1 contains any of the multisets from the set $\{00, 11, 10, 01\}$. In all the following cases initially σ_1 is in state s .

Case 1: Assume that initially σ_1 contains multiset 00. The only applicable rule $s0 \rightarrow s'\lambda$ is applied to multiset 00, one 0 is erased and the state is changed to s' . Now the only applicable rule $s'0 \rightarrow s(0, out)$ is applied to 0, result 0 is sent out to the outer region and the state is changed to s .

Case 2: Assume that initially σ_1 contains multiset 11. The only applicable rule $s1 \rightarrow s''\lambda$ is applied to multiset 11, one 1 is erased and the state is changed to s'' . Now the only applicable rule $s''1 \rightarrow s(1, out)$ is applied to 1, result 1 is sent out to the outer region and the state is changed to s .

Case 3: Assume that initially σ_1 contains any one of the multisets from $\{10, 01\}$. Assume that 10 is present in σ_1 . Two rules 1, 2 are applicable to one of 10. If we apply rule 1, then 0 is erased and state is changed to s' and then rule 4 is applied to 1, result 0 is sent out to outer region. If we apply rule 2, then 1 is erased and state is changed to s'' and then rule 6 is applied to 0, result 0 is sent out to outer region. Similarly, for 01 we get result 1.

Thus, for any input multiset from set $\{00, 11, 10, 01\}$, $N_{(1,3)}ncoo, \alpha, \beta(\Pi)$ gives AND value, for all $\alpha \in \{min\}, \beta \in \{repl, one, spread\}$.

5.3 Simulation of OR Gate

The following tissue P system with only one cell and non-cooperative rules gives the OR value of two input values. Initially σ_1 contains any one of the multisets from the set $\{00, 11, 10, 01\}$. The tP system is $\Pi = (O, \sigma_1, syn, i_{out})$, where:

- $O = \{0, 1\}$;

- $syn = \emptyset$;
- $i_{out} \in \{1\}$ indicates the *output cell*;
- $\sigma_1 = (Q, s, w, P)$, where
 - $Q = \{s, s', s''\}$ is a finite set of states;
 - $s \in Q$ is the initial state;
 - w is one of the multisets from the set $\{00, 11, 10, 01\}$;
 - P contains the following rules:
 1. $s0 \rightarrow s'\lambda$,
 2. $s1 \rightarrow s''\lambda$,
 3. $s'0 \rightarrow s(0, out)$,
 4. $s'1 \rightarrow s(1, out)$,
 5. $s''1 \rightarrow s(1, out)$,
 6. $s''0 \rightarrow s(1, out)$.

For any input multiset from set $\{00, 11, 10, 01\}$, $N_{(1,3)}\{n_{coo}, \alpha, \beta\}(\Pi)$ gives the OR value for all $\alpha \in \{min\}$, $\beta \in \{repl, one, spread\}$.

5.4 Simulation of XOR Gate

Here we simulate XOR gate using non-cooperative rules.

The tissue P system with only one cell and non-cooperative rules gives XOR value of two input values. Initially σ_1 contains any one of the multisets from the set $\{00, 11, 10, 01\}$. The tP system is $\Pi = (O, \sigma_1, syn, i_{out})$, where:

- $O = \{0, 1\}$;
- $syn = \emptyset$;
- $i_{out} \in \{1\}$ indicates the *output cell*;
- $\sigma_1 = (Q, s, w, P)$, where
 - $Q = \{s, s', s^\circ\}$ is a finite set of states;
 - $s \in Q$ is the initial state;
 - w is one of the multisets from the set $\{00, 11, 10, 01\}$;
 - P contains the following rules:
 1. $s0 \rightarrow s^\circ\lambda$,
 2. $s^\circ0 \rightarrow s(0, out)$,
 3. $s^\circ1 \rightarrow s(1, out)$,
 4. $s1 \rightarrow s'\lambda$,
 5. $s'1 \rightarrow s(0, out)$,
 6. $s'0 \rightarrow s(1, out)$.

For any input multiset from set $\{00, 11, 10, 01\}$, $N_{(1,3)}\{n_{coo}, \alpha, \beta\}(\Pi)$ gives the XOR value for all $\alpha \in \{min\}$, $\beta \in \{repl, one, spread\}$.

6 Simulating the n-Input Logical Gates Using Non-Cooperative Rules in Parallel Mode

In this section we simulate AND (\wedge), and OR (\vee) gates using non-cooperative rules. We take tissue P system with *par* mode as a processing mode for multisets of objects, and *one*, *spread*, *repl* modes as transmitting modes for the processed multisets of objects.

6.1 Simulation of n-Input AND Gate

Here we simulate the AND gate using non-cooperative rules. The n input to the AND gate is represented by a multiset over $\{0, 1\}$.

The tissue P system with one cell and non-cooperative rules gives AND value of n (not necessarily synchronized) input values. Here we consider the *par mode* for processing multiset of objects. Initially σ_1 contains any one of the multisets $w \in \{0, 1\}^*$ with $|w| \geq 2$ from the set. The tP system is $\Pi = (O, \sigma_1, syn, i_{out})$, where:

- $O = \{0, 1\}$
- $syn = \emptyset$
- $i_{out} \in \{1\}$ indicates the output cell;
- $\sigma_1 = (Q, s, w, P)$
 - $Q = \{s, s^\circ, s', s'^\circ\}$ is finite set of states;
 - $s \in Q$ is the initial state;
 - w is one of the multisets from the set $\{0, 1\}^*$ with $|w| \geq 2$;
 - P contains the following rules:
 1. $s0 \rightarrow s^\circ\lambda$,
 2. $s^\circ 1 \rightarrow s'^\circ\lambda$,
 3. $s^\circ\lambda \rightarrow s(0, out)$,
 4. $s'^\circ\lambda \rightarrow s(0, out)$,
 5. $s1 \rightarrow s'\lambda$,
 6. $s'0 \rightarrow s'^\circ\lambda$,
 7. $s'\lambda \rightarrow s(1, out)$.

With the help of few examples we explain how the system works as an AND gate.

- Assume that σ_1 initially contains the multiset 00000 and is in state s . By applying rule 1, all 0's are erased and the state becomes s° . Next, rule 3 is applied on λ , 0 is sent to outer region as a result and the state becomes s .

- Assume that σ_1 initially contains multiset 11111 and is in state s . Rule 5 is applied, all 1's are erased and the state becomes s' . Next, rule 7 is applied on λ , 1 is sent to outer region as a result and the state becomes s .
- Assume that σ_1 initially contains the multiset 101010; any one of the rules 1, 5 can be applied. If we apply rule 1 first, then all 0's are erased and the state becomes s° . Now rule 2 is applied, all 1's are erased and the state becomes s'° . Finally rule 4 is applied and 0 is sent out to outer region as a result. If we apply rule 5 first, then all 1's are erased, and the state becomes s' . Now rule 6 is applied, all 0's are erased, and the state becomes s'° . Finally, rule 4 is applied and 0 is sent out to outer region as a result and the state becomes s .

For a given input multiset with length n , $N_{(1,4)}ncoo, \alpha, \beta(\Pi)$ gives the AND value on that multiset, for all $\alpha \in \{par\}, \beta \in \{repl, one, spread\}$.

6.2 Simulation of n-Input OR Gate

Here we simulate OR gate using non-cooperative rules.

The tissue P system with two cells and non-cooperative rules gives OR value of n (not necessarily synchronized) input values. Here we consider the *par mode* for processing multisets of objects. Initially, σ_1 contains any one of the multisets $w \in \{0, 1\}^*$ with $|w| \geq 2$. The tP system is $\Pi = (O, \sigma_1, syn, i_{out})$, where:

- $O = \{0, 1\}$;
- $syn = \emptyset$;
- $i_{out} \in \{1\}$ indicates the output cell;
- $\sigma_1 = (Q, s, w, P)$, where
 - $Q = \{s, s^\circ, s', s'^\circ\}$ is finite set of states;
 - $s \in Q$ is the initial state;
 - w is one of the multisets from the set $\{0, 1\}^*$ with $|w| \geq 2$;
 - P contains the following rules:
 1. $s0 \rightarrow s^\circ\lambda$,
 2. $s^\circ 1 \rightarrow s'^\circ\lambda$,
 3. $s'^\circ\lambda \rightarrow s(1, out)$,
 4. $s^\circ\lambda \rightarrow s(0, out)$,
 5. $s1 \rightarrow s'\lambda$,
 6. $s'0 \rightarrow s'\lambda$,
 7. $s'\lambda \rightarrow s(1, out)$.

For a given input multiset with length n , $N_{(1,4)}\{ncoo, \alpha, \beta\}(\Pi)$ gives the OR value, for all $\alpha \in \{par\}, \beta \in \{repl, one, spread\}$.

7 Simulating the n-Input Logical Gates Using Non-Cooperative Rules in Minimal Mode

In this section we simulate the AND (\wedge), OR (\vee), and XOR (\oplus) gates using non-cooperative rules. We take tissue P system with the *min* mode as a processing mode for multisets of objects, and *one*, *spread*, *repl* modes as transmitting modes for processed multisets of objects.

7.1 Simulation of n-Input AND Gate

Here we simulate the AND gate using non-cooperative rules. The tissue P system with one cell and non-cooperative rules gives the AND value of n (not necessarily synchronized) input values. Here we consider *min mode* for processing multiset of objects. Initially σ_1 contains any one of the multisets from the set $\{0, 1\}^*$ with $|w| \geq 2$. The tP system is $\Pi = (O, \sigma_1, syn, i_{out})$, where:

- $O = \{0, 1\}$;
- $syn = \emptyset$;
- $i_{out} \in \{1\}$ indicates the output cell;
- $\sigma_1 = (Q, s, w, P)$, where
 - $Q = \{s, s^\circ, s'\}$ is a finite set of states;
 - $s \in Q$ is the initial state;
 - w is one of the multisets from the set $\{0, 1\}^*$ with $|w| \geq 2$;
 - P contains the following rules:
 1. $s0 \rightarrow s^\circ(0, out)$,
 2. $s^\circ 0 \rightarrow s^\circ \lambda$,
 3. $s^\circ 1 \rightarrow s^\circ \lambda$
 4. $s^\circ \lambda \rightarrow s$,
 5. $s1 \rightarrow s'\lambda$,
 6. $s'1 \rightarrow s'\lambda$,
 7. $s'0 \rightarrow s^\circ(0, out)$,
 8. $s'\lambda \rightarrow s(1, out)$.

With the help of few examples we explain how the system works as an AND gate.

- Assume that σ_1 initially contains the multiset 00000 and is in state s . Rule 1 is applied, 0 is sent out to the outer region as a result, and the state becomes s° . Now rule 2 is applied four times for erasing four 0's. Now rule 4 is applied, and the state becomes s .
- Assume that σ_1 initially contains the multiset 11111 and is in state s . Rule 5 is applied, one 1 is erased and the state becomes s' . Now rule 6 is applied 4 times for erasing four 1's, then rule 8 is applied, 1 is sent to outer region as a result and the state becomes s .

- Assume that σ_1 initially contains the multiset 010101 and is in state s . Any one of the rules 1, 5 can be applied. If we apply rule 1, then 0 is sent out as result and state becomes s° . Now rules 2, 3 are applied until all 0's and 1's are erased. Finally, rule 4 is applied on λ , and the state becomes s . If we apply rule 5, then one 1 is erased and state becomes s' and the resulting multiset is 01010. Rules 6 and rule 7 are applicable now. If we apply rule 6 two times, then 1's are erased completely and the state remains the same. Now we apply rule 7, 0 is sent out as a result and the state becomes s° . Then, rule 2 and rule 4 are applied, and finally the state becomes s .

For a given input multiset with length n , $N_{(1,3)}\{ncoo, \alpha, \beta\}(\Pi)$ gives the AND value, for all $\alpha \in \{min\}, \beta \in \{repl, one, spread\}$.

7.2 Simulation of n-Input OR Gate

Here we simulate the OR gate using non-cooperative rules. The tissue P system with one cell and non-cooperative rules gives OR value of n (not necessarily synchronized) input values. Here we consider the *min mode* for processing multiset of objects. Initially σ_1 contains any one of the multisets from the set $\{0, 1\}^*$ with $|w| \geq 2$. The tP system is $\Pi = (O, \sigma_1, syn, i_{out})$, where:

- $O = \{0, 1\}$;
- $syn = \emptyset$;
- $i_{out} \in \{1\}$ indicates the output cell;
- $\sigma_1 = (Q, s, w, P)$, where
 - $Q = \{s, s^\circ, s'\}$ is a finite set of states;
 - $s \in Q$ is the initial state;
 - w is one of the multisets from the set $\{0, 1\}^*$ with $|w| \geq 2$;
 - P contains the following rules:
 1. $s1 \rightarrow s'(1, out)$,
 2. $s'1 \rightarrow s'\lambda$,
 3. $s'0 \rightarrow s'\lambda$,
 4. $s'\lambda \rightarrow s$,
 5. $s^\circ 0 \rightarrow s^\circ \lambda$,
 6. $s^\circ 1 \rightarrow s^\circ \lambda$,
 7. $s^\circ 1 \rightarrow s'(1, out)$,
 8. $s^\circ \lambda \rightarrow s(0, out)$.

For a given input multiset with length n $N_{(1,3)}\{ncoo, \alpha, \beta\}(\Pi)$ gives the OR value for all $\alpha \in \{min\}, \beta \in \{repl, one, spread\}$.

7.3 Simulation of n-Input XOR Gate

Here we simulate the XOR gate using non-cooperative rules. The tissue P system with one cell and non-cooperative rules gives the XOR value of n (not necessarily synchronized) input values. Here we consider the *min mode* for processing multisets of objects. Initially, σ_1 contains any one of the multisets $w \in \{0,1\}^*$ with $|w| \geq 2$. For the multiset w with $|w| \geq 2$, the result depends on number of ones in that multiset, If the number of ones is even, then the output is 0, if the number of ones is odd, then the output is 1. If the input multiset contains only 0's, then the XOR value of that input is 0. The tP system is $\Pi = (O, \sigma_1, syn, i_{out})$, where:

- $O = \{0, 1\}$;
- $syn = \emptyset$;
- $i_{out} \in \{1\}$ indicates the output cell;
- $\sigma_1 = (Q, s, w, P)$, where
 - $Q = \{s, s^\circ, s'\}$ is a finite set of states;
 - $s \in Q$ is the initial state;
 - w is one of the multisets from the set $\{0,1\}^*$ with $|w| \geq 2$;
 - P contains the following rules:
 1. $s0 \rightarrow s^\circ\lambda$,
 2. $s^\circ 0 \rightarrow s^\circ\lambda$,
 3. $s^\circ 1 \rightarrow s'\lambda$,
 4. $s'1 \rightarrow s^\circ\lambda$,
 5. $s'0 \rightarrow s'\lambda$,
 6. $s'\lambda \rightarrow s(1, out)$,
 7. $s^\circ\lambda \rightarrow s(0, out)$,
 8. $s1 \rightarrow s'\lambda$,
 9. $s'0 \rightarrow s'\lambda$,
 10. $s'1 \rightarrow s^\circ\lambda$.

With the help of a few examples we explain how the system works as an XOR gate.

- Assume that initially σ_1 is in state s and contains the multiset 00000. Rule 1 is applied first, one 0 is erased and the state becomes s° and then rule 2 is applied 4 times for erasing the remaining 0's. Finally rule 7 is applied, 0 is sent to the outer region as a result and the state becomes s .
- Assume that initially σ_1 is in state s and contains the multiset 11. Rule 8 is applied, one 1 is erased and the state becomes s' . Rule 10 is applied, one 1 is erased and the state becomes s° . Finally rule 7 is applied, 0 is sent out to the outer region as a result and the state becomes s .

Likewise for all n -input multiset values we find the XOR values. If the number of 1's in input multiset is even, then the XOR value of that input is 0, otherwise the output is 1.

For a given input multiset with length n , $N_{(1,3)}\{n_{coo}, min, \beta\}(\Pi)$ gives the XOR value, for all $\beta \in \{repl, one, spread\}$.

8 Simulating Circuits

We now give an example of how to construct a tissue P system which simulates a Boolean circuit, designed for evaluating a Boolean function, using tissue P systems as constructed in the previous sections. Consider the function $f : \{0, 1\}^4 \rightarrow \{0, 1\}$ given by the formula $f(x_1, x_2, x_3, x_4) = (x_1 \vee x_2) \wedge \neg(x_3 \wedge x_4)$ and the circuit given in Figure 3.

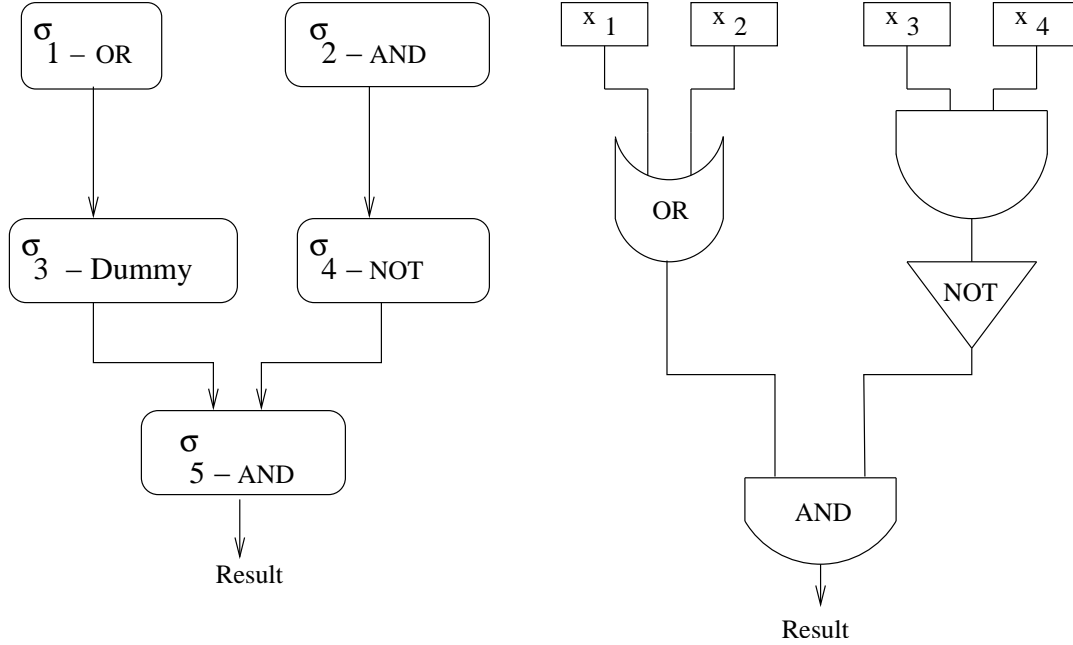


Figure 3: The tissue P system which simulates the computation of its associated circuit

The following tP system simulates the circuit from this figure:

$$\Pi = (O, \sigma_{1-OR}, \sigma_{2-AND}, \sigma_{3-DUMMY}, \sigma_{4-NOT}, \sigma_{5-AND}, syn, i_{out}),$$

where:

- $O = \{0, 1\}$;
- $syn = \{(1, 3), (2, 4), (3, 5), (4, 5)\}$;
- $i_{out} \in \{5\}$ indicates the *output cell*;
- $\sigma_{1-OR} = (Q_1, s, w_1, P_1)$, where
 - $Q_1 = \{s, s', s''\}$ is a finite set of states;

- $s \in Q$ is the initial state;
- w_1 is one of the multisets from the set $\{00, 11, 10, 01\}$;
- P_1 contains the following rules:
 1. $s0 \rightarrow s'\lambda$,
 2. $s1 \rightarrow s''\lambda$,
 3. $s'0 \rightarrow s(0, out)$,
 4. $s'1 \rightarrow s(1, out)$,
 5. $s''1 \rightarrow s(1, out)$,
 6. $s''0 \rightarrow s(1, out)$
- $\sigma_{2-AND} = (Q_2, s, w_2, P_2)$, where
 - $Q_2 = \{s, s', s''\}$ is a finite set of states;
 - $s \in Q$ is the initial state;
 - w_2 is one of the multisets from the set $\{00, 11, 10, 01\}$;
 - P_2 contains the following rules:
 1. $s0 \rightarrow s'\lambda$,
 2. $s1 \rightarrow s''\lambda$,
 3. $s'0 \rightarrow s(0, out)$,
 4. $s'1 \rightarrow s(0, out)$,
 5. $s''1 \rightarrow s(1, out)$,
 6. $s''0 \rightarrow s(0, out)$
- $\sigma_{3-DUMMY} = (Q_3, s, w_3, P_3)$, where
 - $Q_3 = \{s\}$ is a finite set of states;
 - $s \in Q$ is the initial state;
 - $w_3 = \phi$;
 - $P_3 = \{s0 \rightarrow s(0, out), s1 \rightarrow s(1, out)\}$;
- $\sigma_{4-NOT} = (Q_4, s, w_4, P_4)$, where
 - $Q_4 = \{s\}$ is finite set of states;
 - $s \in Q$ is the initial state;
 - $w_4 = \phi$;
 - $P_4 = \{s0 \rightarrow s(1, out), s1 \rightarrow s(0, out)\}$;
- $\sigma_{5-AND} = (Q_5, s_5, w_5, P_5)$, where
 - $Q_5 = \{s, s', s''\}$ is a finite set of states;
 - $s \in Q$ is the initial state;
 - $w_5 = \phi$;
 - P_5 contains the following rules:

1. $s0 \rightarrow s'\lambda$,
2. $s1 \rightarrow s''\lambda$,
3. $s'0 \rightarrow s(0, out)$,
4. $s'1 \rightarrow s(0, out)$,
5. $s''1 \rightarrow s(1, out)$..
6. $s''0 \rightarrow s(0, out)$

Here we consider 2-input $\vee(OR)$, $\wedge(AND)$ gates and 1-input NOT (\neg) gate, which are constructed by tissue P systems. Initially we place multisets of size 2 in σ_{1-OR} and σ_{2-AND} . The other cells do not have any objects in them at this point. The two cells with inputs produce results at the same time as we assume that computation time taken by each cell is the same when two inputs are present in the cell. The $\sigma_{3-Dummy}$ cell sends the input as it is as a result to σ_{5-AND} , and σ_{4-NOT} gives its result to σ_{5-AND} . Finally σ_{5-AND} cell computes AND value of two previous input values. Here we need 5 cells to simulate the circuit. For synchronization we have used the $\sigma_{3-Dummy}$ cell. If this cell is not considered, result from σ_{1-OR} reaches σ_{5-AND} first and the computation starts in σ_{5-AND} . After this symbol is processed, the cell waits for a symbol which is to be sent by σ_{4-NOT} . As soon as this symbol reaches σ_{5-AND} , the computation is carried out by this cell and the result is sent out to the environment. In all the cases the cells compute any one of the gate operations according to the given circuit and send the results to some other cells according to synapses and mode. In this manner Boolean circuits are simulated by tissue P systems.

9 Conclusion

In this paper we propose a model for simulating Boolean circuits with *tissue P systems*. First we simulate gates of circuits and next we combine these gates to get circuits. Here we consider the *min* and *par* modes of processing multisets of objects and all three modes for transmitting the processed multisets from one cell to another cell.

Our model is efficient in a sense that using it we can simulate the elementary logical functions NOT (\neg), AND (\wedge), OR (\vee), XOR (\oplus), and Boolean circuits. We simulate 1-input NOT gate, 2-input logical gates AND, OR, XOR using cooperative and non cooperative rules, and n -input logical gates AND, OR, XOR using non cooperative rules. Here we have taken functions $f : \{0, 1\}^n \rightarrow \{0, 1\}$. If we want to simulate functions $f : \{0, 1\}^n \rightarrow \{0, 1\}^m$, we may need proper synapses and the *repl* mode.

References

- [1] M.A. Arbib, *Brains, Machines, and Mathematics*, Springer, Berlin, 1987.
- [2] N. Balabanian, B. Carlson, *Digital Logic Design Principles*, John Wiley and Sons, Inc., 2001.

- [3] M. Sakthi Balan, K. Krithivasan, Realizing Switching Functions using Peptide-Antibody Interactions, *Aspects of Molecular Computing*, LNCS 2950 (N. Jonoska, Gh. Păun, G. Rozenberg, eds.), Springer, Berlin, 2004.
- [4] R. Ceterchi, D. Sburlan, Simulating Boolean Circuits with P Systems, *Preproceedings of the Workshop on Membrane Computing*, (A. Alhazov, C. Martín-Vide, Gh. Păun, eds.), 2003, 145–160.
- [5] M. Madhu, V. Jaya Prakash, K. Krithivasan, Rewriting Tissue P Systems, *Journal of Universal Computer Science*, 2004, to appear.
- [6] C. Martín-Vide, Gh. Păun, J. Pazos, A. Rodríguez-Patón, Tissue P Systems, *Theoretical Computer Science*, 296, 2 (2003), 295–326.
- [7] M. Ogihara, A. Ray, Simulating Boolean Circuits on a DNA Computer, Technical Report 631, Department of Computer Science, University of Rochester, 1996.
- [8] Gh. Păun, Computing with Membranes, *Journal of Computer and System Sciences*, 61, 1 (2000), 108–143.
- [9] G. Rozenberg, A. Salomaa, *Handbook of Formal Languages*, Springer, Berlin, 1997.