(Very) Initial Ideas on Non-cooperative Polymorphic P Systems and Parallel Communicating ET0L Systems

Anna Kuczik and György Vaszil

Faculty of Informatics, University of Debrecen
Kassai út 26, 4028 Debrecen, Hungary
kuczik.anna@inf.unideb.hu
vaszil.gyorgy@inf.unideb.hu

Summary. In this research, we begin to investigate the relationship between polymorphic P systems and parallel communicating ET0L systems. Our goal is to present an example, based on the definitions, from which it seems that there is some connection between the two systems.

Key words: Polymorphic P systems, P systems with non-cooperative rules, parallel communicating grammar system, parallel communicating ET0L systems

1 Introduction

Membrane systems (P systems) are computational models whose computation is based on the processes taking place in living cells. They consist of several nested membranes, these are called regions. The contents of the regions are multisets. In each step, we apply rule(s) in each region (if applicable), so we apply multiple rewriting rules in parallel until we reach a halting configuration.

The difference between polymorphic P systems and P systems is the relationship between multisets and rules. In polymorphic P systems, the contents of the regions form the rules. As the contents of the regions change, the corresponding rules also change, we call these dynamic rules. Each rule has two regions that make up the left and right sides of the rule. For more information, see the survey [1].

The results of the article [2] demonstrate the power of the model. in the case of using cooperative rules, any recursively enumerable set of numbers is generated. As a result, we deal with the non-cooperative case, which generates languages, interesting, mainly from the point of view that exponential, even super-exponential growth of the number of objects within the system can be achieved.

In this article, we begin to investigate the relationship between non-cooperative polymorphic P systems and parallel communicating ET0L systems. In the follow-

ing, after the necessary preliminaries and definitions in section 2, we present an example in section 3. Through this example, we show that it is possible that some kind of relationship exists between the two models, based on the definitions. We create a parallel communicating ET0L system that simulates the computation of a simple non-cooperative polymorphic P system.

2 Preliminaries and Definitions

In this section, we define the basic definitions and notions we will use. For more information about formal language theory, see [3], and [4, 5] for details about membrane computing.

First, we define the formal alphabet. An *alphabet* V is a finite non-empty set of symbols called letters. A string (or word) over V is a finite sequence of letters, the set of all strings over V is denoted by V^*, and $V^+ = V^* \setminus \{\lambda\}$ where λ denotes the empty string. If we fix an order $V = \{a_1, a_2, \ldots, a_n\}$ of the letters, then the vector $(|w|_{a_1}, |w|_{a_2}, \ldots, |w|_{a_n})$ is called the Parikh vector of the word $w \in V^*$.

If $\mathbb{N}$ denotes the set of nonnegative integers, then a *multiset* over a set U is a mapping $M : U \to \mathbb{N}$ where $M(a)$, for all $a \in U$, is the multiplicity of element a in the multiset M. If U is finite, $U = \{a_1, a_2, \ldots a_n\}$, then M can also be represented by a string $w = a_1^{M(a_1)} a_2^{M(a_2)} \ldots a_n^{M(a_n)}$ (and all permutations of this string) where a^j denotes the string obtained by concatenating $j \in \mathbb{N}$ occurrences of the letter $a \in V$ (with $a^0 = \lambda$).

A. Lindenmayer introduced Lindenmayer systems (L systems for short) in 1968. He introduced these systems with the aim of being able to describe the development of organisms known from biology using formal languages. L systems are parallel rewriting systems, see [6, 7] for more information on this area.

In the following, we define a version of the L systems, the ET0L systems, which are extended, tabled, and interactionless versions of the original L systems.

An *ET0L system* is a quadruple $G = (V, T, U, \omega)$ where V is an alphabet, $T \subseteq V$ is a terminal alphabet, $\omega \in V^+$ is the initial word of G, and $U = (P_1, \ldots, P_m)$ where $P_i,\ 1 \leq i \leq m$, are finite sets of context-free productions over V (called *tables*), such that for each $a \in V$, there is at least one rule $a \to \alpha,\ \alpha \in V^*$ in each table.

In each computational step, G rewrites all the symbols of the current sentential form with the rules of one of the tables in U. The language generated by G consists of all terminal strings which can be generated in a series of computational steps (a derivation) starting from the initial word.

Let $L(G)$ be the language generated by G, then $L(G) = \{u \in T^* \mid w \Rightarrow^* u\}$ where $\Rightarrow$ denotes a computational step , and $\Rightarrow^*$ is the reflexive and transitive closure of $\Rightarrow$.

We are not interested in the character string generated by the ET0L system as a sequence of letters, but only in the multiples of the different letters, i.e. the Parikh vectors of the words. This is necessary because we will connect the ET0L

languages to the multiset languages of the P systems. We denote by $Ps(G)$ the set of Parikh vectors corresponding the strings of $L(G)$ (Parikh set of $L(G)$), and by $PsET0L$ the class of Parikh sets corresponding to the class of languages generated by ET0L systems.

Polymorphic membrane systems were introduced in [2]. The rules in polymorphic P systems are defined by the contents of specific membrane regions corresponding to the left- and right-hand sides of the rule. As a result, the rules belonging to the regions change(s) during the computation. These rules are called dynamic rules.

A *polymorphic P system* is a tuple

$$\Pi = (O, T, \mu, w_s, \langle w_{1L}, w_{1R} \rangle, \ldots, \langle w_{nL}, w_{nR} \rangle, h_o),$$

where O is the alphabet of objects, $T \subseteq O$ is the set of terminal objects, μ is the membrane structure consisting of $2n+1$ membranes labelled by a symbol from the set $H = \{s, 1L, 1R, \ldots, nL, nR\}$, the elements of the multiset w_s are the initial contents of the skin membrane, the pairs of multisets $\langle w_{iL}, w_{iR} \rangle$ correspond to the initial contents of membranes iL and iR, $1 \leq i \leq n$, and $h_o \in H$ is the label of the output membrane.

The rules of the polymorphic membrane system are not given statically in the initial configuration. In each step, they are dynamically derived based on the contents of the left and right (iL and iR, $1 \leq i \leq n$) membrane pairs. Thus, if the membranes iL and iR belonging to the i-th membrane pair contain multisets u and v respectively, then in the next step we transform their parent membrane as if the multiset rewriting rule $u \to v$ were present.

If there is at least one rule in a system Π where the number of objects in u (the multiset on the left-hand side) can grow to be greater than one, then we say that Π is a *cooperative* system, otherwise, it is a *non-cooperative* system. The P system is a series of computational steps in which the rules belonging to the given region are applied in a maximal parallel way. Each object can be rewritten by at most one rule in one step. The P system reaches a halting configuration when no rule can be applied in any of the regions, so no more computational steps are possible.

The set of vectors $N(\Pi)$ generated by the polymorphic P system Π with the terminal alphabet T is the set of Parikh vectors among the strings $w \in T^*$ corresponding to the output h_o of multisets of terminal objects appearing in the region in a halting configuration Π, which is reached by computation starting in the initial configuration of the system.

We need the finitely representable (FIN-representable) property to define the possible objects belonging to the regions of the membrane system. FIN-representable property were introduced in [8]. Before describing the finitely representable property, we need a definition of σ^*.

Let $\Pi = (O, T, \mu, w_s, \langle w_{1L}, w_{1R} \rangle, \ldots, \langle w_{nL}, w_{nR} \rangle, h_o)$ be a polymorphic P system, and let $w_{j,h}$ denote the multiset after the jth step of the computation contained by the region labelled by h of Π for some $j \geq 0$, where $h \in \{s, 1L, 1R, \ldots, nL, nR\}$. We say that $w'_{j,h}$ is in the *successor set* of $w_{j,h}$, denoted

as $w'_{j,h} \in \sigma_{j,h}(w_{j,h})$, if $w'_{j,h}$ can be obtained from $w_{j,h}$ by the maximally parallel applications of the multiset rewriting rules associated to the region h.

If for the same $w_{j,h}$ as above, we fix $\sigma^0_{j,h}(w_{j,h}) = \{w_{j,h}\}$ for $k \geq 0$ and for any $j \geq 0$, we have $\sigma^{k+1}_{j,h}(w_{j,h}) = \sigma_{j+k,h}(\sigma^k_{j,h}(w_{j,h}))$, then we can define

$$\sigma^*_{j,h} = \bigcup_{k \geq 0} \sigma^k_{j,h}(w_{j,h}).$$

Given a polymorphic P system (Π), a region h of Π is *finitely representable* (or $FIN-representable$) if, starting from w_h, the multiset of initial objects of h, the set of successor multisets of w_h is finite, $\sigma^*_{0,h}(w_h)$ is finite.

We will need the definition of a finite transition system. Later on, we can represent the rules for the regions of the membrane system with state transitions. An finite transition system M can be denoted as a triple $M = (Q, q, \delta)$ where Q is a finite set of states, $q \in Q$ is the initial state, and $\delta : Q \to 2^Q$ is the state transition mapping. A state $q' \in \delta(q)$ is called the successor state of q, and $q \in Q$ is called a *halting state* if $\delta(q) = \emptyset$.

Parallel communicating grammar systems with Lindenmayer systems as components were introduced in [7]. In the following we recall the definition of parallel communicating ET0L systems (PC ET0L systems for short) based on [9].

A *parallel communicating grammar system* with n components is a (n+3) tuple

$$\Gamma = (N, K, T, G_1, \ldots, G_n), \text{ where}$$

N is a nonterminal alphabet, T is a terminal alphabet and K is an alphabet of query symbols ($K = \{Q_1, Q_2, \ldots, Q_n\}$). N, K and T are pairwise disjoint sets. In case the components are Lindenmayer systems, then $G_i = (N \cup K, T, P_i, \omega_i)$, where $1 \leq i \leq n$ with nonterminal and terminal alphabets as above, a table of rewriting rules in case of ET0L systems P_i, and an axiom $\omega_i \in (N \cup T)^*$. In most cases we call G_1 the master grammar of Γ.

The *language* generated by a parallel communicating system of extended Lindenmayer systems, $\Gamma = (N, K, T, G_1, \ldots, G_n)$, where $G_i = (N \cup K, T, P_i, \omega_i)$, $1 \leq i \leq n$, is

$$L(\Gamma) = \{\alpha_1 \in T^* | (\omega_1, \ldots, \omega_n) \Rightarrow^* (\alpha_1, \ldots, \alpha_n)\}$$

where G_1 is the master grammar of Γ.

3 Polymorphic P systems vs. PC ET0L systems

In this chapter, we start to examine the relationship between general non-cooperative polymorphic P systems and parallel communicating ET0L systems.

Using the definitions introduced in the previous chapter, we would like to show that probably the PC ET0L systems can generate the same class of languages as non-cooperative polymorphic P systems. Let's examine an example in which we construct a PC ET0L system for a non-cooperative polymorphic P system.

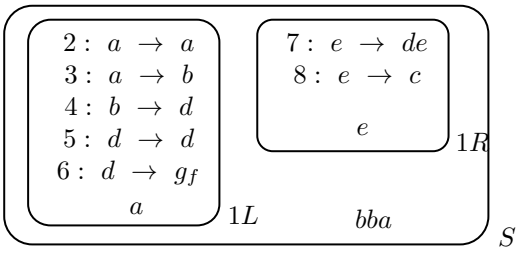

Fig. 1: The polymorphic P system Π of Example 1

Example 1. Consider a non-cooperative polymorphic P system

$$\Pi = (O, T, \mu, w_s, \langle w_{1L}, w_{1R}\rangle, \ldots, \langle w_{8L}, w_{8R}\rangle, s)$$

where $O = T = \{a, b, c, d, e, g_f\}$ and the membrane structure is

$$\mu = [\ [\ldots]_{1L}\ [\ldots]_{1R}\ ]_s,$$

where the child membranes of 1L are $[\ ]_{2L}\ [\ ]_{2R} \ldots [\ ]_{6L}\ [\ ]_{6R}$, the children of 1R are $[\ ]_{7L}\ [\ ]_{7R}$, $[\ ]_{8L}\ [\ ]_{8R}$.

Let

$$w_s = bba,\ w_{1L} = a,\ w_{1R} = e,$$

and using the simplified notation for static (non-dynamic) rules, let the rules corresponding to 1L be

$$r_2 : a \to a,\ r_3 : a \to b,\ r_4 : b \to d,\ r_5 : d \to d, r_6 : d \to g_f,$$

and the rule corresponding to 1R be $r_7 : e \to de, r_8 : e \to c$, as illustrated in Figure 1.

According to the non-cooperative property, in each step 1-1 letter changes in $1L$. In the following we show that $1L$ is FIN-representable. Concerning $1L$, observe that $\sigma^*_{0,1L}(a) = \{a, b, d, g_f\}$ with $\sigma_{0,1L}(a) = \sigma_{j,1L}(a) = \{a, b\}$, $\sigma_{0,1L}(b) = \sigma_{j,1L}(b) = \{d\}$, $\sigma_{0,1L}(d) = \sigma_{j,1L}(d) = \{d, g_f\}$, and $\sigma_{0,1L}(g_f) = \sigma_{j,1L}(g_f) = \emptyset$ for all $j \geq 0$.

The $1R$ region is not FIN-representable, but from the point of view of the task, for us, the examination of the stopping of the regions on the right is not essential. It is sufficient that the FIN-representable property is true for the left sides, which will be true in all cases due to the non-cooperative property.

The skin region is not FIN-representable, as the dynamical rule r_1 given by the membranes labelled with $1L$ and $1R$ has more than one symbol on its right-hand side in each computational step, and this means that the number of symbols in the skin region is increasing with each rule application.

Following the steps introduced in [8], starting from the deepest (static) rules of the membrane system, we create the transition systems for the regions. To simplify

the example, we do not create transition systems for the static rules, but we know that they are the basis of the transition systems created for the higher levels. Due to the non-cooperativity and the FIN-representable property, a transition system can be constructed for each left-side membrane. Based on this, let's construct a transition system for the $1L$ region.

Start with the construction of $M_{2\ldots6} = (R_{2\ldots6}, \bar{r}_{2\ldots6}, \delta_{2\ldots6})$ as

- $R_{2\ldots6} = \{\bar{r}_{2\ldots6}\}$ where
- $\bar{r}_{2\ldots6} = (\bar{r}_2, \ldots, \bar{r}_6)$ with $\bar{r}_2 = ((a,\emptyset),(a,\emptyset))$, $\bar{r}_3 = ((a,\emptyset),(b,\emptyset))$, $\bar{r}_4 = ((b,\emptyset),(d,\emptyset))$, $\bar{r}_5 = ((d,\emptyset),(d,\emptyset))$, $\bar{r}_6 = ((d,\emptyset),(g_f,\emptyset))$, and
- $\delta_{2\ldots6}(\bar{r}_{2\ldots6}) = \emptyset$.

Note that the rule set corresponding to $\bar{r}_{2\ldots6}$ is $\{r_2, r_3, r_4, r_5, r_6\} = \{a \to a, a \to b, b \to d, d \to d, d \to g_f\}$.

Now we can construct M_{1L} transition system as follows: $M_{1L} = (Q_{1L}, q_0, \delta_{1L})$, where the set of possible states is $Q_{1L} = \{a, b, d, g_f\} \times \{(\bar{r}_2, \bar{r}_3, \bar{r}_4, \bar{r}_5, \bar{r}_6)\}$, that is,

$$Q_{1L} = \{q_0 : (a,\ (\bar{r}_2, \bar{r}_3, \bar{r}_4, \bar{r}_5, \bar{r}_6)),\ q_1 : (b,\ (\bar{r}_2, \bar{r}_3, \bar{r}_4, \bar{r}_5, \bar{r}_6)),$$
$$q_2 : (d,\ (\bar{r}_2, \bar{r}_3, \bar{r}_4, \bar{r}_5, \bar{r}_6)),\quad q_3 : (g_f,\ (\bar{r}_2, \bar{r}_3, \bar{r}_4, \bar{r}_5, \bar{r}_6))\},$$

the initial state is q_0, and the transition mapping is defined as

$$\delta_{1L}(q_0) = \{(q_0), (q_1)\},$$
$$\delta_{1L}(q_1) = \{(q_2)\},$$
$$\delta_{1L}(q_2) = \{(q_2), (q_3)\},$$
$$\delta_{1L}(q_3) = \emptyset.$$

It follows from this transition map that the state that starts with g_f is a final state. This shows that the $1L$ region is FIN-representable.

Construct a PC ET0L system that simulates the operation of the polymorphic membrane system. During construction, the current states of the FIN-representable regions must be recorded in PC ET0L sentential form.

Let us consider a PC ET0L system $\Gamma = (N, K, T, G_{s1L}, G_{1R}, G_m, G_c)$, simulating this membrane system.

In the following $|P_i|$ denote the number of tables in P_i, and $P_{i,j}$ denote the j-th table of P_i, $1 \le i \le n$, $1 \le j \le |P_i|$. $|P_{s1L}| = 4, |P_{1R}| = 1, |P_m| = 1, |P_c| = 1$. Let

$$N = \{q_0, q_1, q_2, q_3, F, S_{1R}, S_m, S_c\} \text{ non-terminals},$$
$$K = \{Q_{1R}, Q_{s1L}, Q_c\} \text{ query symbols and}$$
$$T = \{a, b, c, d, e, \} \text{ terminals}.$$

The initial string of the G_{s1L} component be ω_{s1L} and the associated tables are as follows:

$$\begin{aligned}
\omega_{s1L} &= bbaq_0,\\
P_{s1L,1} &= \{q_0 \to q_0, q_0 \to q_1, q_1 \to F, q_2 \to F, q_3 \to F, a \to Q_{1R}\},\\
P_{s1L,2} &= \{q_1 \to q_2, q_0 \to F, q_2 \to F, q_3 \to F, b \to Q_{1R}\},\\
P_{s1L,3} &= \{q_2 \to q_2, q_2 \to q_3, q_0 \to F, q_1 \to F, q_3 \to F, d \to Q_{1R}\}\\
P_{s1L,4} &= \{q_3 \to q_3, q_0 \to F, q_1 \to F, q_2 \to F\}
\end{aligned}$$

The initial string of the G_{1R} component be ω_{1R} and the associated tables are as follows:

$$\begin{aligned}
\omega_{1R} &= S_{1R},\\
P_{1R,1} &= \{S_{1R} \to e, e \to de, e \to c\}.
\end{aligned}$$

The initial string of the G_m component be ω_m and the associated tables are as follows:

$$\begin{aligned}
\omega_m &= S_m,\\
P_{m,1} &= \{S_m \to S_m, S_m \to Q_{s1L}, q_3 \to Q_c, x \to F | x \neq q_3\},
\end{aligned}$$

The initial string of the G_c component be ω_c and the associated tables are as follows:

$$\begin{aligned}
\omega_c &= S_c,\\
P_{c,1} &= \{S_c \to S_c, S_c \to Q_{1R}, x \to \lambda, e \to F | x \in (c, d)\},
\end{aligned}$$

In the case of P systems, we denote the current states as follows: (u, v, w), where u denotes the contents of s, v denotes the contents of $1L$ and w denotes the contents of $1R$. To simplify the example, static regions are not marked separately, since their content is always constant. With this triple, we only consider those regions whose contents change.

The initial configuration is based on the construction of the P system, we can choose between two rules (rule 2 and rule 3) for the object a in $1L$ during the first step. Consequently, there are two different cases. Let's examine both cases. First, let's look at an example where Π applies $r_2 : a \to a$ during steps 1 and 2, and the rule 3 .

$$\begin{aligned}
&(bba, a, e) \Rightarrow^{(1,2,7)} (bbe, a, de) \Rightarrow^{(-,2,7)}\\
&(bbe, a, dde) \Rightarrow^{(-,3,8)} (bbe, b, ddc) \Rightarrow^{(1,4,-)}\\
&(ddcddce, d, ddc) \Rightarrow \ldots
\end{aligned}$$

The triple index above the arrows are the numbers of the currently applied rules in each region. For example: $(bba, a, e) \Rightarrow^{(1,2,7)} (bbe, a, de)$ means, that we used rule 1 in s, rule 2 in $1L$ and rule 7 in $1R$.

So we start with (bba, a, e), apply $r_1 : a \to e$ rule in s, rule 2 in $1L$ and rule 7 in $1R$. After applying these rules, the regions change as (bbe, a, de). At this point we are not able to apply rule 1 in s, but we can use rule 2 or rule 3 in $1L$, rule 7 or 8 in $1R$. Use rule 2 and rule 7: (bbe, a, dde). We (still) cannot apply rule 1 in s, apply rule 3 in $1L$ and rule 8 in $1R$: (bbe, b, ddc). Apply rule 1 in s, rule 4 in $1L$: $(ddcddce, d, ddc)$.

In the case of PC ET0L systems, we denote the current strings of components as follows: $(u_{s1L}, u_{1R}, u_m, u_c)$, where u_{s1L} denotes the sentential form of G_{s1L}, where s index denotes the content of S in Π and $1L$ index denotes the content of $1L$ region in Π, u_{1R} denotes the sentential form of G_{1R}, u_m denotes the sentential form of G_m, u_c denotes the sentential form of G_c.

We simulate the steps performed by the polymorphic P system:

$$
\begin{aligned}
&(\ bbaq_0, S_{1R}, S_m, S_c) \Rightarrow^{(1,1,1,1)} (bbQ_{1R}q_0, e, S_m, S_c) \Rightarrow_{com} \\
&(\ bbeq_0, e, S_m, S_c) \Rightarrow^{(1,1,1,1)} (bbeq_0, de, S_m, S_c) \Rightarrow^{(1,1,1,1)} \\
&(\ bbeq_1, dde, S_m, S_c) \Rightarrow^{(2,1,1,1)} (Q_{1R}Q_{1R}eq_2, ddc, S_m, S_c) \Rightarrow_{com} \\
&(\ ddcddceq_2, ddc, S_m, S_c) \Rightarrow \dots
\end{aligned}
$$

The quadruple index above the arrows (except for arrows for communication steps) are the indexes of the currently applied tables in each component. For example: $(bbaq_0, S_{1R}, S_m, S_c) \Rightarrow^{(1,1,1,1)} (bbQ_{1R}q_0, e, S_m, S_c)$ means, that we used the first tables in all components.

Start with $(bbaq_0, S_{1R}, S_m, S_c)$, apply $a \to Q_{1R}$ and $q_0 \to q_0$ rules from $P_{s1L,1}$ in G_{S1L}, $S_{1R} \to e$ rule from $P_{1R,1}$ in G_{1R}, and $S_m \to S_m$, $S_c \to S_c$ from $P_{m,1}$ and $P_{c,1}$ in G_m an G_c (apply these rules until the end of the calculation). With the appearance of the Q_{1R} query symbol, a communication step follows ($\Rightarrow_{com}$ denotes the communication steps). In the communication step, with the help of Q_{1R}, we can insert the sentential form from G_{1R} into G_{s1L}.

Following the further steps, it can be seen that the same values appear in component G_{s1L} as in the s region in Π_2.

With the help of query symbols, PC ET0L can simulate the rewriting of the contents of s in the P system in one step + one communication step.

Let's look at an example where the P system applies rule 2 during step 1 and rule 3 during step 2, i.e. $r_2 : a \to a$ first, and then $r_3 : a \to b$.

$$
\begin{aligned}
&(\ bba, a, e) \Rightarrow^{(1,3,7)} (bbe, b, de) \Rightarrow^{(1,4,7)} (dedee, d, dde) \Rightarrow^{(1,5,7)} \\
&(\ ddeeddeee, d, d^3e) \Rightarrow^{(1,5,8)} (d^3ed^3eeed^3ed^3eeee, d, d^4e) \Rightarrow \dots
\end{aligned}
$$

Similar to the previous example, by taking one step + one-communication steps, the PC ET0L simulates the steps of the polymorphic P system.

$(bbaq_0, S_{1R}, S_m, S_c) \Rightarrow^{(1,1,1,1)} (bbQ_{1R}q_1, e, S_m, S_c) \Rightarrow_{com}$
$(bbeq_1, e, S_m, S_c) \Rightarrow^{(2,1,1,1)} (Q_{1R}Q_{1R}eq_2, de, S_m, S_c) \Rightarrow_{com}$
$(dedeeq_2, de, S_m, S_c) \Rightarrow^{(3,1,1,1)} (Q_{1R}eQ_{1R}eeq_2, dde, S_m, S_c) \Rightarrow_{com}$
$(ddeeddeeeq_2, dde, S_m, S_c) \Rightarrow^{(3,1,1,1)}$
$(Q_{1R}Q_{1R}eeQ_{1R}Q_{1R}eeeq_2, ddde, S_m, S_c) \Rightarrow_{com}$
$(d^3ed^3eeed^3ed^3eeeeq_2, ddde, S_m, S_c) \Rightarrow \ldots$

As we can see in the previous two examples, we can assuem that PC ET0L can simulate the first few steps of the Polymorphic P system in the subsequent steps using the appropriate tables.

In general the configuration of the polymorphic P system can be denoted by a triple, where α is the content of S, x is the content of $1L$, a single object due to the non-cooperative property, and β' is the content of $1R$:

(α, x, β') where $\alpha = \alpha_1 x \alpha_2 x \ldots x \alpha_k$
then we can denote the triple as follows: $(\alpha_1 x \alpha_2 x \ldots x \alpha_k, x, \beta')$.

In general the configuration of the PC ET0L system can be denoted with a 4-tuple, with its components. The sentential form of the first component: u_{s1L}, in which s index means that this component contains the content of s and $1L$ index means that this component also contains the content of $1L$. The sentential form of the second component: β, where β is the ancestor of β' appearing in the polymorphic P system. The sentential form of the third component: S_m, the content of G_m, and the sentential form of the fourth component: S_c, the content of G_c

$(u_{s1L}, \beta, S_m, S_c)$ where $u_{s1L} = \alpha_1 x \alpha_2 x \ldots x \alpha_k q_i$,
then the four component can be denoted as follows:
$(\alpha_1 x \alpha_2 x \ldots x \alpha_k q_i, \beta, S_m, S_c)$ where

the x's are denotes the first components of q_i (i.e., a, b, d, or g_f).

In the case of this general construction, examine how the content of the P system changes after one step. Apply the first rule, which is $x \rightarrow \beta'$, to the content of s, i.e., the first element of the triple. Apply a rule to x from the applicable rules of $1L$, if it exists, let x's successor be x'. Similarly for β', apply a rule from the applicable rules of $1R$, if it exists, let β''s successor be β'':

$$(\alpha_1 x \alpha_2 x \ldots x \alpha_k, x, \beta') \Rightarrow (\alpha_1 \beta' \alpha_2 \beta' \ldots \beta' \alpha_k, x', \beta'').$$

In order for the PC ET0L system to be able to simulate this step, it must take one step and one communication step. In the first component, applying the rule corresponding to x, it rewrites the x's to the query symbols Q_{1R}. In parallel, apply one of the following applicable rules to q_i , the same one that the polymorphic P

system applied to $1L$. In the second component, apply one of the following applicable rules to β, the one that transforms β'. In the third and fourth component, apply $S_m \to S_m$ and $S_c \to S_c$ rules. This step is followed by the communication step. In accordance with the PC ET0L, the content of the corresponding component, in this case the content of the second component, G_{1R}, is copied to the place of the query symbols in the first component. The content of the second component is β', which is the same as the right side of rule 1 applied in the polymorphic P System. It can be seen that the content of the first component of Γ $(\alpha_1\beta'\alpha_2\beta' \dots \beta'\alpha_k q_i')$ is the same as the content of s and $1L$ after one step in the P system:

$$
\begin{aligned}
&(\ \alpha_1 x \alpha_2 x \dots x \alpha_k q_i, \beta, S_m, S_c) \Rightarrow \\
&(\ \alpha_1 Q_{1R} \alpha_2 Q_{1R} \dots Q_{1R} \alpha_k q_i', \beta', S_m, S_c) \Rightarrow_{com} \\
&(\ \alpha_1 \beta' \alpha_2 \beta' \dots \beta' \alpha_k q_i', \beta', S_m, S_c).
\end{aligned}
$$

The polymorphic P system reaches a halting state when there are no rules in any region that can be applied to its content. In this example, the system reaches the halting configuration when there are no applicable rules for the contents of $1L, 1R$ and s.

In $1L$ region, the computation stops, after applying rule $(d \to g_f)$. In $1R$ region, the computation stops, after applying rule $(e \to c)$.

After applying $(d \to g_f)$ in $1L$, rule 1 can never be applied in s again.

In general, we can say that the polymorphic P system is in a halting state if its configuration:

$$(\alpha, g_f, d \dots dc).$$

Then α is the word (multiset) generated by the polymorphic P system.

To generate the α with the PC ET0L system we need G_m and G_c components. The G_m and G_c components apply the corresponding $S_m \to S_m$, $S_c \to S_c$ rules as long as the calculation is in progress. In order to generate α, the rule $S_m \to Q_{S1L}$ must be applied in G_m if the symbol q_3 appeared in G_{S1L}. In parallel, we apply the rule $S_c \to Q_{1R}$ in G_c:

$$
\begin{aligned}
&(\ \alpha q_3, \beta, S_m, S_c) \text{ where } \beta = dd \dots dc \\
&(\ \alpha q_3, \beta, S_m, S_c) \Rightarrow (\alpha q_3, \beta, Q_{S1L}, Q_{1R}) \Rightarrow_{com} \\
&(\ \alpha q_3, \beta, \alpha q_3, \beta) \Rightarrow \dots .
\end{aligned}
$$

Then, after the communication step, the rules of the $P_{m,1}$ table belonging to the G_m component and the $P_{c,1}$ table belonging to the G_c can be applied.

In G_c, $c \to \lambda$ and $d \to \lambda$ rules, delete all c and d letters. If the rules $S_m \to Q_{s1L}$, $S_c \to Q_{1R}$ were applied at the appropriate time of the calculation, all values of the G_c component will be deleted by these rules.

It is important to delete only those letters for which there is no rule, i.e., they will no longer change.

If the process stops at an inappropriate time, then at least one letter e must remain in 1R, that cannot be deleted, and apply the rule $e \to F$, which results in an error.

After the deletions, apply the $q_3 \to Q_c$ in the G_m component; if there is x in α where $x \neq q_3$, then apply the $x \to F$ rule. A trap letter will appear, indicating that the calculation was incorrect.

If the letter F does not appear and the entire content of the G_c component has been deleted, then after the communication step, the content of the G_m component (the master) will be α, which is an accepted word consisting of terminal letters.

$$(\alpha q_3, \beta, \alpha Q'_0, \lambda) \Rightarrow_{com}$$
$$(\alpha q_3, \beta, \alpha, \lambda).$$

4 Conclusion

Based on the example developed in section 3, we can assume that, using similar methods, we can create a PC ET0L system that can simulate its operation for other, even more complex, deeper non-cooperative polymorphic P systems. Our work is an initial step in finding the relationship between non-cooperative polymorphic P systems and parallel communicating ET0L systems.

Acknowledgements

Supported by the University of Debrecen Scientific Research Bridging Fund (DETKA).

References

1. Alhazov, A., Ivanov, S., Freund, R.: Polymorphic P systems: A survey. Bulletin of the International Membrane Computing Society **2** (2016) 79–101
2. Alhazov, A., Ivanov, S., Rogozhin, Y.: Polymorphic P systems. In Gheorghe, M., Hinze, T., Păun, G., Rozenberg, G., Salomaa, A., eds.: Membrane Computing. Volume 6501 of Lecture Notes in Computer Science., Berlin, Heidelberg, Springer-Verlag (2011) 81–94
3. Rozenberg, G., Salomaa, A., eds.: Handbook of Formal Languages. Springer-Verlag, Berlin Heidelberg (1997)
4. Păun, G.: Membrane Computing: An Introduction. Springer-Verlag, Berlin, Heidelberg (2002)
5. Păun, G., Rozenberg, G., Salomaa, A., eds.: The Oxford Handbook of Membrane Computing. Oxford University Press, Oxford (2010)

6. Lindenmayer, A.: Mathematical models for cellular interactions in development I. Filaments with one-sided inputs. Journal of Theoretical Biology **18**(3) (1968) 280–299
7. Păun, G.: Parallel communicating systems of L systems. In: Lindenmayer systems: impacts on theoretical computer science, computer graphics, and developmental biology. Springer Science and Business Media, Berlin, Heidelberg (1992) 405–418
8. Kuczik, A., Vaszil, G.: Simple variants of non-cooperative polymorphic P systems. Journal of Membrane Computing (to appear)
9. Vaszil, G.: On parallel communicating lindemayer systems. In Păun, G., Salomaa, A., eds.: Grammatical Models of Multi-Agent Systems. Volume 8 of Topics in Computer Mathematics., Gordon and Breach Science Publishers (1999) 99–112