Identifiable Kernel P Systems

Marian Gheorghe[1] and Florentin Ipate[2]

[1] School of Electrical Engineering and Computer Science
University of Bradford
Bradford BD7 1DP, UK
m.gheorghe@bradford.ac.uk
[2] Department of Computer Science
Faculty of Mathematics and Computer Science and ICUB
University of Bucharest
Str. Academiei 14, Bucharest 010014, Romania
florentin.ipate@ifsoft.ro

Abstract. This paper introduces the concept of *identifiability* for kernel P systems and establishes a fundamental set of properties for identifiable kernel P systems.

1 Introduction

Inspired by the structure and functioning of living cells, *membrane computing*, the research field initiated by Gheorghe Păun [14], has been intensively investigated in the last fifteen years. Its main research themes investigated so far refer to the computational power of different variants of membrane systems (also *called P systems*), hierarchies of languages or multiset classes produced by these devices, their capability to solve hard problems, decidability and complexity aspects [15]. There have also been significant developments in using the P systems paradigm to model various systems [2]. More recently, variants of P systems have been introduced in connection with modelling problems in various areas, e.g., systems and synthetic biology [5], information in biotic systems [17], synchronisation of distributed systems [3], grid algorithms [13], parallel algorithms utilised in 3D graphics [11].

Since P systems have been extensively used in various applications, it is a natural question to ask whether these applications and their implementations are correct and error-free. As *testing* is an essential part of software development and in many cases consumes large percentages of project efforts and budgets (test generation, in particular), it has been recently considered in the context of P systems applications. Some widely used testing approaches, such as mutation testing or transition cover have been considered [12] or adapted (rule coverage [6]) for P systems specifications. Recently a clas of generic P systems, called identifiable P systems, have been introduced and studied in connection with testing [9].

More recently, P systems have been used to model and simulate systems and problems from various areas. This has led to a multitude of P system variants. The newly introduced concept of *kernel P systems (kP systems, for short)* [7, 8] combines various features of P systems in a coherent manner, allowing to model complex applications and analyse these models.

The kP system model is supported by a modelling language, called kP-Lingua, capable of mapping a kP system specification into a machine readable representation. Furthermore, kP systems are supported by a software framework, kPWorkbench [10], which integrates a set of related simulation and verification methods and tools.

In this paper, we present the concept of identifiability for kernel P systems, continuing the research introduced in [9]. Our contribution has the following significant components:

- *Definition of identifiable kernel P systems,* a non-trivial extension of identifiable P systems.

– *A reformulation of some previous results regarding identifiable P systems in the context of kernel P systems.* In order to be self-contained some results regarding identifiable P systems have been slightly changed and presented in this paper. Some results are new, showing the capabilities of the kernel P systems.

The paper is structured as follows. Section 2 introduces basic concepts related to kernel P systems. Section 3 investigates the issue of identifiability in the context of the kernel P system model. Section 3 presents the main results, with complete proofs of the results introduced.

2 Preliminaries

Before proceeding, we introduce the notations used in the paper. For a finite alphabet $A = \{a_1, ..., a_p\}$, A^* denotes the set of all strings (sequences) over A. The empty string is denoted by λ and $A^+ = A \setminus \{\lambda\}$ denotes the set of non-empty strings. For a string $u \in A^*$, $|u|_a$ denotes the number of occurrences of a in u, where $a \in A$. For a subset $S \subseteq A$, $|u|_S$ denotes the number of occurrences of the symbols from S in u. The length of a string u is given by $\sum_{a_i \in A} |u|_{a_i}$. The length of the empty string is 0, i.e. $|\lambda| = 0$. A multiset over A is a mapping $f : A \to \mathbb{N}$. Considering only the elements from the support of f (where $f(a_{i_j}) > 0$, for some j, $1 \leq j \leq p$), the multiset is represented as a string $a_{i_1}^{f(a_{i_1})} \dots a_{i_p}^{f(a_{i_p})}$, where the order is not important. In the sequel multisets will be represented by such strings.

2.1 Kernel P System Basic Definitions

We start by introducing the concept of a *compartment type* utilised later in defining the compartments of a kernel P system (shortly called kP system).

Definition 1. *T is a* set of compartment types, *$T = \{t_1, \dots, t_s\}$, where $t_i = (R_i, \sigma_i)$, $1 \leq i \leq s$, consists of a set of rules, R_i, and an execution strategy, σ_i, defined over $Lab(R_i)$, the labels of the rules of R_i.*

The compartments that appear in the definition of the kP systems will be constructed using compartment types introduced by Definition 1. Each compartment, C, will be defined by a tuple (t, w), where $t \in T$ is the type of the compartment and w the initial multiset of it. The types of rules and the execution strategies occurring in the compartment types will be introduced and discussed later.

Definition 2. *A* kP system *of degree n is a tuple*

$$k\Pi = (A, \mu, C_1, \dots, C_n, i_0),$$

where A is a non-empty finite set, its elements are called objects*; μ defines the* initial membrane structure*, which is a graph, (V, E), where V is the set of vertices indicating compartments of the kP system, and E is the set of edges; $C_i = (t_i, w_i)$, $1 \leq i \leq n$, is a* compartment *of the kP system, as presented above; i_o, $1 \leq i_0 \leq n$, is the label of the* output compartment*, where the result is obtained.*

2.2 Kernel P System Rules

Each rule occurring in a kP system definition has the form $r : t \ \{g\}$, where r identifies the rule, is its **label**, t is the **rule** itself and g is its **guard**. The part t is also called the **body** of the rule, denoted also $b(r)$.

The guards are constructed using multisets over A, as operands, and relational or Boolean operators. The definition of the guards is now introduced. We start with some notations.

Let us denote $Rel = \{<, \leq, =, \neq, \geq, >\}$, the set of relational operators, $\gamma \in Rel$, a relational operator, and a^n a multiset, consisting of n copies of a. We first introduce an *abstract relational expression.*

Definition 3. *If g is the* abstract relational expression *denoting γa^n and w a multiset, then the guard g applied to w denotes the* relational expression $|w|_a \gamma n$.

The abstract relational expression g is true for the multiset w, if $|w|_a \gamma n$ is true.

We consider now the following Boolean operators $\neg$ (negation), $\wedge$ (conjunction) and $\vee$ (disjunction). An *abstract Boolean expression* is defined by one of the following conditions:

– any abstract relational expression is an abstract Boolean expression;
– if g and h are abstract Boolean expressions then $\neg g$, $g \wedge h$ and $g \vee h$ are abstract Boolean expressions.

The concept of a guard, introduced here, is a generalisation of the promoter and inhibitor concepts utilised by some variants of P systems.

Definition 4. *If g is an* abstract Boolean expression *containing g_i, $1 \leq i \leq q$, abstract relational expressions and w a multiset, then g applied to w means the* Boolean expression *obtained from g by applying g_i to w for any $i, 1 \leq i \leq q$.*

As in the case of an abstract relational expression, the guard g is true with respect to the multiset w, if the abstract Boolean expression g applied to w is true.

Example 1. If g is the guard defined by the abstract Boolean expression $\geq a^5 \wedge < b^3 \vee \neg > c$ and w a multiset, then g applied to w is true if it has at least 5 a's and no more than 2 b's or no more than one c.

In this paper we will use a simplified version of kP systems, called *simple kP systems*, using only rewriting and communication rules. In what follows these will be called simply kP systems.

We say that a rule $r : x \to y \ \{g\}$ is applicable to a multiset w iff $x \subseteq w$ and g is true for w.

Definition 5. *Any rule from a compartment $C_{l_i} = (t_{l_i}, w_{l_i})$ will be a* **rewriting and communication** *rule: $x \to y \ \{g\}$, where $x \in A^+$ and y has the form $y = (a_1, t_1) \ldots (a_h, t_h)$, $h \geq 0$, $a_j \in A$ and t_j, $1 \leq j \leq h$, indicates a compartment type from T (see Definition 1) associated with a compartment linked to the current one; t_j might also indicate the type of the current compartment, $C_{t_{l_i}}$; if a link does not exist (i.e., there is no link between the two compartments in E) then the rule is not applied; if a target, t_j, refers to a compartment type that appears in more than one compartments connected to C_{l_i}, then one of them will be non-deterministically chosen.*

2.3 Kernel P System Execution Strategies

In kP systems the way in which rules are executed is defined for each compartment type t from T – see Definition 1. As in Definition 1, $Lab(R)$ is the set of labels of the rules of R.

Definition 6. *For a compartment type $t = (R, \sigma)$ from T and $r \in Lab(R)$, $r_1, \ldots, r_s \in Lab(R)$, the* execution strategy*, σ, is defined by the following*

– *$\sigma = \lambda$, means no rule from the current compartment will be executed;*
– *$\sigma = \{r\}$ – the rule r is executed;*
– *$\sigma = \{r_1, \ldots, r_s\}$ – one of the rules labelled $r_1, \ldots, r_s$ will be non-deterministically chosen and executed; if none is applicable then nothing is executed; this is called* alternative *or* choice*;*

– $\sigma = \{r_1, \ldots, r_s\}^*$ – *the rules are applied an arbitrary number of times (*arbitrary parallelism*);*
– $\sigma = \{r_1, \ldots, r_s\}^\top$ – *the rules are executed according to the* maximal parallelism *strategy;*
– $\sigma = \sigma_1 \& \ldots \& \sigma_s$ – *this means executing sequentially* $\sigma_1, \ldots, \sigma_s$*, where* σ_i*,* $1 \leq i \leq s$*, describes any of the above cases; if one of* σ_i *fails to be executed then the rest is no longer executed.*

Definition 7. *A* configuration *of a kP system, kΠ, with n compartments, is a tuple* $c = (c_1, \ldots, c_n)$*, where* $c_i \in A^*$*,* $1 \leq i \leq n$*, is the multiset from compartment i. The* initial configuration *is* $(w_1, \ldots, w_n)$*, where* $w_i \in A^*$ *is the initial multiset of the compartment i,* $1 \leq i \leq n$*.*

A *transition* (or *computation step*), introduced by the next definition, is the process of passing from one configuration to another.

Definition 8. *Given two configurations* $c = (c_1, \ldots, c_n)$ *and* $c' = (c'_1, \ldots, c'_n)$ *of a kP system, kΠ, with n compartments, where for any* $i, 1 \leq i \leq n$*,* $u_i \in A^*$*, and a multiset of rules* $M_i = r_{1,i}^{n_{1,i}} \ldots r_{k_i,i}^{n_{k_i,i}}$*,* $n_{j,i} \geq 0$*,* $1 \leq j \leq k_i, k_i \geq 0$*, a* transition *or a* computation step *is the process of obtaining* c' *from* c *by using the multisets of rules* M_i*,* $1 \leq i \leq n$*, denoted by* $c \Longrightarrow^{(M_1,\ldots,M_n)} c'$*, such that for each i,* $1 \leq i \leq n$*,* c'_i *is the multiset obtained from* c_i *by first extracting all the objects that are in the left-hand side of each rule of* M_i *from* c_i *and then adding all the objects a that are in the right-hand side of each rule of* M_i *represented as* (a, t_i) *and all the objects b that are in the right-hand side of each rule of* M_j*,* $j \neq i$*, such that b is represented as* (b, t_i) *and an appropriate link between compartments exists.*

In the theory of kP systems, each compartment might have its own execution strategy. In the sequel we will consider that all the compartments will use the same execution strategy and the focus will be on three such execution strategies, namely maximal parallelism, arbitray parallelism (also called asynchronous execution) and sequential execution. These will be denoted by *max*, *async* and *seq*, respectively. When in a transition from c to c' using $(M_1, \ldots, M_n)$ we intend to refer to a specific transition mode *tm*, $tm \in \{max, async, seq\}$, this will be denoted by $c \Longrightarrow_{tm}^{(M_1,\ldots,M_n)} c'$.
A *computation* in a kP system is a sequence of transitions (computation steps).
A configuration is called *final configuration* if no rule can be applied to it. In a final configuration the computation stops.
As usual in P systems, we only consider terminal computations, i.e., those arriving in a final configuration and using one of the above mentioned transition modes. We are now ready to define the result of a computation.

Definition 9. *For a kP system kΠ using the transition mode tm,* $tm \in \{max, async, seq\}$*, in each compartment, we denote by* $N_{tm}(\Pi)$ *the number of objects appearing in the output compartment of any final configuration.*

Two kP systems *kΠ* and *kΠ′* are called *equivalent* with respect to the transition mode *tm*, $tm \in \{max, async, seq\}$, if $N_{tm}(k\Pi) = N_{tm}(k\Pi')$.
In this paper we will only deal with kP systems having *one single compartment* as this does not affect the general method introduced here and makes the presentation easier to follow. Indeed, limiting the investigation to one compartment kP systems does not affect the generality of the results due to the fact that there are ways of flattening an arbitrary P system, including the kP system discussed in this paper, into a P system with one single compartment. For details regarding the flattening of a P system we refer mainly to [4], but similar approaches are also presented in other papers ([16], [1]). Such a kP system will be denoted $k\Pi = (A, \mu_1, C_1, 1)$, where μ_1 denotes the graph with one node. On the right-hand side of the rules there will simply be multisets over A, as in the case of one single compartment there is no need to indicate where objects are sent to.
We now introduce the key concept we aim to investigate in this paper, namely *identifiability* of kP systems. This concept has been studied for generic P systems and is now investigated for kP systems where some additional constraints appear. The identifiability concept is first introduced for simple rules and then is generalised for multisets of rules.

Definition 10. *Two rules $r_1 : x_1 \to y_1 \ \{g_1\}$ and $r_2 : x_2 \to y_2 \ \{g_2\}$ from R_1, are said to be* identifiable *in configuration c if (i) they are applicable to c and (ii) if $c \Longrightarrow^{r_1} c'$ and $c \Longrightarrow^{r_2} c'$ then $b(r_1) = b(r_2)$. Otherwise the rules are said to be non identifiable.*

According to the above definition, the rules r_1 and r_2 are identifiable in c if whenever the result of applying them to c is the same, their bodies, $x_1 \to y_1$ and $x_2 \to y_2$, are identical.

A multiset $M = r_1^{n_1} \ldots r_k^{n_k}, M \in R_1^*$, where $r_i : x_i \to y_i \ \{g_i\}$, $1 \le i \le k$, is applicable to the multiset c iff $x_1^{n_1} \ldots x_k^{n_k} \subseteq c$ and g_i is true in c for any i, $1 \le i \le k$.

Notation. Given a multiset $M = r_1^{n_1} \ldots r_k^{n_k}$, where $r_i : x_i \to y_i \ \{g_i\}$, $1 \le i \le k$, we denote by r_M the rule $x_1^{n_1} \ldots x_k^{n_k} \to y_1^{n_1} \ldots y_k^{n_k} \{g_1 \wedge \cdots \wedge g_k\}$, i.e., the concatenation of all the rules in M.

Definition 11. *The multisets of rules $M', M'' \in R_1^*$, are said to be* identifiable*, if there is a configuration c where M' and M'' are applicable and if $c \Longrightarrow^{M'} c'$ and $c \Longrightarrow^{M''} c'$ then $b(r_{M'}) = b(r_{M''})$.*

Definition 12. *We say that a kP system $k\Pi$ has identifiable rules if any two multisets of rules, $M', M'' \in R_1^*$, are identifiable.*

3 Identifiable transitions in kP systems

In this section we investigate the property of identifiability for kP system rules and multisets of rules. We start by introducing a notation utilised in this section.

One can observe, based on Definition 11, that the applicability of the multiset of rules M to a certain configuration is equivalent to the applicability of the rule r_M to that configuration. It follows that one can study first the usage of simple rules.

Remark 1. For any two rules $r_i : x_i \to y_i \ \{g_i\}$, $1 \le i \le 2$, when we check whether they are identifiable or not one can write them as $r_i : uv_i \to wz_i \ \{g_i\}$, $1 \le i \le 2$, where for any $a \in A$, a appears in at most one of the v_1 or v_2, i.e., all the common symbols on the left-hand side of the rules are in u.

We first show that the identifiability of two rules does not depend on the configurations in which they are applicable. For the two rules introduced in Remark 1 let us denote by c_{r_1,r_2}, the configuration uv_1v_2. Obviously this is the smallest configuration in which r_1 and r_2 are applicable, given that g_1 and g_2 are true in uv_1v_2.

Remark 2. If $r_i : x_i \to y_i \ \{g_i\}$, $1 \le i \le 2$, are applicable in a configuration c and $c \subseteq c'$ then they are not always applicable to c'. They are applicable to c' when all g_i, $1 \le i \le 2$, are true in c'.

Remark 3. If the rules r_1, r_2 are not applicable to c_{r_1,r_2} then there must be minimal configurations c where the rules are applicable and they are minimal, i.e., there is no c_1, $c_1 \subset c$, where the rules are applicable. Such minimal configurations where r_1, r_2 are applicable are of the form tc_{r_1,r_2}, where $t \in A^*$, $t \neq \lambda$.

Lemma 1. *Two rules which are identifiable in a configuration c are identifiable in any configuration containing c in which they are applicable.*

Proof. Applying the two identifiable rules, $r_1 : x_1 \to y_1 \ \{g_1\}$ and $r_2 : x_2 \to y_2 \ \{g_2\}$, to the configuration c, one gets c' and c'' and $c' \neq c''$. If the rules are applicable to another configuration c_1 bigger than c, i.e, $c_1 = ct$, then $x_1, x_2 \subseteq c_1$ and g_1 and g_2 are true for c_1. In this case the results are $c_1' = c't$ and $c_1'' = c''t$ and obviously $c_1' \neq c_1''$, hence r_1 and r_2 are identifiable in c_1.

Lemma 2. *Two rules which are identifiable in a minimal configuration c are identifiable in any other minimal configuration c' in which they are applicable.*

Proof. According to Remark 3, the configurations c, c' can be written as $c = t_1 c_{r_1,r_2}$ and $c' = t_2 c_{r_1,r_2}$, where $t_1, t_2 \in A^*$, and $c \neq c'$ iff $t_1 \neq t_2$. According to Remark 1 the two rules can be written as $r_i : uv_i \rightarrow wz_i \ \{g_i\}$, $1 \leq i \leq 2$, where for any $a \in A$, a appears in at most one of the v_1 or v_2. The two rules are identifiable in $t_1 c = t_1 uv_1v_2$, i.e., $t_1 wz_1v_2 \neq t_1 wv_1z_2$. This is true iff $z_1v_2 \neq v_1z_2$. The rules are applicable in $c' = t_2uv_1v_2$, i.e., $uv_i \subseteq c'$ and g_i is true in c', $1 \leq i \leq 2$. Given that $z_1v_2 \neq v_1z_2$, it follows that $t_2wz_1v_2 \neq t_2wv_1z_2$. This means that the two rules are identifiable in $c' = t_2uv_1v_2 = t_2c_{r_1,r_2}$.

Corollary 1. *Two rules r_1 and r_2 identifiable in a minimal configuration tc_{r_1,r_2}, $t \in A^*$, are identifiable in any configuration in which they are applicable.*

Proof. The result is an immediate consequence of Lemmas 1 and 2, and Remarks 1, 2 and 3.

One can formulate a similar result for two multisets of rules.

Corollary 2. *Two multisets of rules M_1 and M_2 identifiable in $tc_{r_{M_1},r_{M_2}}$, $t \in A^*$, are identifiable in any configuration in which they are applicable.*

Proof. The result is an immediate consequence of Corollary 1 and Notation above.

From now on, we will always verify the identifiability (or non identifiability) only for the smallest configurations associated with rules or multisets of rules and will not mention these configurations anymore in the results to follow.

The applicability of two rules (multisets of rules) to a certain configuration depends not only on the fact that their left hand sides (the concatenation of the left hand sides) must be contained in the configuration and the gards must be true, but takes into account the execution strategy.

Remark 4. For the *async* transition mode two multisets of rules (and two rules) applicable in a configuration are also applicable in any other bigger configuration, when the corresponding guards are true. For the *seq* mode this is true only for multisets with one single element and obviously for simple rules. In the case of the *max* mode the applicability of the multisets of rules (or rules) to various configurations depends on the contents of the configurations and other available rules. For instance if we consider a kP system containing the rules $r_1 : a \rightarrow a \ \{\geq a\}; r_2 : ab \rightarrow abb \ \{\leq b^{100}\}; r_3 : bb \rightarrow c \ \{\geq b^2\}$ and the configuration $c = ab$ then in c only r_1 and r_2 are applicable and identifiable, but in $c_1 = abb$, containing c, r_1 is no longer applicable, but instead we have r_2 and the multiset r_1r_3 applicable. In ab^{101} r_2 and any multiset containing it are not applicable due to the guard being false; also r_1 is no longer applicable, but $r_1r_3^{55}$ is now applicable, due to maximal parallelism.

Remark 5. In the following results whenever we refer to arbitrary rules or multisets of rules they are always meant to be applicable with respect to the transition mode.

We now provide a characterisation of the two rules to be (non) identifiable.

Theorem 1. *The rules $r_1 : x_1 \rightarrow y_1 \ \{g_1\}$ and $r_2 : x_2 \rightarrow y_2 \ \{g_2\}$, are not identifiable if and only if they have the form $r_1 : uv_1 \rightarrow wv_1 \ \{g_1\}$ and $r_2 : uv_2 \rightarrow wv_2 \ \{g_2\}$ and for any $a \in A$, a appears in at most one of v_1 or v_2.*

Proof. Let us start with this implication "$\Longrightarrow$". As we have already discussed one can use the rules as $r_1 : uv_1 \rightarrow y_1 \ \{g_1\}$ and $r_2 : uv_2 \rightarrow y_2 \ \{g_2\}$ and for any $a \in A$, a appears in at most one of v_1 or v_2; and one can consider one of the smallest configurations where they are applicable, $tc_{r_1,r_2} = tuv_1v_2$, where $t \in A^*$. Applying these rules to tc_{r_1,r_2}, the following computations are obtained:

$$tc_{r_1,r_2} \Longrightarrow^{r_1} ty_1v_2; tc_{r_1,r_2} \Longrightarrow^{r_2} ty_2v_1.$$

As these rules are not identifiable it turns out that the results of the two computations are the same, i.e., $ty_1v_2 = ty_2v_1$ and this is true iff $y_1v_2 = y_2v_1$. Given that for any $a \in A$, a appears in at most one of v_1 or v_2, it follows that y_1 contains v_1 and y_2 contains v_2, i.e., $y_1 = w_1v_1$ and $y_2 = w_2v_2$. From the equality of the results of the computations it follows that $w_1 = w_2 = w$ and this proves the result.

Let us consider the opposite "$\Longleftarrow$". In this case the rules are $r_1 : uv_1 \to wv_1\ \{g_1\}$, $r_2 : uv_2 \to wv_2\ \{g_2\}$ and for any $a \in A$, a appears in at most one of v_1 or v_2. We consider again one of the smallest configurations where the rules are applicable, $tc_{r_1,r_2} = uv_1v_2$, $t \in A^*$, and apply the two rules; then one can obtain:

$$tc_{r_1,r_2} \Longrightarrow^{r_1} twv_1v_2; tc_{r_1,r_2} \Longrightarrow^{r_2} tv_1wv_2.$$

Hence, r_1 and r_2 are not identifiable.

The above proof assumes that v_1 and v_2 are not empty multisets. The result remains true when one of them or both are empty. In the latter case we have the same body of the rules, which might have the right-hand side λ.

Based on the result provided by Theorem 1 one can state when two rules are identifiable.

Corollary 3. *The rules $r_1 : uv_1 \to wz_1\ \{g_1\}$ and $r_2 : uv_2 \to wz_2\ \{g_2\}$, such that for any $a \in A$, a appears in at most one of v_1 or v_2, are identifiable if and only if $v_1 \neq z_1$ or $v_2 \neq z_2$.*

With the results obtained so far one can determine, for a kP system, whether any two rules are identifiable or not. In various transition modes utilised in kP systems – maximal parallelism or asynchronous mode – in any computation step either single rules or multisets of rules are involved. It is therefore important to determine whether the identifiability of single rules can be lifted to multisets of rules. More precisely, we want to know whether it is true that the identifiability of any pair of simple rules is inherited by the multisets of rules. Unfortunately, this is not true in general, as it is shown by the next example.

Example 2. Let us consider a P system with the following four rules: $r_1 : a \to b\ \{\geq a\}$, $r_2 : b \to a\ \{\geq b\}$, $r_3 : c \to d\ \{\geq c\}$, $r_4 : d \to c\ \{\geq d\}$. According to Corollary 3, any two rules are identifiable, but $M_1 = r_1r_2$ and $M_2 = r_3r_4$ are not, as $r_{M_1} : ab \to ab\ \{\geq a \land \geq b\}$ and $r_{M_2} : cd \to cd\ \{\geq c \land \geq d\}$ are identity rules and according to Theorem 1 they are not identifiable.

However, one can show that some particular multisets of rules are identifiable when their components are. More precisely, we have the following result.

Theorem 2. *If r_1 and r_2 are identifiable then r_1^n and r_2^n are identifiable, for any $n \geq 1$.*

Proof. According to Corollary 3 the rules can be written $r_1 : uv_1 \to wz_1\ \{g_1\}$ and $r_2 : uv_2 \to wz_2\ \{g_2\}$, such that for any $a \in A$, a appears in at most one of v_1 or v_2, and $v_1 \neq z_1$ or $v_2 \neq z_2$. This implies that $v_1^n \neq z_1^n$ or $v_2^n \neq z_2^n$, for any $n \geq 1$, i.e., r_1^n and r_2^n are identifiable.

One can show that identifiability of any two multisets of rules can be achieved in some special circumstances. More precisely, we show that for any kP system one can construct an equivalent kP system, in which the rules are slightly modified and an additional one introduced, and the later has any multiset of rules identifiable (according to Definition 12).

We first make a notation based on some simple observations. Given a rule $r : x \to y\ \{g\}$, this is applicable in a configuration w iff $x \subseteq w$ and g is true in w. It follows that the rule is not applicable in w iff $x \nsubseteq w$ or g is not true in w. If $x = a_1^{n_1}...a_h^{n_h}$ and g_x denotes the abstract Boolean expression, $\geq a_1^{n_1} \dots a_h^{n_h}$, then $x \subseteq w$ iff g_x is true in w and $x \nsubseteq w$ when g_x is false in w.

One can now formulate a result similar with Theorems 2 and 3 in [9], but stronger than those given the context provided by the kP systems model.

Theorem 3. *For any kP system $k\Pi$ there is a kP system $k\Pi_L$ such that (i) $N_{tm}(\Pi) = N_{tm}(\Pi_L)$, for any of the transition modes tm, $tm \in \{max, async, seq\}$, and (ii) $k\Pi_L$ has identifiable rules.*

Proof. (i) Let us consider a kP system $k\Pi = (A, \mu_1, C_1, 1)$, as introduced in Definition 2. The compartment $C_1 = (t_1, w_1)$ and $t_1 = (R_1, tm)$, $tm \in \{max, async, seq\}$.

We build the following kP system

$$k\Pi_L = (A', \mu_1, C'_1, 1)$$

where $A' = A \cup Lab(R_1) \cup \{\#\}$, with $Lab(R_1) = \{r \mid r : x \to y \ \{g\} \in R_1\}$ and $\#$ a new symbol. $C'_1 = (t'_1, w_1)$ and $t'_1 = (R'_1, tm)$. In order to define R'_1 we make a few notations.

Let us consider R_1 denoting the set of rules $\{r_1 : x_1 \to y_1 \ \{g_1\}, \dots, r_p : x_p \to y_p \ \{g_p\}\}$. For each rule $r_i : x_i \to y_i \ \{g_i\}$, we consider the abstract Boolean expression, g_{x_i}, introduced above for the left-hand side of the rule r_i, $1 \le i \le p$. We now denote g_{R_1} the abstract Boolean expression $(g_{x_1} \wedge g_1) \vee \dots \vee (g_{x_p} \wedge g_p)$. One can observe that g_{R_1} ($\neg g_{R_1}$) is true (false) in a multiset w iff at least a rule (none of the rules) of R_1 is applicable to w.

The multiset R'_1 contains the following rules

- (1) $r'_i : x_i \to y_i r_i \ \{g_i \wedge < \#\}$, $r''_i : x_i \to y_i r_i \# \ \{g_i \wedge < \#\}$, for $r_i : x_i \to y_i r_i \ \{g_i\} \in R_1$, $1 \le i \le p$.
- (2) $r'''_i : r_i \to \lambda \ \{\ge \#\}$, $1 \le i \le p$.
- (3) $r'_{p+1} : \# \to \lambda \ \{\neg g_{R_1}\}$.
- (4) $r'_{p+2} : \# \to \# \ \{g_{R_1}\}$.

In order to show that $k\Pi$ and $k\Pi_L$ are equivalent for the transition mode tm, $tm \in \{max, async, seq\}$, one can observe that for any terminal computation in $k\Pi$

$$u_0 = w_1 \Longrightarrow_{tm}^{M_1} u_1 \dots u_{n-1} \Longrightarrow_{tm}^{M_n} u_n$$

there is a terminal computation in $k\Pi_L$ and vice versa. Firstly, one can obtain

$$u'_0 = w_1 \Longrightarrow_{tm}^{M'_1} u'_1 = u_1 Lab(M_1) \dots u'_{n-1} = u_{n-1} Lab(M_1) \dots Lab(M_{n-1})$$

$$\Longrightarrow_{tm}^{M'_n} u'_n = u_n Lab(M_1) \dots Lab(M_{n-1}) Lab(M_n) \#^h,$$

where $Lab(M)$ denote the multiset of labels of rules occurring in M. M'_i, $1 \le i \le n-1$, is obtained from M_i, $1 \le i \le n-1$, by replacing each rule $r \in R_1$ by its corresponding $r' \in R'_1$; M'_n is obtained from M_n, with r' rules and h r'' rules ($h \ge 1$) introducing $\#^h$. Then we have the following cases

- $tm = max$, implies $u'_n = u_n Lab(M_1) \dots Lab(M_{n-1}) Lab(M_n) \#^h \Longrightarrow_{tm}^{M'_{n+1}} u'_{n+1} = u_n$, where the multiset of rules M'_{n+1} includes rules of type (2) that remove the labels of the rules $Lab(M_1) \dots Lab(M_n$ and rule (3) erasing $\#^h$.
- when $tm = async$, the rules of the multiset M'_{n+1} may be applied in more than a step leading to the same result, u_n.
- for $tm = seq$, we have $h = 1$ and as only one rule is applied in a step, u_n is obtained after a number of step given by the cardinal of M'_{n+1} .

One can observe that in any computation in $k\Pi_L$ in the first steps only rules r'_i, $1 \le i \le p$ of type (1) are applicable. When at least a rule r''_i, $1 \le i \le p$ of type (1) is applied, a symbol $\#$ is introduced. In the next step the rules of type (2) are applicable. The rule (3) is used only when none of the rules of type (1) are applicable, i.e., a computation simulating a terminal computation in $k\Pi$. Otherwise, a rule of type (4) is applicable, leading to an infinite computation.

(ii) Let us consider two multisets of rules, M_1 and M_2, applicable to a configuration in given transition mode. Using a notation introduced earlier, one can obtain the rules, r_{M_1} and r_{M_2}, associated with the multisets of rules, and Remark 1 for providing the following format of them:

– $r_{M_i} : uv_i \rightarrow wz_i Lab(M_i)\ \{g_i\}, 1 \leq i \leq 2$, when rules of type (1) are used; and
– $r_{M_i} : uv_i \rightarrow wz_i\ \{g_i\}, 1 \leq i \leq 2$, otherwise.

Obviously, $v_i \neq z_i Lab(M_i)$, $1 \leq i \leq 2$, in the first case, and $v_i \neq z_i$, $1 \leq i \leq 2$, in the last one. According to Corollary 3 the rules are identifiable and consequently the multisets of rules, M_1 and M_2.

4 Conclusions

This paper extends the concept of identifiability, previously introduced in the context of cell-like P systems, to kernel P systems and establishes a fundamental set of properties for identifiable kP systems. Future work will aim at developing a testing approach for kernel P systems. This will be based on automata and X-machine theory and will be able to determine that, under certain, well defined conditions, the implementation conforms to the specification.

Acknowledgements

This work is supported by a grant of the Romanian National Authority for Scientific Research, CNCS-UEFISCDI, project number PN-III-P4-ID-PCE-2016-0210.

References

1. Agrigoroaiei, O., Ciobanu, G.: Flattening the transition P systems with dissolution. In: Proceedings of the 11th International Conference on Membrane Computing. pp. 53–64. CMC'10, Springer-Verlag (2010)
2. Ciobanu, G., Pérez-Jiménez, M.J., Păun, Gh.: Applications of Membrane Computing (Natural Computing Series). Springer-Verlag (2005)
3. Dinneen, M.J., Yun-Bum, K., Nicolescu, R.: Faster synchronization in P systems. Natural Computing 11(4), 637–651 (2012)
4. Freund, R., Leporati, A., Mauri, G., Porreca, A.E., Verlan, S., Zandron, C.: Flattening in (tissue) P systems. In: Revised Selected Papers of the 14th International Conference on Membrane Computing - Volume 8340. pp. 173–188. CMC 2013, Springer-Verlag (2014)
5. Frisco, P., Gheorghe, M., Pérez-Jiménez, M.J. (eds.): Applications of Membrane Computing in Systems and Synthetic Biology. Springer-Verlag; Emergence, Complexity and Computation Series, Vol. 7 (2014)
6. Gheorghe, M., Ipate, F.: On testing P systems. In: Membrane Computing, pp. 204–216. Springer-Verlag (2009)
7. Gheorghe, M., Ipate, F., Dragomir, C.: Kernel P systems. In: Pre-Proc. 10th Brainstorming Week on Membrane Computing, pp. 153–170. BWMC10, Fénix Editora, Universidad de Sevilla (2012)
8. Gheorghe, M., Ipate, F., Dragomir, C., Mierlă, L., Valencia-Cabrera, L., García-Quismondo, M., Pérez-Jiménez, M.J.: Kernel P systems – version 1. In: Pre-Proc. 11th Brainstorming Week on Membrane Computing
9. Gheorghe, M., Ipate, F., Konur, S.: Testing based on identifiable P systems using cover automata and X-machines. Information Sciences 372, 565–578 (2016)
10. Gheorghe, M., Konur, S., Ipate, F., Mierlă, L., Bakir, M., Stannett, M.: An integrated model checking toolset for kernel P systems. In: Revised Selected Papers of the 16th International Conference on Membrane Computing - Volume 9504. pp. 153–170. CMC 2015, Springer-Verlag (2015)
11. Gimel'farb, G.L., Nicolescu, R., Ragavan, S.: P system implementation of dynamic programming stereo. Journal of Mathematical Imaging and Vision 47(1–2), 13–26 (2013)
12. Ipate, F., Gheorghe, M.: Mutation based testing of P systems. International Journal of Computers, Communication and Control 4(3), 253–262 (2009)

13. Nicolescu, R.: Structured grid algorithms modelled with complex objects. In: Revised Selected Papers of the 16th International Conference on Membrane Computing - Volume 8340. pp. 56–79. CMC 2013, Springer-Verlag (2014)
14. Păun, Gh.: Computing with membranes. Journal of Computer and System Sciences 61(1), 108–143 (2000)
15. Păun, Gh., Rozenberg, G., Salomaa, A.: The Oxford Handbook of Membrane Computing. Oxford University Press, Inc. (2010)
16. Verlan, S.: Using the formal framework for P systems. In: Revised Selected Papers of the 14th International Conference on Membrane Computing - Volume 9504. pp. 321–337. CMC 2015, Springer-Verlag (2015)
17. Vincenzo, M.: Infobiotics. Springer-Verlag; Emergence, Complexity and Computation Series, Vol. 3 (2013)