Notes on Language Relations among Transition P Systems

Richelle Ann B. Juayong, Henry N. Adorna,
Kelvin C. Buño, Francis George C. Cabarle
Algorithms and Complexity Laboratory
University of the Philippines Diliman
Quezon City, Philippines
{rbjuayong,fccabarle}@up.edu.ph
{hnadorna,kcbuno}@dcs.upd.edu.ph

ABSTRACT

We explore language relations of specific Transition P (TP) systems without membrane dissolution. The main models we investigate are TP systems having the following characteristic: for every cooperative rule that needs a multiset u, each object a (also called a *trigger*) in the multiset u can also evolve through a non-cooperative rule defined in the same region. We call such models as TP systems having independent triggers only (or TP-ind systems). A TP system having only non-cooperative rules is a special case of a TP-ind system. In the case of TP-ind system, triggers are said to be *independent* since every rule trigger can also evolve independently. Otherwise, a (cooperative) rule trigger is said to be *dependent*. In this paper, we look at characteristics of some TP systems Π with dependent triggers (or TP-dep systems) wherein the language of Π can also be generated by a TP-ind system Π'.

Keywords: Membrane computing, Transition P systems, String languages

1. INTRODUCTION

Transition P (TP) systems is a well-known cell-like computing model that use multiset-rewriting rules with target indication. As one of the earliest models introduced in [7], there is already plenty of literature on the different aspects of TP systems, e.g. capability and complexity as a computing model. These aspects were investigated while also considering restrictions and additional features, e.g. restricted forms of rules, rule priorities are employed, or membrane operations are allowed (e.g. in [1, 3, 8, 7]).

In this study, we contribute to previous studies by exploring language-based relations of some restricted TP systems. We explore TP systems with cooperative rules restricted as follows: (a) at least one trigger is independent, and (b) rules triggered by several copies of only one dependent object (i.e., not involved in a cooperative rule). Our goal is to construct TP-ind systems for TP systems with rules that have cooperative rules of the form (a) and/or (b). The results we obtain here extend to other cell-like models with similar multiset-rewriting rules. We note that in our TP systems, we only consider membranes that are static all throughout the computation. We do not employ dissolution rules i.e. rules that can dissolve membranes.

This paper is organized as follows: Section 2 provides preliminaries, definitions of TP systems, independent and dependent triggers. Our main work is detailed in Section 3. Conclusions are given in Section 4.

2. PRELIMINARIES

It is assumed that the readers are familiar with the basics of membrane computing. A good reference is [8], an introduction can be found in [9], with recent results at `http://ppage.psystems.eu`. Notions and notations in formal language theory [5] as presented in [8] are also used. We only briefly mention some which will be useful throughout the paper.

Let V be an alphabet, V^* is the free monoid over V with respect to concatenation and the identity element λ (empty string). The set of all non-empty strings over V is denoted as V^+ so $V^+ = V^* - \{\lambda\}$. The length of a string $w \in V^*$ is denoted by $|w|$. For $a \in V$, $|w|_a$ represents the number of a in string w.

Let U be an arbitrary set. A multiset (over U) is a mapping $M : U \to \mathbf{N}$. The value $M(a)$, for $a \in U$, is the *multiplicity of a in the multiset M*. The *support* of a multiset M is the set $supp(M) = \{a \in U \mid M(a) > 0\}$. A multiset M is empty when its support is empty (it is then denoted by $\emptyset$). A multiset M of finite support can also be represented by a string: $w = a_1^{M(a_1)} a_2^{M(a_2)} \dots a_n^{M(a_n)}$ where $a_i \in supp(M)$, $1 \leq i \leq n$. In string w and all its permutation, $|w|_{a_i} = M(a_i)$. Thus, string w and all its permutations precisely identify and refer to the same multiset M.

We denote by $Perm(w)$ all permutations of a string w. We use the phrase "multiset w" where w is a string to refer to the multiset represented by the string w. Note that while a string is ordered, a multiset is unordered.

2.1 Transition P systems (TP systems)

We provide a general definition of TP systems, and describe how they generate a language. We define a TP system without dissolution, similar to [1] as follows:

DEFINITION 1. *A Transition P (TP) system without dissolution is a construct of the form* $\Pi = (O, \mu, w_1, \ldots, w_m, R_1, \ldots, R_m, h_{out})$ *where:*

(i) m is the total number of membranes;

(ii) O is the alphabet of objects;

(iii) μ is the membrane structure hierarchy denoted by a string of matching square brackets with labels. A membrane h containing a membrane j is a *parent membrane* if j is immediately contained in h i.e. $[_h \ldots [_j \,]_j]_h$. Consequently, j is a *child membrane* of membrane h. A *region* refers to the area delimited by a membrane. We say region h is delimited by membrane h.

(iv) $w_1, \ldots, w_m$ are strings over O^* where w_h, $1 \leq h \leq m$, denotes the multiset of objects present in the region h.

(v) $R_1, \ldots, R_m$ are finite sets of multiset-rewriting rules, each associated with a region in μ. A rule takes the form $u \to v$ where multiset $u \in O^+$, $v \in (O \times Tar_h)^*$ and $Tar_h = \{here, out\} \cup \{in_j \mid j \text{ is a child membrane of } h\}$, $1 \leq h \leq m$. A rule with $|u| > 1$ is said to be cooperative; rules having $|u| = 1$ are non-cooperative. A rule $u \to v$ labelled r is denoted by $r : u \to v$. We say that given a rule r, $LHS(r)$ is the multiset u.

A rule $r \in R_h$ is applicable if a multiset u is present in region h. Applying rule r means the multiset u is removed from region h and a multiset of objects v is produced in the next time step. Symbols *here*, *out* and in_j indicate the destination of the objects produced (target *here* is typically omitted).

In our illustrations, we use strings in the right-hand side of rule r. If v contains (v', tar) where $v' \in O^+$, $tar \in Tar_h$, this means that the string v' is placed in the region corresponding to tar.

(vi) $h_{out} \in \{0, 1, \ldots, m\}$ is the output membrane. If $h_{out} = 0$, this means that the environment (region outside the skin) is the placeholder of the output.

The system runs by applying rules in the multiset present in each region. We use the term 'copy of object a' to refer to an instance of object a present in a multiset w, (e.g. multiset a^2b contains two copies of a and one copy of b). Given multisets in all region, applying rules is done in a non-deterministic and maximally parallel manner. Non-determinism implies that at a certain step, if there are more than two rules that can be applied to a copy of an object, the system non-deterministically chooses the rule to be applied for each copy of an object. For example, say a region contains rules $r : a \to ba$ and $r' : ab \to bc$ and a copy of objects a and b. For the copy of a, only one of rules r and r' can be used to consume a; when r' is chosen, the copy of object b is consumed as well. Maximal parallelism implies that all copies of objects that can evolve must do so. To illustrate, suppose our example above contains a different multiset, say a^2b. Using r', copy of a and copy of b evolves, however, the remaining copy of a can still evolve through rule r. Thus, to satisfy maximal parallelism, both rules r and r' must be applied once. Due to non-determinism, it is possible that all copies of a evolve via rule r. In such case, a copy of b remains unevolved. Two application of rule r still satisfies maximal parallelism since there's no rule that requires only a single copy of b.

A configuration at any time t denoted by C_t, is the state of a system: it consists of the multiset of objects within each region. One way to represent a configuration is by using a string of matching square brackets with labels, as in the representation of μ. A multiset w contained inside a pair of brackets labelled h indicate that the multiset is in region h, e.g. the notation $[_h w [_g w']_g]_h$ represents that multiset w is in region h while multiset w' is in region g. (Also, membrane g is contained in membrane h).

A transition from C_t to C_{t+1} through non-deterministic and maximally parallel manner of rule application can be denoted as $C_t \Rightarrow C_{t+1}$. A series of transitions is said to be a computation, denoted as $C_t \Rightarrow^* C_{t'}$ for $t < t'$. Computation succeeds when the system halts, i.e. when the system reaches a configuration wherein none of the rules can be applied. This configuration is called a *halting configuration*. If there is no halting configuration, i.e. if the system does not halt, computation fails because the system did not produce any output.

We only consider models here that function as string generators. In this case, we follow [1] wherein the result of a successful computation is the sequence of objects sent to the environment, i.e. $h_{out} = 0$. The order of how objects are sent to the environment dictates their position in the output string. For example, if objects a and b are sent to the environment at times t and t', respectively $(t < t')$, then in the output string, the position of a is lower than the position of b. In the category where multiple objects are sent at a given time, the output string is formed from any of their permutations. For example, if objects a and b are sent to the environment at the same time, the resulting strings s and s' have ab and ba as substrings, respectively. The language generated by Π is denoted by $L(\Pi)$.

EXAMPLE 1. *Let* Π_y *be a TP system where:*

$$\Pi_y = (\{a, b, S\}, [_1[_2]_2]_1, S, \lambda, \{r_1 : S \to aSb, r_2 : S \to (S, in_2), r_3 : a \to (a^2, out), r_4 : b \to (b^2, out), r_5 : ab \to (ab, out)\}, \lambda, 0)$$

An example computation of Π_y *is given by the sequence:* $[_1 S [_2]_2]_1 \Rightarrow [_1 aSb [_2]_2]_1$ *through rule* r_1, $[_1 aSb]_1 \Rightarrow [_1 aSb]_1$ *through rules* r_1, r_3 *and* r_4 *outputing* a^2b^2 *in the environment,* $[_1 aSb]_1 \Rightarrow [_1]_1$ *through rules* r_2 *and* r_5 *outputing* ab *in the environment. This computation thus generates* $Perm(a^2b^2)Perm(ab)$. *In particular, this refers to set of strings* $\{a^2b^3a, a^2b^2ab, abab^2a, ababab, ba^2b^2a, ba^2bab, bababa, baba^2b, b^2a^2ba, b^2a^3b\}$.

A general description of computations of Π_y *proceeds as follows: Initially, region* 1 *has a single copy of object* S. *In the*

next step, S can continuously evolve through r_1 until r_2 is applied. In which case, S is sent to region 2. Since no rule is triggered by S, S in region 2 will no longer evolve. Each time S in region 1 evolves through the former rule, a copy of both a and b are produced. These copies of objects can either be consumed through rule r_5 or are evolved independently via rules r_3 and r_4. Therefore, $L(\Pi_y) = (Perm(a^2b^2) \cup Perm(ab))^*$.

EXAMPLE 2. *Shown below is another TP system example:*

$$\Pi_z = (\{S, \bar{S}, a, b\}, [_1]_1, S, \{r_1 : S \to Sb(a, out), r_2 : S \to \bar{S}, r_3 : \bar{S}b \to \bar{S}(b, out)\}, 0)$$

An example computation of Π_z is given by the sequence: $[_1 S]_1 \Rightarrow [_1 Sb]_1$ through rule r_1 outputing a copy of a, $[_1 Sb]_1 \Rightarrow [_1 \bar{S}b]_1$ through rule r_2, $[_1 \bar{S}b]_1 \Rightarrow [_1 \bar{S}]_1$ through rule r_3 outputing a copy of b. This computation generates the string ab.

A general description of computations of Π_z proceeds as follows: In the initial configuration, a copy of a is in region 1. The applicable rules for the copy of S are r_1 and r_2. Through repeated use of rule r_1, some copies of a are sent out to the environment while the same copies of b remain in region 1. Upon use of rule r_2, the copy of S evolves to $\bar{S}$. The b's in region 1 are then sent out in the next steps sequentially through repeated use of rule r_3. When no more b's are sent out, the system halts since rule r_3 can no longer be applied. Since the system generates the same number of a's and b's but all a's are sent out before b's, $L(\Pi_z) = \{a^n b^n \mid n \geq 0\}$.

LEMMA 1. *Any language generated by a TP system with m membranes can be generated by a TP system with only one membrane.*

PROOF. We use the known technique for flattening membrane in P systems (given e.g. in [1, 4, 10]). Let TP system with m membranes be $\Pi = (O, \mu, w_1, \ldots, w_m, R_1, \ldots, R_m, 0)$. We construct another TP system $\Pi' = (O', [_1]_1, w_1', R_1', 0)$ from Π. For every $a \in O$ and every region h, $1 \leq h \leq m$, we define an object $a_h \in O'$. For every rule $r : u \to v$ in region h, we create a rule $r' : u' \to v' \in R_1'$. For every b in u, there is a b_h in u'. For every $(b, here)$ in v, there is a b_h in v'. For every (b, out) in v, there is a b_k in v' where k is the label of parent membrane of h. For every (b, in_j) in v, there is a b_j in v'. The rule r in Π is simulated by its counterpart rule r' in Π'. For every copy of object a consumed (produced) in region h of Π, a copy of object a_h in region 1 of Π' is consumed (produced). As can be observed, there is a unique correspondence between configurations of Π and Π' and the output in the environment is the same. □

2.2 TP-ind systems and TP-dep systems

In our study, we divide TP systems based on the role of so-called rule triggers. Given a TP system, a trigger corresponds to an object that exists on the left-hand side (LHS) of a rule.

DEFINITION 2. **(Trigger, Independent Trigger, Dependent Trigger)** *Given TP system Π, an object $o \in O$ is a trigger in a region h if there exists a rule $r \in R_h$ and object o is in the LHS of rule r. If $r : u \to v$, then object o is in multiset u. Object o is an* **independent trigger in region** *h if there exists a rule $r' \in R_h$ that is a non-cooperative rule ($r' : o \to v'$). Otherwise, object o is said to be a* **dependent trigger in region** *h. When the region is clear, we say that either a trigger is dependent or independent.*

DEFINITION 3. **(coop-ind rule, coop-dep rule)** *Let Π be a TP system. A cooperative rule having independent triggers only (or coop-ind rule) is a cooperative rule $r : u \to v$ defined in Π where all object o in multiset u are independent triggers. If at least one object o' in multiset u is a dependent trigger, then we say that r is a cooperative rule with dependent triggers (or coop-dep rule).*

DEFINITION 4. **(TP-ind system, TP-dep system)** *A TP system having independent triggers only (or TP-ind system) is a TP system Π where all triggers in all regions are independent triggers. This implies that if Π contains cooperative rules, then these rules are all coop-ind rules. A TP system with dependent triggers (or TP-dep system) is a TP system that contains at least one coop-dep rule.*

We now examine the TP system given in Examples 1 and 2. In TP system Π_y, all rules are non-cooperative except for rule r_5. The triggers for this rule are objects a and b. For triggers a and b, there exist non-cooperative rules r_3 and r_4, respectively. Thus, r_5 is a coop-ind rule and Π_y is a TP-ind system. In contrast, Π_z is a TP-dep system; rule r_3 of Π_y is a coop-dep rule.

TP-ind systems require that all cooperative rules are coop-ind rules. Note however that coop-ind rules can also occur in TP-dep systems. Notice that independent triggers are always consumed when some copies occur in a configuration; this means, TP-ind system continues evolving as long as a trigger exists since all triggers are independent. On the other hand, dependent triggers can either be consumed or can be carried over (when the other triggers of the involved cooperative rule is not available). This implies that the halting configuration of a TP-ind system doesn't have triggers whereas some dependent triggers may occur in the halting configuration of a TP-dep system. Further details are provided in the next section.

3. TP-IND SYSTEMS VS. TP-DEP SYSTEMS

In this section, we show that the language of some TP-dep systems can be generated by a TP-ind system. First, we provide an analysis of coop-dep rules in the former model.

DEFINITION 5. **(Categories for coop-dep rules)** *Given a TP-dep system Π, a coop-dep rule in a region h can be classified as one of the following:*

cat 1: *A cooperative rule $u \to v \in R_h$ must have some dependent trigger in multiset u but at least one object o in u is an independent trigger.*

cat 2: *A cooperative rule $o^k \to v \in R_h$ ($k \geq 2$) has object o not an independent trigger.*

cat 3: *A cooperative rule $u \to v \in R_h$ has at least two distinct objects involved in multiset u and no object in u is an independent trigger.*

Use of any of these categories imply that a TP system is a TP-dep system. We denote by $LTP(ind)$ the family of languages generated using TP-ind systems. We denote by $LTP(dep)$ the family of languages generated using TP-dep systems. If all coop-dep rules are of cat x ($x \in \{1,2,3\}$), we use the notation $LTP(dep, C_x)$. If two of the three cases are used for cooperative rules, we use the notation $LTP(dep, C_{xy})$ where $x, y \in \{1,2,3\}$, $x \neq y$.

In what follows, we distinguish TP-dep systems from TP-ind systems by appending an apostrophe ($'$) in the definition of the latter. We also use apostrophe to distinguish between configurations and rules in the two systems. For example, Π refers to a TP-dep system while Π' pertains to a TP-ind system. The next three examples are TP-dep systems that will be used in the next subsections.

EXAMPLE 3. $\Pi_{x1} = (\{a,b,c\}, [_1]_1, abc, \{r_1 : ab \to (b^3, out), r_2 : a \to (a, out), r_3 : ac \to (b^2, out), r_4 : c \to (a^2, out)\}, 0)$

TP-dep system Π_{x1} contains two cooperative rules r_1 and r_3. While rule r_3 is a coop-ind rule, rule r_1 is a coop-dep rule. This rule is triggered by a copy of object a and b. The former is an independent trigger (due to rule r_2) while the latter is a dependent trigger. The TP system Π_{x1} generates $Perm(b^3a^2)$ using rules r_1 and r_4, a^3 using rules r_2 and r_4, and b^2 using rule r_3.

EXAMPLE 4. $\Pi_{x2} = (\{a,b,c\}, [_1]_1, ab^2, \{r_1 : b^2 \to (b, out), r_2 : a \to c(a^2, out), r_3 : c \to (a, out)\}, 0)$

TP-dep system Π_{x2} contains only one cooperative rule r_1. Rule r_1 is cat 2 coop-dep rule since it requires only one dependent trigger b. Note that in order to apply rule r_1, two copies of b must be present in region 1. The TP system Π_{x2} outputs to the environment the multiset ba^2 via rules r_1 and r_2 at the first transition and multiset a via rule r_3 in the next transition. Thus, the system generates $Perm(ba^2)a$ ($L(\Pi_{x2}) = \{ba^3, a^2ba, aba^2\}$).

EXAMPLE 5. $\Pi_{x12} = (\{a,b,c,d,e,f\}, [_1]_1, a^2bcef, \{r_1 : cb \to b(a, out), r_2 : a^2 \to d(b, out), r_3 : ab \to (c, out), r_4 : d \to (d, out), r_5 : b \to (e, out), r_6 : c \to (f, out), r_7 : ef \to (a, out), r_8 : e \to (c, out), \}, 0)$

TP-dep system Π_{x12} contains only four cooperative rules. Rule r_1 is a coop-ind rule; rules r_3 and r_7 are cat 1 coop-dep rules; rule r_2 is a cat 2 coop-dep rule. We leave it to the readers to verify that TP system Π_{x12} generates strings in $Perm(aba)Perm(de)$, $Perm(befa)d$, $Perm(abc)Perm(de)$, $Perm(cfa)$, $Perm(befc)d$ and $Perm(cfc)$.

3.1 From TP-dep to TP-ind system: An Initial Approach

We are interested in determining if the language generated by TP-dep systems can also be generated by TP-ind systems. Given a TP-dep system Π, an initial approach to accomplish this is to include a rule $o \to o$ in a region h ($1 \leq h \leq m$) for each object $o \in O$ that is a dependent trigger in the region. This rule can be used to consume dependent triggers that may remain unevolved. Since the added rule simply produces the same object as the consumed, it offers the same effect as carrying over a dependent trigger to the next configuration. The resulting TP-ind system when the initial approach is aplied to TP-dep system Π_{x1} in Example 3 is as follows: $\Pi'_{init1} = (\{a,b,c\}, [_1]_1, abc, \{r'_1 : ab \to (b^3, out), r'_2 : a \to (a, out), r'_3 : ac \to (b^2, out), r'_4 : c \to (a^2, out), r'_{dep1} : b \to b\}, 0)$.

In Π'_{init1}, the strings in $Perm(b^3a^2)$ can still be generated via rules r'_1 and r'_4. The weakness of this approach is when considering strings a^3 and b^2. The system Π'_{init1} fails to generate strings a^3 and b^2 because the halting configurations for producing these strings involve a copy of object b. This copy of b will then continuously evolve through the additional rule r'_{dep1} leading the system to a non-halting configuration.

The resulting TP-ind system by adding $o \to o$ for dependent triggers in TP-dep system Π_{x2} in Example 4 is

$\Pi'_{init2} = (\{a,b,c\}, [_1]_1, ab^2, \{r'_1 : b^2 \to (b, out), r'_2 : a \to c(a^2, out), r'_3 : c \to (a, out), r'_{dep1} : b \to b\}, 0)$.

In Π'_{init2}, all strings in $L(\Pi_{x2})$ are produced in Π'_{init2}. However, an additional string $aaab$ is also produced. This happens when rule r'_2 followed by rule r'_3 is applied first before applying rule r'_1. This means, during the application of rules r'_2 and r'_3, instead of using r'_1, both copies of b are consumed through the additional rule r'_{add1}.

The initial construction of a Π' from a given Π is ineffective due to issues with handling dependent triggers. In the next subsections, we try to improve this initial approach for TP-dep systems having cat 1 and/or cat 2 coop-dep rules.

3.2 TP-dep systems with Cat 1 coop-dep rules

Recall that if a rule $r : u \to v$ is a cat 1 coop-dep rule in a certain region, at least one object o in multiset u is used in a non-cooperative rule $r' : o \to v'$. This is in contrast with a TP-ind rule which requires that all objects in the LHS are independent triggers. As a consequence, in the presence of multiset u, two non-deterministic choice exists. Either a multiset u is consumed via rule r or copies of object o in u is consumed via rule r'. In the latter case, some dependent triggers in u will be carried over to the next configuration.

Referring to our initial approach to constructing a TP-ind system, for TP systems with cat 1 cooperative rule, we add two additional rules $r'_1 : o \to o$ and $r'_2 : o \to \bar{o}$ for every dependent trigger o in a region. Rule r'_1 is used to imitate the event (in the input TP system) of passing dependent triggers unevolved from current to next configuration whereas rule r'_2 is used to imitate the event where o is no longer used in the next transitions and therefore, will already be part of a halting configuration.

Formally, given a TP-dep system $\Pi_1 = (O, \mu, w_1, R_1, 0)$ where coop-dep rules are classified as cat 1, we construct a $\Pi'_1 = (O', \mu, w_1, R'_1, 0)$ where $O' = O \cup \{\bar{o} \mid o \in O,\ o$ is a dependent trigger in region 1. Every rule set $R'_1 = R_1 \cup \{o \to$

$o, o \to \bar{o} \mid o \in O$, object o is a dependent trigger in region 1. For easier reference to rules in R'_1, we let $R'_{dep} = \{r \in R'_1 \mid r : o \to o \notin R_1\}$, and $R'_{\overline{dep}} = \{r \in R'_1 \mid r : o \to \bar{o} \notin R_1\}$.

EXAMPLE 6. *For the TP-dep system Π_{x1} in Example 3, the resulting TP-ind system following the construction of Π'_1 is shown below:* $\Pi'_{x1} = (\{a, b, c, \bar{b}\}, [_1]_1, abc, \{r'_1 : ab \to (b^3, out), r'_2 : a \to (a, out), r'_3 : ac \to (b^2, out), r'_4 : c \to (a^2, out), r'_{dep1} : b \to b, r'_{\overline{dep1}} : b \to \bar{b}\}, 0)$.

LEMMA 2. $L(\Pi_1) \subseteq L(\Pi'_1)$.

PROOF. Suppose $s \in L(\Pi_1)$. Then, there exists a halting computation path $C_0 \Rightarrow^* C_{halt}$ that generates s. Our goal is to establish a halting computation $C'_0 \Rightarrow^* C'_{halt'}$ generating s in Π'_1.

By definition, C'_0 is the same as C_0. Let G_t be the set of rules used in a transition $C_t \Rightarrow C_{t+1}$ $(0 \leq t < halt)$. Also, let H_t be the set of unconsumed dependent triggers in $C_t \Rightarrow C_{t+1}$. We can derive a transition $C'_t \Rightarrow C'_{t+1}$ in Π'_1 (where C_t is the same as C'_t and C_{t+1} is the same as C'_{t+1}) by applying $(R'_1 \cap G_t) \cup \{o \to o \mid o \in H_t\}$. Each rule in $(R'_1 \cap G_t)$ is applied the same number of times as its equivalent rule in $C_t \Rightarrow C_{t+1}$. The number of times each rule in $o \to o \in R'_{dep}$ is applied is equal to the number of unconsumed copies of o from $C_t \Rightarrow C_{t+1}$.

Let K be the set of dependent triggers in configuration C_{halt}. If $K = \emptyset$, C'_{halt} is already a halting configuration (i.e. $halt = halt'$). Otherwise, $C'_{halt} \Rightarrow C'_{halt+1}$ by applying $o \to \bar{o} \in R'_{\overline{dep}}$ for all $o \in K$ (i.e. $halt' = halt + 1$). □

To illustrate Lemma 2, we simulate the production of some strings in $L(\Pi_{x1})$. Generating strings in $Perm(b^3a^2)$ are generated in Π'_{x1} through the use of rules r'_1 and r'_4 (the corresponding rules for r_1 and r_4 in Π_{x1}). Generating string a^3 are produced in Π'_{x1} via rules r'_2, r'_4 and r'_{dep1}. Afterward, rules $r'_{\overline{dep1}}$ can evolve b to $\bar{b}$ so that the next configuration becomes halting.

LEMMA 3. $L(\Pi_1) \supseteq L(\Pi'_1)$.

PROOF. A string $s \in L(\Pi'_1)$ is generated by a halting computation $C'_0 \Rightarrow^* C'_{halt}$. As in Lemma 2, C_0 is the same as C'_0. To show that there exists a computation $C_0 \Rightarrow^* C_{halt'}$ generating s in Π_1, we describe how a transition in Π'_1 corresponds to a valid transition in Π_1.

While any multiset involving non-triggers and independent triggers in Π_1 is uniquely represented in Π'_1, every dependent trigger o in Π_1 is represented by any of o or $\bar{o}$ in Π'_1; an occurrence of o in Π'_1 triggers rules in R'_{dep} and $R'_{\overline{dep}}$.

Let G'_t be the set of rules used in a transition $C'_t \Rightarrow C'_{t+1}$ in Π'_1. Transition $C'_t \Rightarrow C'_{t+1}$ $(1 \leq t \leq halt)$ is represented in Π_1 by a transition $C_t \Rightarrow C_{t+1}$ that uses the set $G'_t \cap R_1$. Each rule in $(R_1 \cap G_t)$ is applied the same number of times as its equivalent rule in $C'_t \Rightarrow C'_{t+1}$. Note that if configuration C'_t contains $\bar{o}$, the counterpart o in C_t cannot be evolved by any coop-dep rule r triggered by o in the transition $C_t \Rightarrow C_{t+1}$. However, even in this case, any transition $C'_t \Rightarrow C'_{t+1}$ is still mapped to a valid transition $C_t \Rightarrow C_{t+1}$. This is because if a coop-dep rule can be applied in C_t (and it cannot be applied in C'_t due to the occurrence of $\bar{o}$ instead of o), some non-cooperative rules will be used in $C_t \Rightarrow C_{t+1}$ since cat 1 coop-dep rules need independent triggers (as stated in Definition 5). In this case, the resulting halting computation in Π'_1 from transition $C'_t \Rightarrow C'_{t+1}$ will correspond to a computation in Π_1 with a dependent trigger o in its halting configuration.

From this, it can be concluded that there is a computation $C_0 \Rightarrow^* C_{halt'}$ in Π_1 which generates the same string as in a given computation $C'_0 \Rightarrow^* C'_{halt}$. The value $halt' \leq halt$ since a dependent trigger o can continuously evolve via rule $o \to o$ even when no other rules are applied in the system. Eventually however, o becomes $\bar{o}$ through $o \to \bar{o}$. □

To illustrate Lemma 3, we continue examining TP system Π'_{x1}. In Π'_{x1}, there are several ways to generate a^3. Note that in Π_{x1}, production of string a^3 leaves a copy of b in region 1 of the halting configuration of Π_{x1}. In Π'_{x1}, it can be produced by rules r'_2, r'_4 and $r'_{\overline{dep1}}$. Other ways to produce it includes initial use of rules r'_2, r'_4 and r'_{dep1}, afterwards, we continuously use r'_{dep1} for the single copy of b until use of rule $r'_{\overline{dep1}}$.

THEOREM 1. *For every TP-dep system $\Pi_1 = (O, [_1]_1, w_1, R_1, 0)$ where coop-dep rules are cat 1, there exists a TP-ind system Π'_1 such that $L(\Pi_1) = L(\Pi'_1)$.*

PROOF. The construction of Π'_1 from a given Π_1 is as given right before Lemma 2. The theorem follows from Lemmas 2 and 3. □

COROLLARY 1. $LTP(dep, C_1) \subseteq LTP(ind)$.

PROOF. Given a TP-dep system Π where coop-dep rules are cat 1, based on Lemma 1, we can construct a single membrane TP system Π_2 where the original and the constructed TP system has the same language. In the resulting TP system, the role of an object in triggering rules is preserved. Thus, the coop-dep rules in Π_2 remain cat 1. We can now use Theorem 1 to show that we can construct a Π'_2 from Π_2 such that $L(\Pi_2) = L(\Pi'_2)$. □

3.3 TP-dep systems with Cat 2 coop-dep rules

To provide us a hint on the approach used for this type of TP-dep system, we first examine an example of cat 2 coop-dep rule. Suppose a region has a rule $r : a^4 \to v$. If the multiset in the given region is one of: a, a^2 and a^3, then no dependent triggers will be consumed in this region. Now suppose the region has a multiset a^{10}. Due to maximal parallelism, only eight copies of a can be consumed and only two copies of object a will be unconsumed. If we add some more rules in the region, say a rule $r' : a^3 \to v'$, then the number of possible multiset that leaves a unconsumed will be decreased. The only multiset of object a carried over from

a current to next configuration will be multisets a and a^2. In the presence of an added rule $r'' : b^2 \rightarrow v''$, the number of multisets involving unconsumed dependent triggers will be increased. These are a, a^2, a^3, b, ab, a^2b and a^3b.

From the analysis provided above, the major difference between cat 2 and cat 1 coop-dep rules will be the number of multisets of possible unconsumed dependent triggers in a given region. While in the latter, this number can be unbounded (e.g. there can be unbounded copies of b of unconsumed in a region with a rules $ab \rightarrow v$ and $a \rightarrow v'$), in the former, this number is only a finite number. We capitalize on this notion of finiteness to construct a TP-ind system for a TP-dep system with cat 2 coop-dep rules. At this point, we formally define a set of multisets fin containing multisets that can possibly be left unconsumed due to cat 2 coop-dep rules.

We use the notation Π_2 for a TP-dep system $\Pi_2 = (O, [_1]_1, w_1, R_1, 0)$ where all coop-dep rules are classified as cat 2.

DEFINITION 6. **(Finite Set fin)** *Let D be the set of all dependent triggers in region 1 of Π_2. For each subset $Y = \{y_1, y_2, \ldots, y_{|Y|}\} \subseteq D$, $Y \neq \emptyset$,*

$$p_1 p_2 \ldots p_{|Y|} \in fin$$

where each $p_j \in \{y_j^f \mid \text{for all } f\text{: } 1 \leq f \leq q_j - 1\}$ for all f : $1 \leq j \leq |Y|$. We obtain q_j by finding a rule $r : y_j^{q_j} \rightarrow v$ and rule r uses the least number of copies of y_j in its LHS.

From Definition 6, in the construction of fin, the set of dependent triggers in region 1 is first identified as D. Afterwards, all possible subsets Y of D (except for the empty set) is determined. A multiset in fin is defined by examining each Y. For each element y_j $(1 \leq j \leq |Y|)$ of Y, we can associate a rule $r : y_j^{q_j} \rightarrow v$ that requires the least number of y_j. For each subset Y, a multiset in fin is represented as a sequence $p_1 p_2 \ldots p_{|Y|}$ where every p_j is an element in $\{y_j^f \mid \text{ for all f: } 1 \leq f \leq q_j - 1\}$.

EXAMPLE 7. *Suppose a TP-dep system following Π_2 has $R_1 = \{b^3 \rightarrow b(b, out), b^4 \rightarrow b^2(b^2, out),\ c^2 \rightarrow c\}$. The set of dependent triggers in region 1 will be $D = \{b, c\}$. The rule that requires the least number of c is $c^2 \rightarrow c$ whereas rule $b^3 \rightarrow b(b, out)$ requires the least number of b. The set fin is thus achieved by combining elements in the set $B = \{b, b^2\}$ and $C = \{c\}$. Since Y is a subset of D, $Y \in \{\{b\}, \{c\}, \{b, c\}\}$. For $Y = \{b\}$, multisets b and b^2 is added to fin. For $Y = \{c\}$, multiset c is added to fin. For $Y = \{b, c\}$, multisets bc and b^2c is added to fin. Specifically, the set $fin = \{b, b^2, c, bc, b^2c\}$.*

LEMMA 4. *In a TP-dep system Π_2, any possible multiset of dependent triggers that will remain unconsumed in any transition of Π_2 is in fin.*

PROOF. In Π_2, all coop-dep rules are cat 2. Thus, for every dependent trigger y, any rule having y in its LHS has the form $r : y^k \rightarrow v$. From Definition 6, all multiset w in fin has $|w|_y > q$ (q is the number of copies of y for the rule that least requires q). Thus, no rule r triggered by y is enabled by multiset w.

To show that every multiset unconsumed is in fin, we first determine what needs to be considered in finding all desired multisets: any such multiset is obtained by (1) combining possible unconsumed multiset (2) per dependent trigger and (3) considering that in some configurations, some dependent triggers may not appear. From Definition 6, the range of f for every dependent trigger involved in fin ensures (1), construction of D ensures (2) while considering all possible subsets Y of D (except $Y = \emptyset$) ensures (3). □

We now construct a TP-ind system Π_2' from Π_2. We shall use fin (and the notations in Definition 6) in constructing the rules of the system. Given TP-dep system $\Pi_2 = (O, \mu, w_1, R_1, 0)$ where all coop-dep rules are cat 2, we now construct a TP-ind system $\Pi_2' = (O', \mu, w_1', R_1', 0)$ where $O' = O \cup \{\bar{y} \mid y \in D\} \cup \{\alpha, \#\}$. The initial multiset w_1' includes multiset w_1 and a copy of an additional object α. The set $R_1' = R_1 \cup \{\alpha \rightarrow \alpha, \alpha \rightarrow \lambda, \# \rightarrow \#\} \cup \{y \rightarrow \# \mid y \in D\} \cup \{\alpha p \rightarrow \alpha p, \alpha p \rightarrow \bar{p} \mid p \text{ is in } fin(h) \text{ and for each object } y \text{ in } p,\ \bar{y} \text{ is in } \bar{p}\}$.

Notice that in Section 3.2, rules of the form $o \rightarrow o$ and $o \rightarrow \bar{o}$ are added in the constructed TP-ind system to imitate the event (in the original model) where dependent triggers can still be consumed in future transitions, and the event where these triggers will no longer be consumed, respectively. In the latter, these triggers will be part of the halting configuration of a certain computation. Our approach in finding a desired TP-ind system for TP systems with cat 2 coop-dep rule also employs such technique. However, aside from such restriction, there is also a need to monitor the possible multiset of dependent triggers that will be carried over in a transition. Lemma 4 shows that the number of such possible multisets is finite and can be enumerated. Thus, rules of the form $\alpha p \rightarrow \alpha p$ and $\alpha p \rightarrow \bar{p}$ is created. If the multiset of unconsumed (or carry over) dependent triggers in a transition is not in fin, then some dependent triggers will evolve to the trap symbol $\#$ and thus, induces a non-halting computation. The role of α is further explained in Lemmas 5 and 6.

For easier reference to rules in R_1', we let $R_{fin}' = \{r \in R_1' \mid r : \alpha p \rightarrow \alpha p \notin R_1,\ p \text{ is in } fin\}$, $R_{\overline{fin}}' = \{r \in R_1' \mid r : \alpha p \rightarrow \bar{p} \notin R_1,\ p \text{ is in } fin \text{ and for each } y \text{ in } p,\ \bar{y} \text{ is in } \bar{p}\}$, $R_{trap}' = \{y \rightarrow \# \mid y \in D\}$ and $R_{add}' = \{\alpha \rightarrow \alpha, \alpha \rightarrow \lambda, \# \rightarrow \#\}$.

EXAMPLE 8. *For the TP-dep system Π_{x2} in Example 4, the resulting TP-ind system following the construction of Π_2' is shown below: $\Pi_{x2'} = (\{a, b, c, \bar{b}, \alpha, \#\}, [_1]_1, \alpha ab^2, \{r_1' : b^2 \rightarrow (b, out), r_2' : a \rightarrow c(a^2, out), r_3' : c \rightarrow (a, out), r_{fin1}' : \alpha b \rightarrow \alpha b, r_{\overline{fin1}}' : \alpha b \rightarrow \bar{b}, r_{trap1}' : b \rightarrow \#, r_{add1}' : \alpha \rightarrow \alpha, r_{add2}' : \alpha \rightarrow \lambda, r_{add3}' : \# \rightarrow \#, \}, 0)$*

LEMMA 5. $L(\Pi_2) \subseteq L(\Pi_2')$.

PROOF. Suppose $s \in L(\Pi_2)$ is generated through a halting computation path $C_0 \Rightarrow^* C_{halt}$. Our goal is to show

that there is a halting computation $C'_0 \Rightarrow^* C'_{halt'}$ in Π'_{x2} that generates s.

Let the set G_t (G'_t) be the set of rules used at a transition $C_t \Rightarrow C_{t+1}$ ($C'_t \Rightarrow C'_{t+1}$), $t \geq 0$. From their definition, C'_0 contains the same multiset as in C_0 with an additional copy of an object α. The set G'_t contains rules in $(R'_1 \cap G_t)$. Each rule in $(R'_1 \cap G_t)$ is applied the same number of times as its equivalent rule in $C_t \Rightarrow C_{t+1}$. Application of other rules in R'_1 is dependent whether: (i) some dependent triggers are carried over from C_t to C_{t+1}, or (ii) all triggers present in region h at C_t are consumed in the transition.

For case (i), suppose the multiset p is the multiset of dependent triggers left unconsumed in $C'_t \Rightarrow C'_{t+1}$. Then rule $\alpha p \rightarrow \alpha r \in R'_{fin}$ is also added in G'_t. The existence of $\alpha p \rightarrow \alpha p$ is guaranteed because of Lemma 4. Through rule $\alpha p \rightarrow \alpha p$, no more dependent triggers are left unconsumed in region 1. Therefore, $C'_t \Rightarrow C'_{t+1}$ is applied in a maximally parallel manner.

For case (ii), when all triggers are consumed, the α in C_t is evolved through rule $\alpha \rightarrow \alpha \in R'_{add}$. This move allows the possibility that some multiset of dependent triggers remain in consideration for evolution in future transitions.

For the halting computation C_{halt}, if no dependent trigger occurs, the last transition in Π'_2 applies rule $\alpha \rightarrow \lambda \in R'_{add}$ (i.e. $halt' = halt$). Otherwise, a rule $\alpha p \rightarrow \bar{p} \in R'_{\overline{fin}}$ is applied in the next step; thus, halting configuration is achieved in Π'_2 in $halt' = halt + 1$. □

To illustrate Lemma 5, we simulate the production of some strings in $L(\Pi_{x2})$. Generating strings in $Perm(ba^2)a$ are generated in Π'_{x2} through in the initial transition, using rules r'_1, r'_2 (the corresponding rules for r_1 and r_2 in Π_{x2}) and rule r'_{add1} outputing multiset ba^2 in the environment. In the next transition, rule r'_3 (the corresponding rule for r_3 in Π_{x2}) and rule r'_{add1} is used outputing multiset a in the environment. The last transition uses rule r'_{add2} to remove α and halt.

LEMMA 6. $L(\Pi_2) \supseteq L(\Pi'_2)$.

PROOF. Let $s \in L(\Pi'_2)$ be generated via a halting computation, $C'_0 \Rightarrow^* C'_{halt}$. We need to show that s can also be generated via a halting computation in $C_0 \Rightarrow C'_{halt'}$ in Π_2.

Based on the construction of Π'_2, C_0 is of the same multiset as in C'_0 without the copy of α. Let G'_t be the set of rules in region 1 used in configuration C'_t ($1 \leq t \leq halt - 1$). The rules used to consume object α in region 1 is one of the following: (a) $\alpha \rightarrow \alpha$ (b) $\alpha p \rightarrow \alpha p$ where $p \in fin$, (c) $\alpha \rightarrow \lambda$, (d) $\alpha p \rightarrow \bar{p}$ where $p \in fin$, $\bar{y} \in \bar{p}$ if $y \in p$. In the first two items, α is retained in C'_1; in the last two items, α is removed.

Suppose one of rule (a) and (c) is used in G'_0. The only other possible rules that will consume the remaining multiset in region 1 of C'_0 will be rules in R_1. (Although it is also possible to use a rule of the form $y \rightarrow \# \in R'_{trap}$ for some dependent trigger y, it will result to a non-halting computation due to rule $\# \rightarrow \# \in R'_{add}$.) This means, $C_0 \Rightarrow C_1$ uses rules in $G'_0 \cap R_1$ wherein the number of application of rules in G_0 is the same as the number of applications of their equivalent rule in G'_0. Note that upon use of (a) or (c), no dependent trigger is left unconsumed in transition $C'_0 \Rightarrow C'_1$, thus, the rules used for transition $C_0 \Rightarrow C_1$ is maximally parallel.

In the case where one of rules (b) and (d) is used in G'_0, again, the only other possible rules that can be applied to the remaining multiset will be rules in R_1. If we let G_0 be rules in G'_0 without the rule $\alpha p \rightarrow \alpha p$ (when (b) is used) or $\alpha p \rightarrow \bar{p}$ (when (d) is used), then applying G_0 for transition $C_0 \Rightarrow C_1$ means dependent triggers in multiset p are unconsumed. However, since Lemma 4 established that any multiset $p \in fin$ doesn't enable any rule in R_1, the rules used in region 1 for the transition $C_0 \Rightarrow C_1$ is also maximally parallel. Observe that in applying rules (a),(b) or (c), as in the relation of C_0 and C'_0, multiset per region in C_1 is just the same as C'_1 without object α. Additionally, in applying rule (d), objects $\bar{o}$ in C'_1 becomes object o in C_1.

Using the reasoning above, it can be observed that so long as region 1 initially containing an object α retains α in configuration C'_t, then for every transition $C'_t \Rightarrow C'_{t+1}$, there is a transition $C_t \Rightarrow C_{t+1}$ that satisfies maximal parallelism. We now look at the event wherein a configuration C_t doesn't contain an α. This means, in previous transitions, a rule has already been used to remove α. In such case, G'_t is only composed of rules in R_1. From here, transition $C_t \Rightarrow C_{t+1}$ can be derived by using $G_t = G'_t$. If no object $\bar{y}$ is present in C'_t, then this means no dependent triggers are left unconsumed in $C_t \Rightarrow C_{t+1}$, thus making rules in G_t maximally parallel with respect to region 1's multiset in C_t. If some multiset of dependent triggers $\bar{y}$ is present, then this means, some dependent triggers are left unconsumed in $C_t \Rightarrow C_{t+1}$, however, since rule $\alpha p \rightarrow \bar{p}$ can only be applied once in a region 1, the only possible multiset of dependent triggers in $\bar{y}$ in region 1 is $p \in fin$. Thus, no rule can be applied to the unconsumed multiset of dependent triggers making rules G_t maximally parallel when applied to transition $C_t \Rightarrow C_{t+1}$.

The analysis above shows that given a halting computation $C'_0 \Rightarrow^* C'_{halt}$, there is a halting computation $C_0 \Rightarrow^* C_{halt'}$ that satisfies maximal parallelism and with applied rules having the same collective effect as the former. Thus, if string s is generated in the former, then it can also be generated in the latter. The value of $halt' = halt$ if at least one region retains object α until C'_{halt-1} and only eliminates it at transition $C'_{halt-1} \Rightarrow C_{halt}$. Otherwise, $halt' < halt$. □

To illustrate Lemma 6, we continue examining TP system Π'_{x2}. In Π'_{x2}, there are several ways to generate $Perm(ab^2)a$. However, all these ways involve using rule r'_1 and r'_2 in the initial configuration and rule r'_3 in the next configuration. In generating $Perm(ab^2)a$, handling the copy of α involves zero or more applications of rule r'_{add1} until use of rule r'_{add2}. The copies of b and α also enables rules r'_{fin1}, $r'_{\overline{fin1}}$ and r'_{trap1}. Upon use of any of these rules, rule r'_{add3} is applied leading to a non-halting computation. Note that rules r'_{fin1} and $r'_{\overline{fin1}}$ lead to the use of r'_{trap1} since both rules need only one b to accompany α and there are two copies of b in the initial configuration.

THEOREM 2. *For every TP-dep system* $\Pi_2 = (O, [_1]_1, w_1, R_1, 0)$ *where coop-dep rules are cat 2, there exists a TP-ind system* Π'_2 *such that* $L(\Pi_2) = L(\Pi'_2)$.

PROOF. The construction of Π'_2 from a given Π_2 is as given right before Lemma 5. The theorem follows from Lemma 5 and Lemma 6. □

As in Corollary 1, we have the following corollary.

COROLLARY 2. $LTP(dep, C_2) \subseteq LTP(ind)$

PROOF. This follows from Lemma 1 and Theorem 2. □

3.4 TP-dep systems with Cat 1 and Cat 2 coop-dep rules

From the previous subsections, TP-dep systems with cat 1 and cat 2 coop-dep rules can both occur in a given TP system. To construct a TP-ind system for such model, there is a need to handle dependent triggers that are both in the LHS of a cat 1 and cat 2 cooperative rule. We first examine an instance. Suppose rules $r_1 : ab \to v$, $r_2 : a \to v'$ is defined in region 1. Given these rules, so long as no copy of a is used to consume a b, there can be unbounded copies of trigger b unconsumed in region h. However, if an additional rule, $r_3 : b^2 \to v''$ occurs in region 1, then, the possible multiset that can be consumed will be limited to multiset b. Thus, handling dependent trigger b as a trigger for a cat 2 coop-dep rule prevails over handling it as a trigger for a cat 1 coop-dep rule.

Given a TP-dep system $\Pi_{12} = (O, \mu, w_1, R_1, 0)$ where all coop-dep rules are either cat 1 or cat 2, a TP-ind system Π'_{121} can be constructed by first constructing a TP-dep system Π_{121} achieved by removing all cat 2 coop-dep rule. Let D_{12} be the set of all dependent triggers in region 1 involved in a cat 2 coop-dep rule. Given TP-dep system Π_{121} where coop-dep rule are cat 1, a TP-ind system $\Pi_{121}{}'$ can be constructed following Section 3.2. From Π'_{121}, a TP-dep system with cat 2 coop-dep rule Π_{122} can be constructed by putting back all cat 2 coop-dep rule removed previously and removing all additional rules $o \to o$ and $o \to \bar{o}$ where $o \in D_{12}$. Following Section 3.3, a TP-ind system Π'_{12} can be constructed from Π_{122}.

EXAMPLE 9. *For the TP-dep system* Π_{x12} *in Example 5, it can be observed that objects* $b, d, c,$ *and* e *are independent triggers, object* f *is a cat 1 dependent trigger, and object* a *both a cat 1 and cat* 2 *dependent trigger. The resulting TP-ind system following the construction of* Π'_{12} *is shown below:* $\Pi'_{x12} = (\{a, b, c, d, e, f, \bar{a}, \bar{f}, \alpha, \#\}, [_1]_1, \alpha a^2 bcef, \{r'_1 : cb \to b(a, out), r'_2 : a^2 \to d(b, out), r'_3 : ab \to (c, out), r'_4 : d \to (d, out), r'_5 : b \to (e, out), r'_6 : c \to (f, out), r'_7 : ef \to (a, out), r'_8 : e \to (c, out), r'_{dep1} : f \to f, r'_{\overline{dep1}} : f \to \bar{f}, r'_{fin1} : \alpha a \to \alpha a, r'_{\overline{fin1}} : \alpha a \to \bar{a}, r_{trap1} : b \to \#, r'_{add1} : \alpha \to \alpha, r'_{add2} : \alpha \to \lambda, r'_{add3} : \# \to \#\}, 0)$

THEOREM 3. *For every TP-dep system* $\Pi_{12} = (O, \mu, w_1, R_1, 0)$ *where all coop-dep rules are either cat 1 or cat 2, there exists a TP-ind system* Π'_{12} *such that* $L(\Pi_{12}) = L(\Pi'_{12})$.

PROOF. The construction of Π'_{12} is as given right before Theorem 3. If every dependent trigger in Π_{12} can be associated with only one of the two cases: cat 1 or cat 2 coop-dep rule, then no object that is in the LHS of both cat 1 and cat 2 coop-dep rule exists in Π. Every dependent trigger can then be handled depending on whether it is a trigger for a cat 1 coop-dep rule, in which case, it will be handled using rules in Section 3.2, or cat 2, thus it will be handled using rules in Section 3.3. This implies that discussion in Section 3.2 and Section 3.3 can be employed to show that every set of rules G_t applied in a transition $C_t \Rightarrow C_{t+1}$ has a corresponding set of rules G'_t in transition $C'_t \Rightarrow C'_{t+1}$ that has the same effect as in the former and satisfies maximal parallelism. The other direction also holds. Hence, for each computation $C_0 \Rightarrow^* C_{halt}$ that generates $s \in L(\Pi_{12})$, there is a computation $C'_0 \Rightarrow^* C'_{halt'}$ that also generates s and vice versa.

In the case where a dependent object o functions as a dependent trigger for cat 1 and cat 2 coop-dep rule, such object will be handled as a trigger for cat 2 rule. Notice that if object o is treated as a cat 1 dependent trigger, then available rules handling unconsumed object o will be of the form $o \to o$ and $o \to \bar{o}$. The former rule implies that some objects can continue being unconsumed until it can be consumed by some rule in future transitions. On the contrary, the latter implies an object o will no longer be used in any rule at any future transition and will certainly be part of the halting configuration (since $\bar{o}$ are non-triggers in Π'). If object o is treated as a cat 2 dependent trigger, then it is part of any multiset p such that rules $\alpha p \to \alpha p$ and $\alpha p \to \bar{p}$ are applicable. The multiset p guarantees that the number of object o unconsumed in region 1 doesn't exceed that which can trigger a cat 2 coop-dep rule. However, the effect of rules $o \to o$ and $o \to \bar{o}$ for object o are preserved in rules $\alpha p \to \alpha p$ and $\alpha p \to \bar{p}$ since object o is in p. □

COROLLARY 3. $LTP(dep, C_{12}) \subseteq LTP(ind)$

PROOF. This follows from Lemma 1 and Theorem 3. □

4. CONCLUSIONS

In this work we established the constructions necessary for certain TP systems to produce the same language. The constructions are based on the type of rules that exist in the TP systems: a TP-ind system Π' is constructed with respect to a TP system with some dependent triggers Π, where their generated language are equal. The rules of Π can be of (or a combination of) 3 categories from section 3: case 1 to case 3. This work only considers combinations of case 1 and case 2. Considering case 3 for TP systems, as well as for other cell-like P systems with similar multi-set rewriting rules, is of theoretical as well as practical interest. We end the paper by recalling that considering case 3 implies answering the question: can a Π' be constructed from a Π with case 3 cooperative rules where $L(\Pi') = L(\Pi)$?

Acknowledgements

Juayong and Cabarle are supported by DOST-ERDT scholarships. Buño is supported by the DMCI teaching research grant. Adorna is funded by a DOST-ERDT grant and the Semirara Mining Corporation Professorial Chair.

5. REFERENCES

[1] Alhazov, A., Ciubotaru, C., Ivanov, S., Rogozhin, Y.: The Family of Languages Generated by Non-cooperative Membrane Systems. LNCS 6501, pp. 65-80 (2011)

[2] Adorna, H., Păun, G., Pérez-Jiménez, M.J.: On Communication Complexity in Evolution-Communication P systems. ROMJIST vol 13(2) pp. 113-130 (2010)

[3] Ciobanu, G., Resios, A.: Computational Complexity of Simple P systems. Fundamenta Informaticae vol 87(1) (2008)

[4] Freund, R., Verlan, S.: A Formal Framework for Static (Tissue) P systems. In: Eleftherakis, G., Kefalas, P., Păun, G., Rozenberg, G., Salomaa, A. (eds.) WMC 2007. LNCS, vol. 4860, pp. 271–284. Springer, Heidelberg (2007)

[5] Hopcroft, J.E., Ullman, J.D.: Introduction to Automata Theory, Languages, and Computation. Addison-Wesley (1979)

[6] Juayong, R., Adorna, H.: On the Language Relation of TP systems and ECPe Systems [Accepted] 9th BIC-TA, Wuhan, China (2014)

[7] Păun, G.: Computing with membranes. J. Computer and System Sciences vol 61, pp. 108-143 (2000)

[8] Păun, G.: Membrane Computing. Springer-Verlag Berlin Heidelberg (2002)

[9] Păun, G.: Introduction to Membrane Computing. Ciobanu, Pérez-Jiménez, Păun (eds): Applications of Membrane Computing. Springer, pp.1-42 (2006)

[10] Qi, Z., You, J., Mao, H.: P systems and Petri Nets. In: Martín-Vide, C. et. al (eds.) WMC 2003. LNCS, vol. 2933, pp. 286–303. Springer, Heidelberg (2004)