Computing by Carving with P Systems. A First Approach[⋆]

José M. Sempere

Departamento de Sistemas Informáticos y Computación
Universidad Politécnica de Valencia
Valencia, Spain
E-mail: `jsempere@dsic.upv.es`

Summary. In this work, we propose a P system which carries out computing by carving. Computing by carving was proposed by Gh. Păun as a technique to generate formal languages which can even be non recursively enumerable. Hence, it can be considered a hypercomputational technique. Here, we propose a first scheme based on P systems in order to perform computing by carving any formal language. So, the paper shows indirectly that these systems, under certain assumptions, can be considered a model for hypercomputation.

1 Introduction

Computing by carving is a computational strategy to generate formal languages proposed by Gh. Păun in 1999 [4]. This technique has been proved to generate even non recursively enumerable languages, hence it can be considered a hypercomputational technique. Furthermore, in the same work, Păun proved that it can be a used as a solution to language approximation problems. Here, we will propose a membrane system architecture to perform computing by carving. Hence, we indirectly prove that membrane systems can be considered hypercomputational models.

Hypercomputational models have been proposed along the time that solve some problems proved to be unsolvable by classical Turing machines. Most of these models need some kind of infinite resources (infinite tape alphabets, sets of states, etc.) as pointed out in [10]. Here, we will introduce infinite membrane regions and objects as a source for hypercomputing with P systems.

The structure of this work is as follows: In the following section, we will introduce basic concepts about formal language theory and membrane computing. Then, we will introduce the basic aspects and results about computing by carving. In section 4, we will propose a P system to perform computing by carving.

⋆ Work supported by the Spanish Ministerio de Educación y Ciencia under project TIN2007-60769

We will discuss different approximations to make so by using membrane creation and other ways to obtain potentially infinite resources. Finally, we will draw some conclusions and we will point out new problems for future research.

2 Basic concepts on formal language theory and membrane computing

We will introduce some basic concepts from formal language theory according to [2, 7] and from membrane computing according to [5].

An alphabet Σ is a finite nonempty set of elements named symbols. A string defined over Σ is a finite ordered sequence of symbols from Σ. The infinite set of all the strings defined over Σ will be denoted by Σ^*. Given a string $x \in \Sigma^*$ we will denote its length by $|x|$. The empty string will be denoted by λ and Σ^+ will denote $\Sigma^* - \{\lambda\}$. A language L defined over Σ is a set of strings from Σ. The difference between two languages L_1 and L_2 is defined by $L_1 - L_2 = \{x \in L_1 : x \notin L_2\}$.

A *generalized sequential machine* can be defined by the tuple $T = (\Sigma, \Delta, Q, R, q_0, F)$, where Σ and Δ are aphabets, Q is a finite states set, $R \subseteq Q \times \Sigma^* \times \Delta^* \times Q$ is a finite transition relation, $q_0 \in Q$ is the initial state, and $F \subseteq Q$ is a set of final (or acceptance) states. The machine takes an initial string $x \in \Sigma^*$ and, after applying the transitions defined by T, it obtains an output string $y \in \Delta^*$ whenever it finishes in a final state. Then, we will say that T performs a function g such that $g(x) = y$.

A general P system of degree m is a construct

$$\Pi = (V, T, C, \mu, w_1, \cdots, w_m, (R_1, \rho_1), \cdots, (R_m, \rho_m), i_0),$$

where:

- V is an alphabet (the *objects*)
- $T \subseteq V$ (the *output alphabet*)
- $C \subseteq V$, $C \cap T = \emptyset$ (the *catalysts*)
- μ is a membrane structure consisting of m membranes
- w_i, $1 \leq i \leq m$, is a string representing a multiset over V associated with the region i
- R_i, $1 \leq i \leq m$, is a finite set of *evolution rules* over V associated with the ith region and ρ_i is a partial order relation over R_i specifying a *priority.*
  An evolution rule is a pair (u, v) (or $u \rightarrow v$) where u is a string over V and $v = v'$ or $v = v'\delta$ where v' is a string over
  $$\{a_{here}, a_{out}, a_{in_j} \mid a \in V, 1 \leq j \leq m\}$$
  and δ is an special symbol not in V (it defines the *membrane dissolving action*). From now on, we will denote the set *tar* by $\{here, out, in_k \mid 1 \leq k \leq m\}$.
- i_0 is a number between 1 and m and it specifies the *output* membrane of Π (in the case that it equals ∞ the output is read outside the system).

The language generated by Π in external mode ($i_0 = \infty$) is denoted by $L(\Pi)$ and it is defined as the set of strings that can be defined by collecting the objects that leave the system arranged in the leaving order (if several objects leave the system at the same time, then all permutations are allowed). The set of numbers that represent the objects in the output membrane i_0 will be denoted by $N(\Pi)$. Obviously, both sets $L(\Pi)$ and $N(\Pi)$ are defined only for *halting computations*.

One of the multiple variations of P systems is related to the modification of membrane structures. There have been several works in which these variants have been proposed (see, for example, [1, 3, 6]).

In the following, we enumerate some kinds of rules which are able to modify the membrane structure:

1. 2-division: $[_h a]_h \rightarrow [_{h'} b]_{h'} [_{h''} c]_{h''}$
2. Creation: $a \rightarrow [_h b]_h$
3. Dissolving: $[_h a]_h \rightarrow b$

The power of P systems with the previous operations and other ones (e.g., *exocytosis, endocytosis*, etc.) has been widely studied in the previously mentioned works and other papers.

3 Computing by carving

Păun introduced in [4] *computing by carving* as a technique to compute formal languages inspired by the search of solutions by filtering conditions in DNA computing. The main ingredients of computing by carving are the following:

1. A target language L
2. An initial couple of languages L_0 and L_1
3. An initial language M
4. A generalized sequential machine (gsm) g

The way of obtaining the target language L can be explained as follows: First, select a broader (regular) language M and an initial couple of (regular) languages L_0 and L_1 and calculate $L_{i+1} = g(L_i)$ for $i \geq 1$. The ith iteration of g over L_1 can be denoted by $g^i(L_1)$ and $g^*(L_1)$ will denote the language $\bigcup_{i\geq 0} g^i(L_1)$. Then L can be calculated as $L = M - (L_0 \cup g^*(L_1)$. If the latter condition holds, then L is said C-REG computable. Then, the triple (L_0, L_1, g) identifies the sequence that allows the calculation of L. Observe that M can be assumed to be Σ^*.

The following results can be found in [4]

Theorem 1. *Every recursively enumerable language $L \subseteq T^*$ can be written in the form $L = g^*(\{a_0\}) \cap T^*)$ where g is a gsm, depending on L, and a_0 is a fixed symbol not in T.*

Theorem 2. *There are C-REG computable languages which are not recursively enumerable.*

4 A P system for computing by carving

In this section, we propose a first approach to implement the components M, L_0, L_1 and g in order to compute a language L by carving. In general, we will work with sets of integers instead of languages of strings given that the connection of languages to sets of integers are very common in P systems.

The general scheme that we will initially follow in the proposed P system is shown in Fig. 1. First, the projections of the languages M, L_0 and L_1 can be generated by using P systems Π_M, Π_{L_0} and Π_{L_1}. Observe that any recursively enumerable language can be generated by a P system so they can be generated too. Moreover, we can take L_1 to be finite, even a *singleton* as shown in [4], so we can define the set of initial objects in Π_{L_1} to be exactly L_1. With respect to L_0 we can fix it to be Σ^* or the empty set. In both cases, it can be trivially generated even in a lexicographic order.

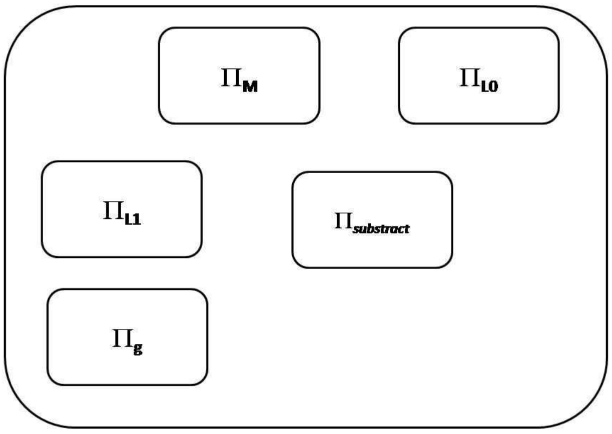

Fig. 1. Initial scheme for the P system

Once, we have generated a string in Π_M, Π_{L_0} or Π_{L_1} it is sent to $\Pi_{substract}$. Observe that, if we generate L_0 and L_1 in lexicographic order then we can make the difference in $\Pi_{substract}$ in lexicographic order too. The generalized sequential machine g can be applied by using a P system Π_g that receives objects from $\Pi_{substract}$. Once the generalized sequential machine has been applied in Π_g two different regions are created by using membrane division and membrane creation: Π_{g_2} and Π_{L_2}. Now, the new regions are used to calculate the second g iteration over L_1 in Π_{g_2} and the language L_2 in Π_{L_2}. The second substraction over the set M is performed again in $\Pi_{substract}$.

This scheme can be generalized to obtain L_j. Observe that the P system structure in this case is shown in Figure 2.

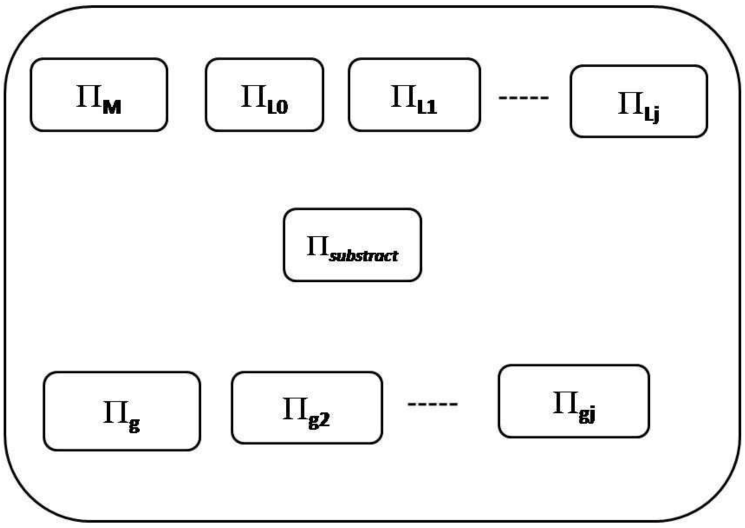

Fig. 2. Calculating L_j through jth iteration for the generalized sequential machine g

Some remarks about the proposed architecture:

1. The region corresponding to $\Pi_{substract}$ always contains the updated approximation to L. That is, when the computation is in progress, some strings will be sent from Π_M to $\Pi_{substract}$ and they will be deleted if the same string appears in any Π_{L_i}.
2. The whole process can be considered as an infinite time one, given that the generation of every L_i is infinite and so is for the updated approximation to L.
3. Here, the resources needed to compute L are infinite. Think about the number of regions and objects in every region. They will be unboundedly increased over the time.

5 Conclusions and future work

In this paper we have made a first approach to computing by carving with P systems. Here, we have only sketched the general ideas behind the full definition. So, we need to define the P systems carrying out the following tasks:

- The generation of any infinite (regular) language in lexicographic order.
- The application of the generalized sequential machine over strings and languages.
- The substraction between languages.

Every task referred before will be investigated in future works.

References

1. A. Alhazov, T.O. Ishdorj. *Membrane operations in P systems with active membranes.* In Proc. Second Brainstorming Week on Membrane Computing. TR 01/04 of RGNC. Sevilla University. pp 37-44. 2004.
2. J. Hopcroft, J. Ullman. Introduction to Automata Theory, Languages and Computation. Addison Wesley Publishing Co.,1979.
3. A. Păun. *On P systems with active membranes.* In Proc. of the First Conference on Unconventionals Models of Computation (UMC2K). pp 187-201. 2000.
4. Gh. Păun. (DNA) computing by carving. *Soft Computing* 3, pp 30-36. 1999
5. G. Păun. *Membrane Computing. An Introduction.* Springer. 2002.
6. G. Păun, Y. Suzuki, H. Tanaka, T. Yokomori. *On the power of membrane division on P systems.* Proc. Conf. on Words, Languages and Combinatorics (Kyoto). 2000.
7. G. Rozenberg, A. Salomaa (Eds.). Handbook of Formal Languages Vol. 1. Springer. 1997.
8. J.M. Sempere. P systems with external input and learning strategies. In *Preproceedings of the Workshop on Membrane Computing* (July 2003). A. Alhazov, C. Martín-Vide and Gh. Păun (editors), pp 445-448. 2003
9. J.M. Sempere. Covering reductions and degrees in P systems. In *Pre-proceedings of the Fifth Workshop on Membrane Computing* (June, 2004), pp 384-391. 2004
10. M. Stannet. Hypercomputational Models. In *Alan Turing: Life and Legacy of a Great Thinker.* C. Teuscher (*editor*). Springer. 2004