Conformon-P Systems with Negative Values

Pierluigi Frisco

School of Mathematical and Computer Sciences
Heriot-Watt University
Edinburgh, EH14 4AS, UK
pier@macs.hw.ac.uk

Summary. Some initial results on the study of conformon-P systems with negative values are reported.

One model of these conformon-P systems is proved to be computationally universal while another is proved to be at least as powerful as partially blind program machines.

1 Introduction

The subdivision of a cell into compartments delimited by membranes inspired G. Păun to define a new class of (distributed and parallel) models of computation called *membrane systems* [8]. The hierarchical structure, the locality of interactions, the inherent parallelism, and also the capacity (in less basic models) for membrane division, represent the distinguishing hallmarks of membrane systems. Research on membrane systems, also called 'P systems' (where 'P' stays for 'Păun'), has really flourished [9].

One of the lines of research within membrane systems deals with the study of the generative power of models of these systems.

Recent results [3, 4] obtained with the use of Petri nets and P/T systems [10] show that the study of the generative variants of computing systems based on symbol objects (membrane systems, program machines, brane calculi, etc.) can be facilitated if someone considers the number of unbounded elements present in these systems. In the present paper we do not introduce the notation of Petri net and P/T systems but only one result obtained with their use. These information can be found in the just mentioned publications.

In particular [Corollary 2] from [4] indicates:

A P/T system with two unbounded elements has computational power equivalent to the one of program machines;

A P/T system with only unbounded number of tokens has computational power equivalent to the one of partially blind program machines;

A P/T system with only unbounded number of places has computational power equivalent to the one of restricted program machines (in this case restrictions in the composition of building blocks are present).

There *unbounded elements* refers to some components of the P/T systems (as, for instance, number of places and tokens) that are present in unbounded quantity. In [4] it is also proved that maximal parallelism is equivalent to the presence of an unbounded number of places. The results proved in [4] indicate that in the study of a computing system the number and kind of unbounded elements can give an indication (upper bounds and precise characterisation) of the computing power of the system.

The research reported in the present paper does not have the level of generality (i.e., the use of Petri nets) used in [4]. It refers to our initial results on the study of conformon-P systems having one 'extended' unbounded element: the value of the conformons ranges from $-\infty$ to $+\infty$, differently from previous studies in which it was ranging from 0 to $+\infty$.

2 Basic definitions

We assume the reader to have familiarity with basic concepts of formal language theory [6] and program machines [7]. We indicate with $\mathbb{N}$ the set of positive integers, $\mathbb{N}_0 = \{0\} \cup \mathbb{N}$ and $\mathbb{Z} = \mathbb{N}_0 \cup \{-i \mid i \in \mathbb{N}\}$ indicates the set of all integers (positive, negative and zero).

2.1 Program machines

A *program machine* (also known as *(multi)counter machines*, *multipushdown machines*, *register machines* and *counter automata*) with n counters ($n \in \mathbb{N}$) is defined as $M = (S, R, s_0, s_d)$, where S is a finite set of *states*, $s_0, s_d \in S$ are respectively called the *initial* and *final* states, R is the finite set of *instructions* of the form (s_i, l_-, s_g, s_u) or (s_i, l_+, s_q), with $s_i, s_g, s_u, s_q \in S,\ s_i \neq s_d, 1 \leq l \leq n$

A *configuration* of a program machine M with n counters is given by an element in the $n+1$-tuples $(s_j,\ \mathbb{N}_0^n),\ s_j \in S$. Given two configurations $(s_i, l_1, \ldots, l_n)$ and $(s'_j, l'_1, \ldots, l'_n)$ we define a *computational step* as $(s_i, l_1, \ldots, l_n) \vdash (s_j, l'_1, \ldots, l'_n)$:

- if $(s_i, l_-, s_g, s_u),\ l = l_p$ and $l_p \neq 0$, then $s_j = s_g,\ l'_p = l_p - 1,\ l'_k = l_k,\ k \neq p,\ 1 \leq k \leq n$;
  if $l = l_p$ and $l_p = 0$, then $s_j = s_u,\ l'_k = l_k,\ 1 \leq k \leq n$;
  (informally: in state s_i if the content of counter l is greater than 0, then subtract 1 from that counter and change state into s_g, otherwise change state into s_u);
- if $(s_i, l_+, s_q),\ l = l_p$, then $s_j = s_q, l'_p = l_p + 1,\ l'_k = l_k,\ k \neq p,\ 1 \leq k \leq n$;
  (informally: in state s_i add 1 to counter l and change state into s_q).

The reflexive and transitive closure of $\vdash$ is indicated by $\vdash^*$.

A *computation* is a finite sequence of transitions between configurations of a program machine M starting from the initial configuration $(s_0, l_1, \ldots, l_n)$ with

$l_1 \neq 0,\ l_k = 0,\ 2 \leq k \leq n$. If the last of such configurations has s_d as state, then we say that M *accepted* the number l_1. The set of numbers accepted by M is defined as $L(M) = \{l_1 \mid (s_0, l_1, \cdots, l_n) \vdash^* (s_d, l''_1, \cdots, l''_n)\}$. For every program machine it is possible to create another one accepting the same set of numbers and having all counters empty in the final state.

Partially blind program machines (also known as *partially blind multicounter machines*) were introduced in [5] and defined as program machines without test on zero. The only allowed operations are increase and decrease of one unit per time of the counters indicated as (s_i, l_+, s_q) and (s_i, l_-, s_g) respectively. In case the machine tries to subtract from a counter having value zero it stops in a non final state. In [5] it is also proved that such machines are strictly less powerful than non blind ones.

2.2 Conformon-P system with negative values

A *conformon-P system with negative values* has conformons, a name-value pair, as objects. If V is an alphabet (a finite set of letters), then we can define a conformon as $[\alpha, a]$, where $\alpha \in V$ and $a \in \mathbb{Z}$ (in our previous works on conformons, see for instance [1, 2], we considered $a \in \mathbb{N}_0$). We say that α is the *name* and a is the *value* of the conformon $[\alpha, a]$. If, for instance, $V = A, B, C, \ldots$, then $[A, 5], [C, 0], [Z, -14]$ are conformons, while $[AB, 21]$ and $[D, 0.5]$ are not.

Two conformons can interact according to an *interaction rule*. An interaction rule is of the form $r : \alpha \xrightarrow{n} \beta$, where r is the label of the rule (a kind of name, it makes easier to refer to the rule) $\alpha, \beta \in V$ and $n \in \mathbb{N}_0$, and it says that a conformon with name α can give n from its value to the value of a conformon having name β. If, for instance, there are conformons $[G, 5]$ and $[R, 9]$ and the rule $r : G \xrightarrow{3} R$, one application of r leads to $[G, 2]$ and $[R, 12]$, another application of r (to $[G, 2]$ and $[R, 12]$) leads to $[G, -1]$ and $[R, 15]$.

The compartments (membranes) present in a conformon-P system have a label (again, a kind of name which makes it easier to refer to a compartment), every label being different. Compartments can be unidirectionally connected to each other and for each connection there is a *predicate*. A predicate is an element of the set $\{\geq n, \leq n \mid n \in \mathbb{Z}\}$. Examples of predicates are: $\geq 5, \leq -2$, etc.

If, for instance, there are two compartments (with labels) m_1 and m_2 and there is a connection from m_1 to m_2 having predicate ≥ 4, then conformons having value greater or equal to 4 can pass from m_1 to m_2. In a time unit any number of conformons can move between two connected membranes as long as the predicate on the connection is satisfied. Notice that we have *unidirectional connections* that is: m_1 connected to m_2 does not imply that m_2 is connected to m_1. Moreover, each connection has its own predicate. If, for instance, m_1 is connected to m_2 and m_2 is connected to m_1, the two connections can have different predicates. It is possible to have multiple connections (with different predicates) between compartments.

The interaction with another conformon and the passage to another membrane are the only *operations* that can be performed by a conformon.

Formally, a *conformon-P system with negative values* of *degree* $m, m \geq 1$, is a construct $\Pi = (V, \mu, \alpha_a, ack, L_1, \ldots, L_m, R_1, \ldots, R_m)$, where V is an alphabet; $\mu = (N, E)$ is a *directed labelled graph* underlying Π. The set N contains *vertices* (the membrane compartments), while the set E defines directed labelled *edges* (the connections) between vertices.

In α_a the value of α can either be *input* or *output*, in the former case Π is an accepting device, in the latter case Π is a generating device, while $a \in \{1, \ldots, m\}$ indicates the input or output membrane, respectively. $ack \in N$ indicates the *acknowledgment membrane*.

The multisets L_i contain conformons associated to region i; R_i are finite sets of rules for conformons interaction associated to region i.

A *configuration* of Π is an m-tuple indicating the multisets of conformons present in each membrane of the system. A *transition* is the passage from one configuration to another as the consequence of the application of operations.

A *computation* is a finite sequence of transitions between configurations of a system Π starting from $(L_1, \ldots, L_m)$, the *initial configuration* characterized by the fact that no conformon is present in the acknowledgment membrane. If used as a generating device, then the result of a computation is given by the multisets of conformons associated to membrane a when any conformon is associated to membrane *ack*. When this happens the computation is halted, that is no other operation is performed even if it could. When a conformon is associated to the acknowledge membrane the number of conformons (counted with their multiplicity) associated to membrane a defines the *number generated* by Π.

If used as an accepting device, then the input is given by the multiset of conformons associated to a in the initial configuration. If Π reaches a configuration with any conformon in *ack*, then no other operation is performed even if it could and Π accepts the input.

Some of the conformon-P systems considered in this paper work under *maximal parallelism*: in every configuration the maximum number of operations that can be performed is performed. If in one configuration some operations are conflicting (so that they cannot be executed together as they involve the same conformons), then any maximum number of non conflicting operations is performed. The passage of two conformons through the same connection is considered as two different operations (similarly for the interactions of two different pairs of conformons due to one rule).

2.3 Some modules for conformon-P systems

In the following we use the concept of *module*: a group of membranes with conformons and interaction rules in a conformon-P system able to perform a specific task.

An example of module is a *splitter* [1]: a module that, when a conformon $[X, x]$ with $x \in \{x_1, \ldots, x_h\}, x_i < x_{i+1},\ 1 \leq i \leq h-1$ is associated with a specific membrane of it, it may pass such a conformon to other specific membranes according to its value x. A detailed splitter is depicted in Figure 1.a. Vertices outgoing a module representation of a splitter, Figure 1.b, have as predicates elements in the set $\{= n \mid \mathbb{N}_0\}$, this is a shorthand indicating the function performed by this module.

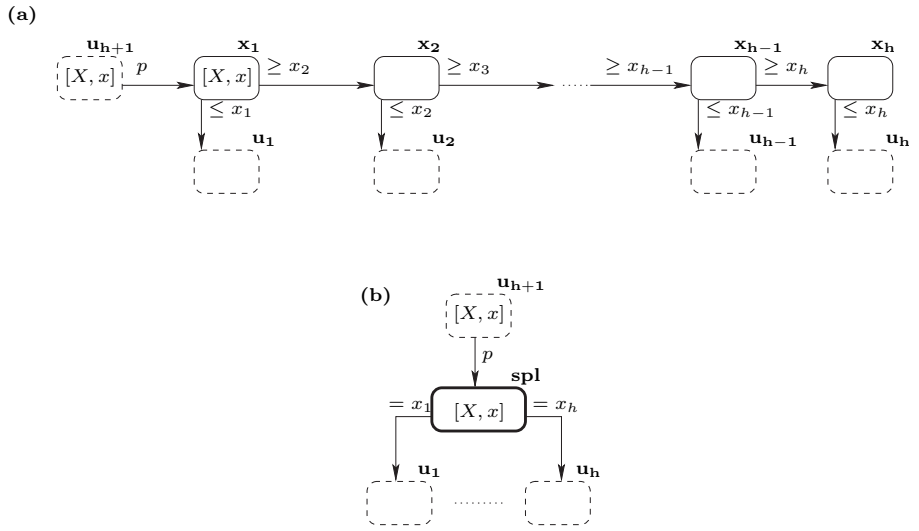

Fig. 1. A detailed splitter (a) and its module representation (b)

It should be clear that if a splitter is part of a conformon-P system with maximal parallelism, then the number of steps required to a conformon to pass from the u_{h+1} membrane to any other of the u membrane depends on the value of the conformon. If we consider the splitter depicted in Figure 1.a, a conformon present in membrane u_{h+1} requires only two steps to pass to membrane u_1 but it requires $h+1$ steps to pass to membrane u_h.

In order to have this time constant (equal to $h+1$ in the example), then delays (i.e., sequences of membranes) have to be introduced. We make this assumption for all the splitters considered in the proof of Theorem 1.

2.4 Figures in this paper

The representation of the conformon-P systems considered in this paper follows some rules aimed to a more concise representation an to an easier understanding of them.

The label of each membrane (a number) is indicated in **bold** on the top right corner of each compartment. Splitters are depicted by a thicker line, their label

(also in **bold**) starts with **spl**, and their edges have '=' as predicate. The module representation of a splitter is depicted in Figure 1.b.

Oval compartments with a label inside are shortcuts for membranes or modules.

Conformons present in the initial configuration of the system are depicted in **bold**, the remaining conformons are the ones that could be present in the membrane during the computation.

The predicate associated to an edge is indicated close to the edge.

Some predicates and the value of some conformons contain a slash (/). This is a shorthand for multiple predicates or values. For instance, the conformon $[A, 3/4]$ indicates that in a membrane the conformons $[A, 3]$ and $[A, 4]$ can be present. If there is a connection from membrane m_1 to membrane m_2 and the connection has predicate $\leq 0/ \geq 5$, then this is equivalent to two connections from m_1 to m_2, one with predicate ≤ 0 and the other with predicate ≥ 5.

If the conformon $[A, a]$ is present in m copies in a certain membrane, then this is indicated with $([A, a], m)$, where an unbounded number of copies is indicated with $+\infty$.

3 Results

Theorem 1. *The class of numbers accepted by conformon-P systems with negative values and with maximal parallelism coincides with the one accepted by program machines.*

Proof. This proofs follows from the one of [Theorem 2] from [1] (where priorities between interaction rules are present) and from [Theorem 1] from [4].

Figure 1 represents such a conformon-P system used as accepting device simulating a program machine. During this proof we refer to this figure.

For each state s_i of the simulated program machine there is a conformon with name s_i. For each instructions of the kind $(s_i, l_+, s_q) \in R$ there is a conformon with name $s'_{q,l}$; for each instruction of the kind $(s_i, l_-, s_g, s_u) \in R$ there are conformons with name $s''_{g,l}$ and $\bar{s}_{g,l}$. For the final state $s_d \in S$ there is one conformon with name s'''_d.

The initial configuration of the conformon-P system with priorities has all conformons with name $s'_{q,l}$, $s''_{g,l}$ and s'''_d and 0 as value in membrane 1; all the ones with name $\bar{s}_{g,l}$ and 1 as value in membrane 17; all the ones with name s_i and 0 as value in membrane 11 except the one with name of the initial state s_0 that is in membrane 1 with value 9 (in Figure 2 the generic conformon $[s_i, 9]$ is present in membrane 1); conformons $[a, 8]$ and $[c, 0]$ are initially present in membrane 6 and 13 respectively. Moreover for each counter l of the simulated machine there are an unbounded number of occurrences of the conformons $[l, 0]$ in membrane 8, while the input membrane (membrane 14 in the figure) contains as many copies of such conformons as the values k_l of the counters at the initial configuration of the simulated machine.

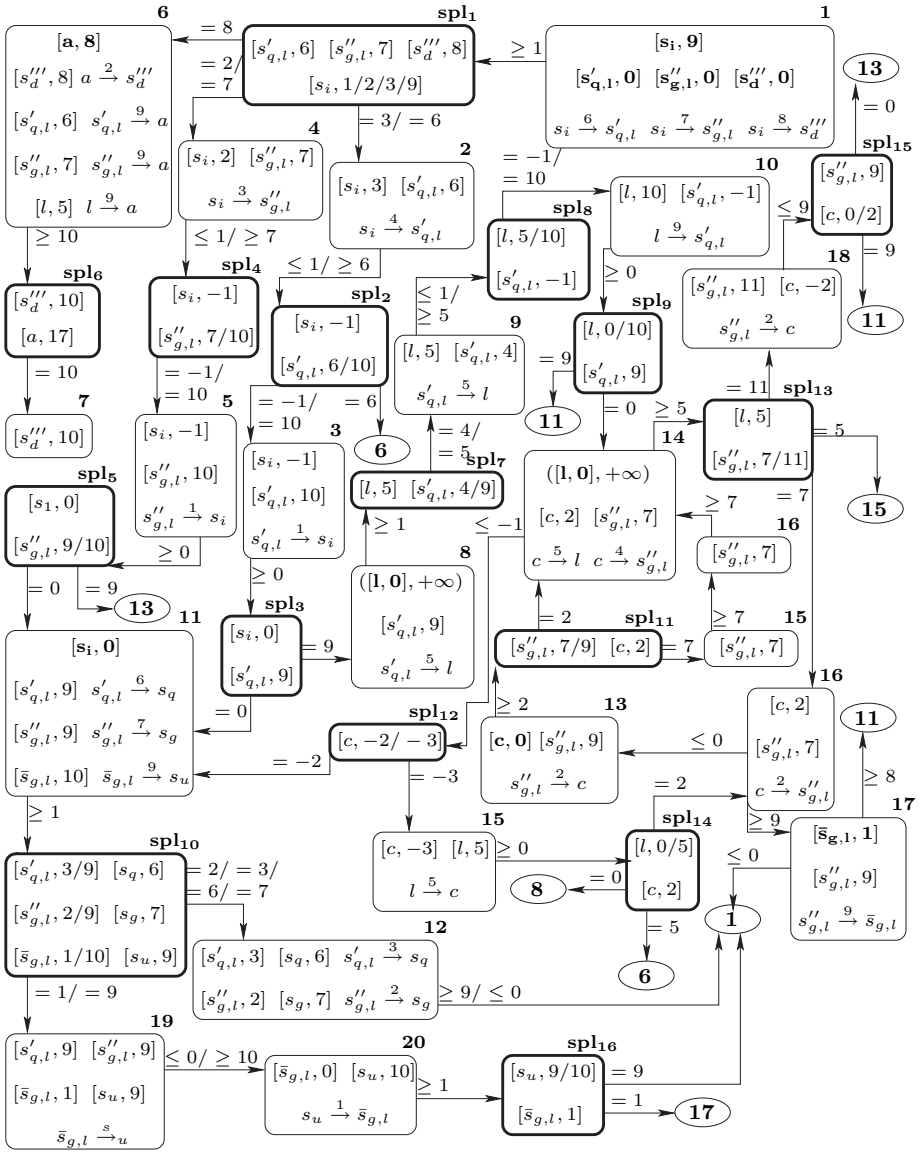

Fig. 2. The conformons-P system related to Theorem 1

The addition of one unit to one counter l is simulated moving one occurrence of the conformon $[l, 0]$ from membrane 8 to membrane 14; the subtraction of one unit is simulated with the passage of one occurrence of the same conformon from membrane 14 to membrane 8.

For each instruction of the type $(s_i, l_+, s_q) \in R$ there is in membrane 1 the rule $s_i \xrightarrow{6} s'_{q,l}$; for each instruction of the kind $(s_i, l_-, s_g, s_u) \in R$, there is in membrane 1 the rule $s_i \xrightarrow{7} s''_{g,l}$; for $s_d \in S$ there is in membrane 1 the rule $s_i \xrightarrow{8} s'''_d$.

Only one conformons of the kind $[s_i, 9]$ may be associated to membrane 1. When such a conformon is present in membrane 1, then one of the interaction rules indicate above can occur.

Let us consider now the case that the rule $s_i \xrightarrow{8} s'''_d$ is applied. As there is only one instance of $[s'''_d, 0]$ then the newly created $[s'''_d, 8]$ passes to spl_1 and from here to membrane 6. After the interaction two sets of rules are applicable: one with a second interaction of s_i and s'''_d and another with the passage of $[s'''_d, 8]$ and $[s_i, 1]$ to spl_1. Maximal parallelism forces this last set to be applied.

In membrane 6 $[s'''_d, 8]$ interacts with $[a, 8]$ such that $[s'''_d, 10]$ is created. When this happens this conformon passes first to spl_6 and then to membrane 7, the acknowledgment membrane, halting in this way the computation.

It is important to notice that the presence of $[a, 8]$ in membrane 6 is necessary for the halting of the computation. If in a configuration the conformon $[s'''_d, 8]$ passes in membrane 6 but $[a, 8]$ is not there, then the simulation does not halt.

If instead the interaction in membrane 1 involves the conformon with name s_i and either $s'_{q,l}$ or $s''_{g,l}$ (due to either $(s_i, l_+, s_q) \in R$ or $(s_i, l_i, s_g, s_u) \in R$), then the following sets of applicable operations are.

1. the conformon with name s_i can interact again with the same instance of either $s'_{q,l}$ or $s''_{g,l}$;
2. if there is either $(s_i, l_+, s_{q'}) \in R$ or $(s_i, l_1, s_{g'}, s_{u'}) \in R$, then the conformon with name s_i can interact with an instance of either $s'_{q',l}$ or $s''_{g',l}$ and the conformon created in the previous interaction (either $[s'_{q,l}, 6]$ or $[s''_{g,l}, 7]$) can pass to spl_1;
3. both s_i and either $[s'_{q,l}, 6]$ or $[s''_{g,l}, 7]$ can pass to spl_1.

Because of maximal parallelism only the second and third set of operations in the previous list can take place (as they contain two elements while the first set contains only one element).

If the second set occurs, then the system never reaches an halting configuration. This can be seen if, for instance, we consider the conformon $[s'_{q,l}, 6]$. Once in spl_1 this conformons passes to membrane 2, then to spl_2 and from here to membrane 6 where it interacts with $[a, 8]$. As a consequence of this interaction the conformon with name a passes to spl_6 so that the system does never halt.

The role of $[a, 8]$ in membrane 6 is just this: if in any stage during the computation the system performed an operation that does not follow the simulation of the program machine, then a conformon passes to membrane 6 and interacts with $[a, 8]$ making it unavailable for $[s'''_d, 8]$.

The creation of $[s_i, 3]$ and $[s'_{q,l}, 6]$ in membrane 1 starts the simulation of the instruction $(s_i, l_+, s_q) \in R$. As we said, the simulation of the addition of 1 to the value of the counter is performed with the passage of one instance of $[l, 0]$ from membrane 8 to membrane 14. When in membrane 2 $[s_i, 3]$ and $[s'_{q,l}, 6]$ interact,

$[s_i, -1]$ and $[s'_{q,l}, 10]$ are created and they pass to spl_2 (in case $[s'_{q,l}, 6]$ passes to spl_2, then the system never halts). From spl_2 $[s_i, -1]$ and $[s'_{q,l}, 10]$ pass together to membrane 3 where they interact, $[s_i, 0]$ and $[s'_{q,l}, 9]$ are created and pass to spl_3. From here $[s_i, 0]$ passes to membrane 11 while $[s'_{q,l}, 9]$ passes to membrane 8.

The membrane-splitter-membrane-splitter sequence that we just described (membrane 2 - spl_2 - membrane 3 - spl_3) is present in other parts of the system. This sequence allows to control the interaction of two conformons in a very precise way and to discard the outcome (i.e., conformons) of undesired interactions.

Once in membrane 8 the conformon with name $s'_{q,l}$ goes under another membrane-splitter sequence. In membrane 8 $[s'_{q,l}, 9]$ interacts with an instance of $[l, 0]$. After this interaction three sets of applicable operations are possible, this situation is similar to the one described before for membrane 1. In this case the undesired computation sees a conformons $[l, 5]$ passing from spl_8 to membrane 6, while a proper simulation sees a conformon $[l, 0]$ passing to membrane 14 and $[s'_{q,l}, 9]$ passing to membrane 11.

In membrane 11 $[s'_{q,l}, 9]$ interacts with $[s_q, 0]$ such that $[s'_{q,l}, 3]$ and $[s_q, 6]$ are created. Again a membrane-splitter sequence allows to create $[s'_{q,l}, 0]$ and $[s_q, 9]$ and let them pass to membrane 1. The instruction $(s_i, l_+, s_q) \in R$ has been performed and the system simulates the program machine being in state q.

The simulation of the instruction $(s_i, l_-, s_g, s_u) \in R$ starts with the interaction of $[s_i, 9]$ and $[s''_{g,l}, 0]$. If when this happens no $[l, 0]$ conformon is present in membrane 14, then the conformon $[s_u, 9]$ passes to membrane 1, otherwise an occurrence of $[l, 0]$ passes from membrane 14 to membrane 8 and the conformon $[s_g, 9]$ passes to membrane 1.

One interaction of $[s_i, 9]$ and $[s''_{g,l}, 0]$ in membrane 1 creates $[s_i, 2]$ and $[s''_{g,l}, 7]$ and, similarly to what described before, they can follow a membrane-splitter sequence at the end of which $[s_i, 2]$ is in membrane 11 and $[s''_{g,l}, 9]$ is in membrane 13.

In this last membrane $[s''_{g,l}, 9]$ interacts with $[c, 0]$ so that $[s''_{g,l}, 7]$ and $[c, 2]$ are created, then these two conformons pass to spl_{11}. From here $[c, 2]$ passes to membrane 14. The conformon $[s''_{g,l}, 7]$ also passes to this membrane but only after two steps (in the meantime it goes in membrane 15 and 16).

If in membrane 14 there is at least an occurrence of $[l, 0]$, then $[c, 2]$ interacts with any of these so that $[c, -3]$ and $[l, 5]$ are created (at the same time $[s''_{g,l}, 7]$ pass to membrane 17). In this configurations a few things can happen. Similarly to the second and third set of operations indicated in the list above $[c, -3]$ can either remain in membrane 14 and interact with another instance of $[l, 0]$ (if present) or it can pass to spl_{12} and from here to membrane 15. In any case $[l, 5]$ passes to spl_{13} and from here to membrane 15. If $[c, -3]$ is not present in this membrane when $[l, 5]$ is present, then this last conformon passes to spl_{14} and from here to membrane 6 (and here it interacts with $[a, 8]$ such that the system never halts).

It should be clear now that if $[c, -3]$ and $[l, 5]$ do not move together out of membrane 14 the system never halts. If they do so, then they pass to membrane

15 at the same time. Here $[l, 5]$ can either pass to spl_{14} (and then to membrane 6) or it can interact with $[c, -3]$ so that $[l, 0]$ and $[c, 2]$ are created and they pass together to spl_{14}. From here $[l, 0]$ passes to membrane 8 and $[c, 2]$ to membrane 16 (where it waits until $[s''_{g,l}, 7]$ arrives).

When $[s''_{g,l}, 7]$ passes to membrane 14 it can be that the conformon with name c is there or not. This last conformon can be in membrane 14 for two reasons: either no occurrence of $[l, 0]$ was in that membrane, or one occurrence of $[l, 0]$ was there and $[c, -2]$ did not pass to spl_{12}. We know from the above that in this last case the system does not halt (because an $[l, 2]$ is heading membrane 6), so we are not going to discuss the consequences of the interaction between $[s''_{g,l}, 7]$ and the conformon with name c when this last has a negative value.

If $[c, 2]$ is present in membrane 14 when $[s''_{g,l}, 7]$ arrives there, too, then two things can happen: the two conformons interact or not. In this last case $[s''_{g,l}, 7]$ passes to spl_{13} and from here to membrane 16 and no operation can happen in the system. If instead the two conforms interact, then $[s''_{g,l}, 11]$ and $[c, -1]$ are created and they pass to membrane 18 (through spl_{13} and spl_{12}, respectively). Here either $[c, -2]$ passes to spl_{15} and no further operation is applied, or the two conformons interact so to create $[s''_{g,l}, 9]$ and $[c, 0]$ and then these two conformons pass to membrane 11 and 13, respectively. So when $[s''_{g,l}, 9]$ is present in spl_{15}, then in the simulation the counter l was empty.

What happens to the conformon with name $s''_{g,l}$ in membrane 11 is similar to what discussed for the conformon with name $s'_{q,l}$ earlier on. The result of these operations is that $[s_g, 9]$ and $[s''_{g,l}, 0]$ are created and pass to membrane 1.

We still have to discuss the case in which no conformon with name c is present in membrane 14 when $[s''_{g,l}, 7]$ arrives. Here maximal parallelism forces this conformon to pass to spl_{13} and from here to membrane 16 where $[c, 2]$ is also present. This means that if $[c, 2]$ and $[s''_{g,l}, 7]$ are present in membrane 16, then the simulation of the subtraction of 1 from counter l has been performed.

When in membrane 16 $[c, 2]$ and $[s''_{g,l}, 7]$ interact so that $[c, 0]$ and $[s''_{g,l}, 9]$ are created and pass to membrane 13 and 17, respectively. In this last membrane $[s''_{g,l}, 9]$ interacts with $[\bar{s}_{g,l}, 1]$ so that $[s''_{g,l}, 0]$ and $[\bar{s}_{g,l}, 10]$ are created. Because of maximal parallelism these last two conformons pass to membrane 1 and 11, respectively.

What happens to the conformon with name $\bar{s}_{g,l}$ in membrane 11 is similar to what discussed for the conformon with name $s'_{q,l}$ earlier on. The result of these operations is that $[s_u, 9]$ and $[\bar{s}_{g,l}, 1]$ are created and pass to membrane 1 and 17 respectively.

If on a given input the program machine reaches an halting state, then the simulating conformon-P system can reach a final configuration.

The assumption that a program machine can simulate any such conformon-P system derives from the Turing-Church thesis. □

In the figure related to the following proof some conformons have a parametric value of the kind $a + bn,\ a, b \in \mathbb{N}, n \geq 0$. This indicates all the possible values that a conformons can have as a consequence of interactions. If, for instance, the conformons $[A, a], [C, c]$ and the interaction rule $C \xrightarrow{b} A$ are present in the same membrane, then the value of the A conformon can change into $a + bn$, where n indicates the number of interactions between the A and the C conformon.

Theorem 2. *Conformon-P systems with negative values and without maximal parallelism can simulate partially blind program machines.*

Proof. This proof follows from the one of [Theorem 1] from [1].

Figure 2 represents such a conformon-P system used as an accepting device simulating a program machine. During this proof we refer to this figure.

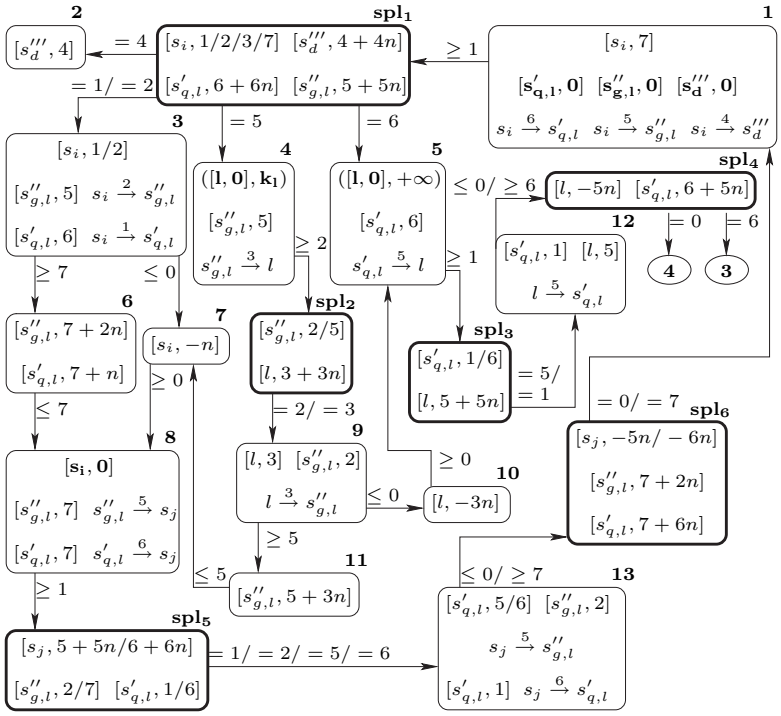

Fig. 3. The conformons-P system related to Theorem 2

For each state s_i of the simulated program machine there is a conformon with name s_i. For each instruction of the kind $(s_i, l_+, s_q) \in R$ there is a conformon with name $s'_{q,l}$; for each instruction of the kind $(s_i, l_-, s_g) \in R$ there is a conformon with name $s''_{g,l}$. For the final state $s_d \in S$ there is one conformon with name s'''_d.

The initial configuration of the conformon-P system has all conformons with name $s'_{q,l}$, $s''_{g,l}$ and s'''_d and 0 as value in membrane 1; all the ones with name s_i and 0 as value in membrane 8 except the one with name of the initial state s_0 that is in membrane 1 with value 7 (in Figure 3 the generic conformon $[s_i, 7]$ is present in membrane 1).

Moreover, for each counter l of the simulated machine there are an unbounded number of occurrences of the conformons $[l, 0]$ in membrane 5, while the input membrane (membrane 4 in the figure) contains as many copies of such conformons as the values k_l of the counters at the initial configuration of the simulated machine. The addition of one unit to one counter l is simulated moving one occurrence of the conformon $[l, 0]$ from membrane 5 to membrane 4; the subtraction of one unit is simulated with the passage of one occurrence of the same conformon from membrane 4 to membrane 5.

For each instruction of the type $(s_i, l_+, s_q) \in R$ there is in membrane 1 the rule $s_i \xrightarrow{6} s'_{q,l}$; for each instruction of the kind $(s_i, l_-, v) \in R$, there is in membrane 1 the rule $s_i \xrightarrow{5} s''_{g,l}$; for $s_d \in S$ there is in membrane 1 the rule $s_i \xrightarrow{4} s'''_d$.

For any configuration of the conformon-P system only one conformon of the kind $[s_i, 7]$ may be associated to membrane 1. As we already said, initially this conformon is the one related to the initial state of the program machine.

When a conformon of the kind $[s_i, 7]$ is present in membrane 1, then one of the interaction rules indicate above can occur.

Let us consider now the case that the rule $s_i \xrightarrow{8} s'''_d$ can be applied. Of course this rule can be applied more than once, if this happens the value of the s_i conformon goes below 0 and the one of the s'''_d conformon is $4n$. If n is at least 1, then the s'''_d conformon can pass to spl_1, but only if $n = 1$, then $[s'''_d, 4]$ passes to membrane 2, the acknowledgment membrane, halting in this way the computation.

This process of 'filtering out' (with splitters) conformons with an undesired value is present in many places in this conformon-P system. If two conformons over interacted, then the system never reaches an halting configuration.

If instead a rule of the kind $s_i \xrightarrow{6} s'_{q,r}$ is applied, then $[s'_{q,r}, 6 + 6n]$ can pass to spl_1, but only $[s'_{q,r}, 6]$ can pass to membrane 5. If in membrane 1 $[s_i, 1]$ is produced, then it passes to membrane 3 (through spl_1).

In membrane 5 two things can happen:

1. $s'_{q,l}$ interacts several times with the same l conformon;
2. $s'_{q,l}$ interacts with different l conformons.

The only case such that $[s'_{q,l}, 1]$ is produced and passes to membrane 12 (through spl_3) is when $[s'_{q,l}, 6]$ interacts only once with one $[l, 0]$ conformons. In all the other cases the value of the $s'_{q,l}$ conformon becomes negative and such a conformon does not pass to membrane 12 (in this way the system never halts).

If $[s'_{q,l}, 1]$ is produced, then also one $[l, 5]$ is produced and, once in membrane 12, these two conformons interact so that $[s'_{q,l}, 6]$ and $[l, 0]$ are recreated (because of spl_4, an over interaction in membrane 12 leads the resulting conformons to remain in that splitter) and they can pass to membrane 3 and 4, respectively.

The passage of an instance of $[l, 0]$ from membrane 5 to membrane 4 simulates the subtraction of one unit from the l counter. If no $[l, 0]$ conformon is present in membrane 5 when a $s'_{q,l}$ conformon gets there, then the system never reaches an halting configuration.

In membrane 3 s_i and $s'_{q,l}$ can interact such that $[s_i, -n]$ and $[s'_{q,l}, 7+n], n \geq 0$ are produced and they pass to membrane 7 and 6 respectively. Only if $n = 0$ (which means that only one interaction took place), then $[s_i, 0]$ and $[s'_{q,l}, 7]$ pass to membrane 8. Here $s'_{q,l}$ interacts with s_j and, in a way similar to what described until now, this can lead to the production of $[s_j, 7]$ and $[s'_{q,l}, 0]$ and these two conformons can pass to membrane 1.

The simulation of an instruction of the kind (s_i, l_-, v) is performed in a similar way.

If on a given input the partially blind program machine reaches an halting state, then the simulating conformon-P system can reach a final configuration. □

4 Final Remarks

The results reported in this paper leave us perplex. How shall we interpret these results in terms of unbounded elements [4]? Must the range of values (from $-\infty$ to $+\infty$) be regarded as one or two unbounded elements or as no unbounded elements at all? (because the two infinities 'cancel' each other)

In this last case [Corollary 2] from [4] is confirmed as Theorem 1 has two unbounded elements (number of conformons and maximal parallelism) and Lemma 2 only one (number of conformons). This also implies that it is possible to prove that a partially blind program machine can simulate any conformon-P system with negative values without maximal parallelism.

In case the range of values is regarded as one unbounded element, then it should be possible to prove that the computational power of conformon-P system with negative values is equivalent to the one of program machines. This implies that Theorem 1 is redundant (because it uses three unbounded elements).

In case the range of values is regarded as two unbounded elements, then some of the results reported in [4] should be extended and made more general.

We believe that the range of values should be regarded an one unbounded element.

Another problem on this line of research regards the characterization of blind program machines in terms of unbounded elements. *Blind program machines* (also known as *blind multicounter machines*) [5] are program machines that cannot sense (so neither halt or check) if the value of a counter is zero. It is know that these machines are strictly less computationally powerful then partially blind program machines.

References

1. P. Frisco. The conformon-P system: A molecular and cell biology-inspired computability model. *Theoretical Computer Science*, 312(2-3):295–319, 2004.
2. P. Frisco. Infinite hierarchies of conformon-P systems. In H. J. Hoogeboom, G. Păun, G. Rozenberg, and A. Salomaa, editors, *Membrane Computing*, volume 4361 of *Lecture Notes in Computer Science*, pages 395–408. Springer-Verlag, Berlin, Heidelberg, New York, 2006. 7^{th} International Workshop, WMC 2006, Leiden, The Netherlands, July 17-21, 2006, Revised, Selected, and Invited Papers.
3. P. Frisco. P systems, Petri nets, and Program machines. In R. Freund, G. Lojka, M. Oswald, and G. Păun, editors, *Workshop on Membrane Computing*, volume 3850 of *Lecture Notes in Computer Science*, pages 209–223. Springer-Verlag, Berlin, Heidelberg, New York, 2006. Proceedings of the 6^{th} International Workshop on Membrane Computing (WMC6).
4. P. Frisco. An hierarchy of recognising computational processes, 2007. submitted, also available at `http://www.macs.hw.ac.uk:8080/techreps/build_table.jsp` as Tech. Rep. 0047.
5. S. A. Greibach. Remarks on blind and partially blind one-way multicounter machines. *Theoretical Computer Science*, 7:311–324, 1978.
6. J. E. Hopcroft and D. Ullman. *Introduction to Automata Theory, Languages, and Computation.* Addison-Wesley, 1979.
7. M. L. Minsky. *Computation: Finite and Infinite Machines.* Automatic computation. Prentice-Hall, 1967.
8. G. Păun. Computing with membranes. *Journal of Computer and System Sciences*, 1(61):108–143, 2000.
9. G. Păun. *Membrane Computing. An Introduction.* Springer-Verlag, Berlin, Heidelberg, New York, 2002.
10. W. Reisig and G. Rozenberg, editors. *Lectures on Petri Nets I: Basic Models*, volume 1491 of *Lecture Notes in Computer Science.* Springer-Verlag, Berlin, Heidelberg, New York, 1998.