Alternative Space Definitions for P Systems with Active Membranes

Artiom Alhazov[1], Alberto Leporati[2], Luca Manzoni[3], Giancarlo Mauri[2], and Claudio Zandron[2]

[1] Vladimir Andrunachievici Institute of Mathematics and Computer Science
Academiei 5, Chișinău, MD-2028, Moldova
artiom@math.md

[2] Dipartimento di Informatica, Sistemistica e Comunicazione (DISCo)
Università degli Studi di Milano-Bicocca
Viale Sarca 336, 20126 Milan, Italy.
{alberto.leporati,giancarlo.mauri,claudio.zandron}@unimib.it

[3] Dipartimento di Matematica e Geoscienze
Università degli Studi di Trieste
lmanzoni@units.it

Summary. The first definition of space complexity for P systems was based on an hypothetical real implementation by means of biochemical materials, and thus it assumes that every single object or membrane requires some constant physical space. This is equivalent to using a unary encoding to represent multiplicities for each object and membrane.

A different approach can also be considered, having in mind an implementation of P systems in silico; in this case, the multiplicity of each object in each membrane can be stored using binary numbers, thus reducing the amount of needed space. In this paper, we give a formal definition for this alternative space complexity measure, we define the corresponding complexity classes and we compare such classes both with standard space complexity classes and with complexity classes defined in the framework of P systems considering the original definition of space.

Key words: Membrane Systems, Computational Complexity, Space Complexity

1 Introduction

P systems with active membranes have been introduced in [6], considering the idea of generating new membranes through division of existing ones. The exponential amount of resources that can be obtained in this way, in a polynomial number of computation steps, naturally leads to the definition of new complexity classes to be compared with the standard ones.

Initially, the research activity focused on the investigation of time complexity, for the various classes of P systems that can be obtained by introducing different features.

The first definition of space complexity for P systems has been introduced in [8], and it was based on an hypothetical real implementation by means of biochemical materials such as cellular membranes and chemical molecules. Under this assumption, it was assumed that every single object or membrane requires some constant physical space, and this is equivalent to using a unary encoding to represent multiplicities.

A different approach can also be considered, focusing the definition on the simulative point of view. By considering an implementation of P systems in silico, it is not strictly necessary to store information concerning every single object: the multiplicity of each object in each membrane can be stored using binary numbers, thus reducing the amount of needed space.

In this paper, we give a formal definition for this alternative space complexity measure, we define the corresponding complexity classes and we compare such classes both with standard space complexity classes and with complexity classes defined in the framework of P systems considering the original definition of space [8]. In particular, we will give partial results concerning the use of constant, polynomial or exponential amount of space, respectively.

The paper is organized as follows. In Section 2 we recall some definitions concerning P systems with active membranes and space requirements in P systems computations. In Section 3, we introduce a different definition for measuring space (which we call *binary space* to underline that information concerning objects is stored in binary) and we give some results following immediately from this definition. In Section 4 we compare the new binary space complexity classes with standard complexity classes and with space complexity classes for P systems based on the standard definition of space. Finally section 5 draws some conclusions and presents some future research topics on this subject.

2 Basic definitions

In this section, we shortly recall some definitions that will be useful while reading the rest of the paper. For a complete introduction to P systems, we refer the reader to *The Oxford Handbook of Membrane Computing* [7].

Definition 1. *A* P system with active membranes *having initial degree* $d \geq 1$ *is a tuple* $\Pi = (\Gamma, \Lambda, \mu, w_{h_1}, \ldots, w_{h_d}, R)$*, where:*

- Γ *is an alphabet, i.e., a finite non-empty set of symbols, usually called* objects*; in the following, we assume* $\Gamma = \{O_1, O_2, \ldots, O_n\}$
- Λ *is a finite set of labels for the membranes;*
- μ *is a membrane structure (i.e., a rooted* unordered *tree, usually represented by nested brackets) consisting of* d *membranes, labelled by elements of* Λ *in a one-*

to-one way, defining regions (the space between a membrane and all membranes immediately inside it, if any);

- $w_{h_1}, \ldots, w_{h_d}$*, with* $h_1, \ldots, h_d \in \Lambda$*, are strings over* Γ *describing the initial multisets of objects placed in the* d *regions of* μ*;*
- R *is a finite set of rules over* Γ*.*

Membranes are polarized, that is, they have an attribute called *electrical charge*, which can be neutral (0), positive (+) or negative (−).

A P system can made a computation step by applying its rules to modify the membrane structure and/or the membrane content.The following types of rules can be used during the computation:

- *Object evolution rules*, of the form $[a \to w]_h^\alpha$
  They can be applied inside a membrane labelled by h, having charge α and containing at least an occurrence of the object a; the object a is rewritten into the multiset w (i.e., a is removed from the multiset in h and replaced by the objects in w).
- *Send-in communication rules*, of the form $a\,[\,]_h^\alpha \to [b]_h^\beta$
  They can be applied to a membrane labelled by h, having charge α and such that the external region contains at least an occurrence of the object a; the object a is sent into h becoming b and, simultaneously, the charge of h is changed to β.
- *Send-out communication rules*, of the form $[a]_h^\alpha \to [\,]_h^\beta\, b$
  They can be applied to a membrane labelled by h, having charge α and containing at least an occurrence of the object a; the object a is sent out from h to the outside region becoming b and, simultaneously, the charge of h is changed to β.
- *Dissolution rules*, of the form $[a]_h^\alpha \to b$
  They can be applied to a membrane labelled by h, having charge α and containing at least an occurrence of the object a; the membrane h is dissolved and its contents are left in the surrounding region unaltered, except that an occurrence of a becomes b.
- *Elementary division rules*, of the form $[a]_h^\alpha \to [b]_h^\beta\, [c]_h^\gamma$
  They can be applied to a membrane labelled by h, having charge α, containing at least an occurrence of the object a but having no other membrane inside (in this case the membrane is said to be *elementary*); the membrane is divided into two membranes having both label h and charges β and γ, respectively; the object a is replaced, respectively, by b and c in the two new membranes, while the other objects in the initial multiset are copied to both membranes.
- *(Weak) Non-elementary division rules*, of the form $[a]_h^\alpha \to [b]_h^\beta\, [c]_h^\gamma$
  These rules operate just like division for elementary membranes, but they can be applied to non–elementary membranes, containing membrane substructures and having a label h. Like the objects, the substructures inside the dividing membrane are replicated in the two new copies of it.

A configuration of a P system with active membranes is described by the current membrane structure (including the electrical charge of each membrane) and the multisets located in the corresponding regions. A computation step changes the current configuration according to the following set of principles:

- Each object and membrane can be subject to at most one rule per step, except for object evolution rules (inside each membrane several evolution rules can be applied simultaneously).
- The application of rules is *maximally parallel*: each object appearing on the left-hand side of evolution, communication, dissolution or division rules must be subject to exactly one of them (unless the current charge of the membrane prohibits it). The same principle applies to each membrane that can be involved in communication, dissolution, or division rules. In other words, the only objects and membranes that do not evolve are those associated with no rule, or only to rules that are not applicable due to the electrical charges.
- When several conflicting rules can be applied at the same time, a nondeterministic choice is performed; this implies that, in general, multiple possible configurations can be reached as the result of a computation step.
- In each computation step, all the chosen rules are applied simultaneously (in an atomic way). However, in order to clarify the operational semantics, each computation step is conventionally described as a sequence of micro-steps as follows. First, all evolution rules are applied inside the elementary membranes, followed by all communication, dissolution and division rules involving the membranes themselves; this process is then repeated to the membranes containing them, and so on towards the root (outermost membrane). In other words, the membranes evolve only after their internal configuration has been updated. For instance, before a membrane division occurs, all chosen object evolution rules must be applied inside it; in this way, the objects that are duplicated during the division are already the final ones.
- The outermost membrane cannot be divided or dissolved, and any object sent out from it cannot re-enter the system again.

A *halting computation* of the P system Π is a finite sequence of configurations $\boldsymbol{\mathcal{C}} = (\mathcal{C}_0, \ldots, \mathcal{C}_k)$, where $\mathcal{C}_0$ is the initial configuration, every $\mathcal{C}_{i+1}$ is reachable from $\mathcal{C}_i$ via a single computation step, and no rules of Π are applicable in $\mathcal{C}_k$. A *non-halting* computation $\boldsymbol{\mathcal{C}} = (\mathcal{C}_i : i \in \mathbb{N})$ consists of infinitely many configurations, again starting from the initial one and generated by successive computation steps, where the applicable rules are never exhausted.

P systems can be used as language *recognizers* by employing two distinguished objects *yes* and *no*; exactly one of these must be sent out from the outermost membrane, and only in the last step of each computation, in order to signal acceptance or rejection, respectively; we also assume that all computations are halting.

In order to solve decision problems (i.e., decide languages over an alphabet Σ), we use *families* of recognizer P systems $\boldsymbol{\Pi} = \{\Pi_x : x \in \Sigma^\star\}$. Each input x is associated with a P system Π_x that decides the membership of x in the language

$L \subseteq \Sigma^\star$ by accepting or rejecting. The mapping $x \mapsto \Pi_x$ must be efficiently computable for each input length [4].

These families of *recognizer* P systems can be used to solve decision problems as follows.

Definition 2. *Let Π be a P system whose alphabet contains two distinct objects yes and no, such that every computation of Π is halting and during each computation exactly one of the objects yes, no is sent out from the skin to signal acceptance or rejection. If all the computations of Π agree on the result, then Π is said to be* confluent*; if this is not necessarily the case, then it is said to be* non-confluent *and the global result is acceptance if and only if there exists an accepting computation.*

Definition 3. *Let $L \subseteq \Sigma^\star$ be a language, $\mathcal{D}$ a class of P systems (i.e. a set of P systems using a specific subset of features) and let $\mathbf{\Pi} = \{\Pi_x \mid x \in \Sigma^\star\} \subseteq \mathcal{D}$ be a family of P systems, either confluent or non-confluent. We say that $\mathbf{\Pi}$* decides *L when, for each $x \in \Sigma^\star$, $x \in L$ if and only if Π_x accepts.*

Complexity classes for P systems are defined by imposing a uniformity condition on $\mathbf{\Pi}$ and restricting the amount of time or space available for deciding a language.

Definition 4. *Consider a language $L \subseteq \Sigma^\star$, a class of recognizer P systems $\mathcal{D}$, and let $f\colon \mathbb{N} \to \mathbb{N}$ be a proper complexity function (i.e. a "reasonable" one, see [5, Definition 7.1]). We say that L belongs to the complexity class $\mathbf{MC}^*_{\mathcal{D}}(f)$ if and only if there exists a family of confluent P systems $\mathbf{\Pi} = \{\Pi_x \mid x \in \Sigma^\star\} \subseteq \mathcal{D}$ deciding L such that:*

- $\mathbf{\Pi}$ *is* semi-uniform*, i.e. there exists a deterministic Turing machine which, for each input $x \in \Sigma^\star$, constructs the P system Π_x in polynomial time with respect to $|x|$;*
- $\mathbf{\Pi}$ *operates in time f, i.e. for each $x \in \Sigma^\star$, every computation of Π_x halts within $f(|x|)$ steps.*

In particular, a language $L \subseteq \Sigma^\star$ belongs to the complexity class $\mathbf{PMC}^*_{\mathcal{D}}$ if and only if there exists a semi-uniform family of confluent P systems $\mathbf{\Pi} = \{\Pi_x \mid x \in \Sigma^\star\} \subseteq \mathcal{D}$ deciding L in polynomial time.

The analogous complexity classes for non*-confluent P systems are denoted by $\mathbf{NMC}^*_{\mathcal{D}}(f)$ and $\mathbf{NPMC}^*_{\mathcal{D}}$.*

Another set of complexity classes is defined in terms of *uniform* families of recognizer P systems:

Definition 5. *Consider a language $L \subseteq \Sigma^\star$, a class of recognizer P systems $\mathcal{D}$, and let $f\colon \mathbb{N} \to \mathbb{N}$ be a proper complexity function. We say that L belongs to the complexity class $\mathbf{MC}_{\mathcal{D}}(f)$ if and only if there exists a family of confluent P systems $\mathbf{\Pi} = \{\Pi_x \mid x \in \Sigma^\star\} \subseteq \mathcal{D}$ deciding L such that:*

- $\mathbf{\Pi}$ *is* uniform*, i.e. for each* $x \in \Sigma^\star$ *deciding whether* $x \in L$ *is performed as follows: first, a polynomial-time deterministic Turing machine, given the length* $n = |x|$ *as a unary integer, constructs a P system* Π_n *with a distinguished input membrane; then, another polynomial-time deterministic Turing machine computes an encoding of the string* x *as a multiset* w_x*, which is finally added to the input membrane of* Π_n*, thus obtaining a P system* Π_x *that accepts if and only if* $x \in L$*.*
- $\mathbf{\Pi}$ *operates in time* f*, i.e. for each* $x \in \Sigma^\star$*, every computation of* Π_x *halts within* $f(|x|)$ *steps.*

In particular, a language $L \subseteq \Sigma^\star$ *belongs to the complexity class* $\mathbf{PMC}_{\mathcal{D}}$ *if and only if there exists a uniform family of confluent P systems* $\mathbf{\Pi} = \{\Pi_x \mid x \in \Sigma^\star\} \subseteq \mathcal{D}$ *deciding* L *in polynomial time.*

The analogous complexity classes for non-*confluent P systems are denoted by* $\mathbf{NMC}_{\mathcal{D}}(f)$ *and* $\mathbf{NPMC}_{\mathcal{D}}$*.*

As stated in the Introduction, the first definition of space complexity for P systems introduced in [8] considered a possible real implementation with biochemical materials, thus assuming that every single object and membrane requires some constant physical space. Such a definition (in the improved version from [3], taking into account also the space required by the labels for membranes and the alphabet of symbols) is the following:

Definition 6. *Considering a configuration* $\mathcal{C}$ *of a P system* Π*, its* size $|\mathcal{C}|$ *is the number of membranes in the current membrane structure multiplied by* $\log|\Lambda|$*, plus the total number of objects from* Γ *they contain multiplied by* $\log|\Gamma|$*. If* $\boldsymbol{\mathcal{C}} = (\mathcal{C}_0, \ldots, \mathcal{C}_k)$ *is a computation of* Π*, then the* space required by $\boldsymbol{\mathcal{C}}$ *is defined as*

$$|\boldsymbol{\mathcal{C}}| = \max\{|\mathcal{C}_0|, \ldots, |\mathcal{C}_k|\}.$$

The space required by Π *itself is then obtained by computing the space required by all computations of* Π *and taking the supremum:*

$$|\Pi| = \sup\{|\boldsymbol{\mathcal{C}}| : \boldsymbol{\mathcal{C}} \text{ is a computation of } \Pi\}.$$

Finally, let $\mathbf{\Pi} = \{\Pi_x : x \in \Sigma^\star\}$ *be a family of recognizer P systems, and let* $s\colon \mathbb{N} \to \mathbb{N}$*. We say that* $\mathbf{\Pi}$ operates within space bound s *if and only if* $|\Pi_x| \leq s(|x|)$ *for each* $x \in \Sigma^\star$*.*

Analogously to what has been done for time complexity classes, we can define space complexity classes. By $\mathbf{MCSPACE}_{\mathcal{D}}(f(n))$ (resp. $\mathbf{MCSPACE}^*_{\mathcal{D}}(f(n))$) we denote the class of languages which can be decided by uniform (resp. semi-uniform) families of confluent P systems of type $\mathcal{D}$ (for example, when we refer to P systems with active membranes, we denote this by setting $\mathcal{D} = \mathcal{AM}$), where each $\Pi_x \in \mathbf{\Pi}$ operates within space bound $f(|x|)$.

In particular, the class of problems solvable in polynomial space by uniform (resp. semi-uniform) confluent systems is denoted by $\mathbf{PMCSPACE}_{\mathcal{D}}$ (resp.

$\mathbf{PMCSPACE}^*_{\mathcal{D}}$), and the class of problems solvable in exponential space by uniform (resp. semi-uniform) confluent systems is denoted by $\mathbf{EXPMCSPACE}_{\mathcal{D}}$ (resp. $\mathbf{EXPMCSPACE}^*_{\mathcal{D}}$).

The corresponding classes for non-confluent systems are $\mathbf{NPMCSPACE}_{\mathcal{D}}$ (resp. $\mathbf{NPMCSPACE}^*_{\mathcal{D}}$) and $\mathbf{NEXPMCSPACE}_{\mathcal{D}}$ (resp. $\mathbf{NEXPMCSPACE}_{\mathcal{D}}$).

3 An Alternative Definition of Space Complexity for P Systems

In this section, we first give a different definition of space complexity for P systems with active membranes. This definition considers the information stored in the objects of the systems, and not the single objects themselves. In other words, we store, using binary numbers, the multiplicity of each object in each membrane, thus reducing the amount of needed space with respect to the definition of space given in the previous section. We will refer to this definition of space by *binary space*, and we will add a symbol B where appropriate, to distinguish between the definitions referring to this new measure and the definitions recalled in the previous section.

Definition 7. *Consider a configuration $\mathcal{C}$ of a P system Π. Let us denote by $h_1, h_2, ..., h_z$ the membranes of the current membrane structure (we stress the fact that z can be smaller, equal, or greater than the initial number of membranes d, due to dissolution and duplication of membranes), and by $|O_{i,j}|$ the multiplicity of object i within region j. The binary size $|\mathcal{C}|_B$ of a configuration $\mathcal{C}$ is defined as:*

$$|\mathcal{C}|_B = z \cdot \log|\Lambda| + \left(\sum_{j=1}^{z}\sum_{i=1}^{n}\lceil\log(|O_{i,j}|)\rceil\right) \cdot \log|\Gamma|$$

that is the number of membranes in the current membrane structure multiplied by $\log|\Lambda|$, plus the number of bits required to store the amount of each object in each membrane multiplied by $\log|\Gamma|$.

If $\boldsymbol{\mathcal{C}} = (\mathcal{C}_0, \ldots, \mathcal{C}_k)$ is a computation of Π, then the binary space *required by $\boldsymbol{\mathcal{C}}$ is defined as*

$$|\boldsymbol{\mathcal{C}}|_B = \max\{|\mathcal{C}_0|_B, \ldots, |\mathcal{C}_k|_B\}.$$

The binary space *required by Π itself is then obtained by computing the binary space required by all computations of Π and taking the supremum:*

$$|\Pi|_B = \sup\{|\boldsymbol{\mathcal{C}}|_B : \boldsymbol{\mathcal{C}} \text{ is a computation of } \Pi\}.$$

Finally, let $\boldsymbol{\Pi} = \{\Pi_x : x \in \Sigma^\star\}$ be a family of recognizer P systems, and let $s\colon \mathbb{N} \to \mathbb{N}$. We say that $\boldsymbol{\Pi}$ operates within binary space bound s *if and only if $|\Pi_x|_B \leq s(|x|)$ for each $x \in \Sigma^\star$.*

We can thus define space complexity classes considering this new size measure like we did in the previous section. By $\mathbf{MCBSPACE}_{\mathcal{D}}(f(n))$ we denote the class of languages which can be decided by uniform families of confluent P systems of type $\mathcal{D}$, where each $\Pi_x \in \mathbf{\Pi}$ operates within space bound $f(|x|)$, considering this new definition of binary space. Similarly, we can define the usual complexity classes like we did in the previous section, simply adding a B to underline the use of this new definition of space. For instance, the class of problems solvable in polynomial binary space will be denoted by $\mathbf{PMCBSPACE}_{\mathcal{D}}$.

Once these notions have been defined, we are ready to state some results obtained by considering various complexity classes defined in terms of binary space. Just like it happens with the classes based on the original definition of space given in [8], some results follow immediately from the definitions (here we state results for semi-uniform families, but it is easy to see that they also hold in the uniform case):

Proposition 1 *The following inclusions hold:*

$$\mathbf{PMCBSPACE}^{\star}_{\mathcal{D}} \subseteq \mathbf{EXPMCBSPACE}^{\star}_{\mathcal{D}}$$
$$\mathbf{NPMCBSPACE}^{\star}_{\mathcal{D}} \subseteq \mathbf{NEXPMCBSPACE}^{\star}_{\mathcal{D}}.$$

Proposition 2 $\mathbf{MCBSPACE}^{\star}_{\mathcal{D}}(f) \subseteq \mathbf{NMCBSPACE}^{\star}_{\mathcal{D}}(f)$ *for each* $f\colon \mathbb{N} \to \mathbb{N}$, *and in particular*

$$\mathbf{PMCBSPACE}^{\star}_{\mathcal{D}} \subseteq \mathbf{NPMCBSPACE}^{\star}_{\mathcal{D}}$$
$$\mathbf{EXPMCBSPACE}^{\star}_{\mathcal{D}} \subseteq \mathbf{NEXPMCBSPACE}^{\star}_{\mathcal{D}}.$$

The results describing closure properties and providing an upper bound for time requirements of P systems operating in bounded binary space are still valid, too:

Proposition 3 *The complexity classes* $\mathbf{PMCBSPACE}^{\star}_{\mathcal{D}}$, $\mathbf{NPMCBSPACE}^{\star}_{\mathcal{D}}$, $\mathbf{EXPMCBSPACE}^{\star}_{\mathcal{D}}$, *and* $\mathbf{NEXPMCBSPACE}^{\star}_{\mathcal{D}}$ *are all closed under polynomial-time reductions.*

Proof. Consider a language $L \in \mathbf{PMCBSPACE}^{\star}_{\mathcal{D}}$ and let M be the Turing machine constructing the family $\mathbf{\Pi}$ that decides L. Let L' be reducible to L via a polynomial-time computable function f.

We can build a Turing machine M' working as follows: on input x of length n, M' computes $f(x)$; then it behaves like M on input $f(x)$, thus constructing $\Pi_{f(x)}$ (we stress the fact that, for the corresponding result concerning the uniform case, the construction of the P system involves two Turing machines, both operating in polynomial time; in this case, we simulate the composition of the two machines). Since $|f(x)|$ is bounded by a polynomial, M' operates in polynomial time and $\Pi_{f(x)}$ in polynomial binary space; it follows that $\mathbf{\Pi}' = \{\Pi_{f(x)} \mid x \in \Sigma^{\star}\}$ is a polynomially semi-uniform family of P systems deciding L' in polynomial binary space. Thus $L' \in \mathbf{PMCBSPACE}^{\star}_{\mathcal{D}}$.

The proof for the three other classes is analogous.

Proposition 4 $\mathbf{MCBSPACE}^{\star}_{\mathcal{D}}(f)$ *is closed under complement for each function* $f\colon \mathbb{N} \to \mathbb{N}$.

Proof. By reversing the roles of objects *yes* and *no*, the complement of a language can be decided.

4 Comparison with standard computational complexity classes

In this section we compare the standard computational complexity classes with the complexity classes defined in the framework of P systems working in binary space.

Most results can be obtained as an immediate consequence of the results given in [8], simply considering that $\mathbf{MCSPACE}_{\mathcal{D}}(f(n)) \subseteq \mathbf{MCBSPACE}_{\mathcal{D}}(f(n))$.

Thus, recalling various results from [8], we have:

Proposition 5 *Let us denote by* $\mathcal{EAM}$ *and* $\mathcal{AM}^0$ *the classes of P systems with active membranes using only elementary membrane division and without polarizations, respectively. The following results hold (we denote a result that holds for both semi-uniform and uniform systems by* $[*]$*):*

$$\mathbf{NP} \cup \mathbf{coNP} \subseteq \mathbf{EXPMCSPACE}^{\star}_{\mathcal{EAM}} \subseteq \mathbf{EXPMCBSPACE}^{\star}_{\mathcal{EAM}}$$
$$\mathbf{PSPACE} \subseteq \mathbf{EXPMCSPACE}^{\star}_{\mathcal{AM}} \subseteq \mathbf{EXPMCBSPACE}^{\star}_{\mathcal{AM}}$$
$$\mathbf{PSPACE} \subseteq \mathbf{EXPMCSPACE}_{\mathcal{AM}} \subseteq \mathbf{EXPMCBSPACE}^{\star}_{\mathcal{AM}}$$
$$\mathbf{PSPACE} \subseteq \mathbf{EXPMCSPACE}^{[*]}_{\mathcal{AM}^0} \subseteq \mathbf{EXPMCBSPACE}^{[*]}_{\mathcal{AM}^0}$$

An interesting research topic concerns the classes for which the inclusion $\mathbf{MCSPACE}_{\mathcal{D}}(f(n)) \subseteq \mathbf{MCBSPACE}_{\mathcal{D}}(f(n))$ is proper and, considering the above inclusions, whether or not the same results can be obtained with stricter binary space classes, by exploiting the improved information storage related to objects with respect to the standard space definition.

Some partial results in this respect are the following:

Theorem 6 *Let us denote by* $\mathcal{NAM}$ *the class of P systems with active membranes that do not use membrane division. The following result holds:* $\mathbf{P} = \mathbf{MCSPACE}^{\star}_{\mathcal{NAM}}(O(1)) = \mathbf{MCBSPACE}^{\star}_{\mathcal{NAM}}(O(1))$

Proof. The inclusion $\mathbf{P} \subseteq \mathbf{MCSPACE}^{\star}_{\mathcal{NAM}}(O(1))$ follows immediately from the definition of semiuniform P systems. Consider a language L in $\mathbf{P}$ and a string x; a deterministic Turing machine can create in polynomial time a P system having a single membrane and one single object *yes* or *no*, directly answering the question whether or not $x \in L$. The inclusion $\mathbf{MCSPACE}^{\star}_{\mathcal{NAM}}(O(1)) \subseteq \mathbf{MCBSPACE}^{\star}_{\mathcal{NAM}}(O(1))$ follows, as stated above, from the definition of binary space.

For the converse, we simply need to recall that a confluent P system without membrane division can be simulated, in polynomial time, by a deterministic Turing machine, like it was shown in [11]. It is easy to see that the proof works both considering the standard space definition as well as the binary space definition for P systems.

Another interesting result concerning the standard definition of space in the framework of P systems was presented in [9], and it focuses on the type of resources used. In particular, a solution for the **PSPACE**-complete problem QUANTIFIED 3SAT was given, for uniform systems using only communication rules (hence no evolution, membrane division and dissolution rules were used), thus proving the inclusion of **PSPACE** in this class. Once again, since the definition of binary space allows a more efficient allocation of space, the result is still valid:

Proposition 7 *Let* $\mathcal{AM}(-\text{ev}, +\text{com}, -\text{dis}, -\text{div})$ *be the class of P systems with active membranes using only communication rules (no evolution, dissolution, nor division of membranes). Then* $\mathbf{PSPACE} \subseteq \mathbf{PMCBSPACE}_{\mathcal{AM}(-\text{ev},+\text{com},-\text{dis},-\text{div})} \subseteq \mathbf{PMCBSPACE}^{*}_{\mathcal{AM}(-\text{ev},+\text{com},-\text{dis},-\text{div})}$.

Once again, it would be interesting to understand whether or not the result remains valid for a smaller binary space class. In this case, the question can be answered negatively, by considering a result presented in [10]. In the article, it was shown that recognizer P systems with active membranes using polynomial space characterize the complexity class **PSPACE**. The result holds for both confluent and nonconfluent systems, and even in the case that non-elementary division is used. In particular, it was pointed out that such systems can be simulated by polynomial space Turing machines.

By considering the alternative definition for binary space, we can thus obtain the corresponding theorem:

Theorem 8 *Let* Π *be a nonconfluent P system with active membranes, running in binary space* S*. Then, it can be simulated by a deterministic Turing machine in space* $O(S)$*.*

Proof. We simulate Π by a non-deterministic Turing machine N, which can then be reduced to polynomial deterministic space by using Savitch's theorem [5].

The current configuration of Π can be stored explicitly by N: the membrane structure is represented as a rooted tree, where each node is a membrane and contains the information concerning its label, its charge, the multiset of objects in the region, and a list of children nodes (i.e. the membranes immediately inside it). To represent the multiset of objects inside each region, tuples of integers encoded in binary can be used, with one entry for each object type in the alphabet.

Since the simulation algorithm is the same as in [10], it is still valid that the space required to store further information needed to carry on the simulation is limited by S.

It follows that the total requested amount of space for the simulation is of the same order as the one required by Π, that is, $O(S)$.

It follows immediately from this theorem and from Proposition 7:

Theorem 9 *Let $\mathcal{D}$ be a class of P systems with active membranes using at least communication rules. Then* $[\mathbf{N}]\mathbf{PMCBSPACE}_{\mathcal{D}}^{[\star]} = \mathbf{PSPACE}$*, where* $[\mathbf{N}]$ *denotes optional nonconfluence, and* $[\star]$ *optional semi-uniformity.*

In [2] it was shown that exponential space Turing machines can be simulated by polynomially uniform exponential-space P systems with active membranes. In view of this result and of Theorem 8, and of the definition of binary space, we have the following:

Theorem 10 $\mathbf{EXPSPACE} = \mathbf{EXPMCBSPACE}_{\mathcal{AM}} = \mathbf{EXPMCBSPACE}_{\mathcal{AM}}^{\star} = \mathbf{NEXPMCBSPACE}_{\mathcal{AM}}^{\star}$

Proof. The following inclusions hold by definition:
$\mathbf{EXPMCBSPACE}_{\mathcal{AM}} \subseteq \mathbf{EXPMCBSPACE}_{\mathcal{AM}}^{\star} \subseteq \mathbf{NEXPMCBSPACE}_{\mathcal{AM}}^{\star}$,
whereas it is easy to see that the inclusion $\mathbf{NEXPMCBSPACE}_{\mathcal{AM}}^{\star} \subseteq \mathbf{EXPSPACE}$ is an immediate corollary of theorem 8.

Finally, the inclusion of **EXPSPACE** in $\mathbf{EXPMCSPACE}_{\mathcal{AM}}$ is proved in [2, Theorem 8]. Recalling that $\mathbf{EXPMCSPACE}_{\mathcal{AM}} \subseteq \mathbf{EXPMCBSPACE}_{\mathcal{AM}}$, it follows $\mathbf{EXPSPACE} \subseteq \mathbf{EXPMCBSPACE}_{\mathcal{AM}}$ □

Hence, also in this case, considering binary space instead of the standard one does not result in improved efficiency. Moreover, when we consider an exponential amount of space, we can show that the classes coincide: in fact, considering the theorem just proved and recalling [1, Corollary 1] proving the same results for classes with the original definition of space for P systems, we have

Corollary 11 $\mathbf{EXPSPACE} = \mathbf{EXPMCSPACE}_{\mathcal{AM}} = \mathbf{EXPMCSPACE}_{\mathcal{AM}}^{\star} = \mathbf{NEXPMCSPACE}_{\mathcal{AM}}^{\star} = \mathbf{EXPMCBSPACE}_{\mathcal{AM}} = \mathbf{EXPMCBSPACE}_{\mathcal{AM}}^{\star} = \mathbf{NEXPMCBSPACE}_{\mathcal{AM}}^{\star}$

5 Conclusions

We have proposed an alternative space complexity measure for P systems with active membranes, where the multiplicity of each object in each membrane is stored by using binary numbers. We have defined the corresponding complexity classes and we have compared some of them both with standard space complexity classes and with complexity classes defined in the framework of P systems considering the original definition of space ([8]).

An interesting research topic is to compare such classes for different amounts of allowed space. In particular, it would be interesting to find specific classes defined in terms of binary space which stricly contain classes defined in terms of standard space in the framework of P systems, thus proving that storing in an efficient way the information concerning objects can really be exploited. We showed that, when

an exponential amount of space is considered, these classes do not differ. Only partial answers have been obtained for a polynomial amount of space. We expect that the differences can be evident when considering sublinear space complexity classes.

References

1. Alhazov, A., Leporati, A., Mauri, G., Porreca, A.E., Zandron, C.: The computational power of exponential-space P systems with active membranes. In: Martínez-del-Amor, M.A., Păun, G., Pérez-Hurtado, I., Romero-Campero, F.J. (eds.) Tenth Brainstorming Week on Membrane Computing, Volume I. pp. 35–60. No. 1/2012 in RGNC Reports, Fénix Editora (2012), http://www.gcn.us.es/icdmc2012_proceedings
2. Alhazov, A., Leporati, A., Mauri, G., Porreca, A.E., Zandron, C.: Space complexity equivalence of P systems with active membranes and Turing machines. Theoretical Computer Science **529**, 69–81 (2014), https://doi.org/10.1016/j.tcs.2013.11.015
3. Leporati, A., Mauri, G., Porreca, A.E., Zandron, C.: A gap in the space hierarchy of P systems with active membranes. Journal of Automata, Languages and Combinatorics **19**(1–4), 173–184 (2014), http://theo.cs.ovgu.de/jalc/search/j19_i.html
4. Murphy, N., Woods, D.: The computational power of membrane systems under tight uniformity conditions. Natural Computing **10**(1), 613–632 (2011), https://doi.org/10.1007/s11047-010-9244-7
5. Papadimitriou, C.H.: Computational Complexity. Addison-Wesley (1993)
6. Păun, G.: P systems with active membranes: Attacking NP-complete problems. Journal of Automata, Languages and Combinatorics **6**(1), 75–90 (2001)
7. Păun, G., Rozenberg, G., Salomaa, A. (eds.): The Oxford Handbook of Membrane Computing. Oxford University Press (2010)
8. Porreca, A.E., Leporati, A., Mauri, G., Zandron, C.: Introducing a space complexity measure for P systems. International Journal of Computers, Communications & Control **4**(3), 301–310 (2009), http://univagora.ro/jour/index.php/ijccc/article/view/2779
9. Porreca, A.E., Leporati, A., Mauri, G., Zandron, C.: P systems with active membranes: Trading time for space. Natural Computing **10**(1), 167–182 (2011), https://doi.org/10.1007/s11047-010-9189-x
10. Porreca, A.E., Leporati, A., Mauri, G., Zandron, C.: P systems with active membranes working in polynomial space. International Journal of Foundations of Computer Science **22**(1), 65–73 (2011), https://doi.org/10.1142/S0129054111007836
11. Zandron, C., Ferretti, C., Mauri, G.: Solving NP-complete problems using P systems with active membranes. In: Antoniou, I., Calude, C.S., Dinneen, M.J. (eds.) Unconventional Models of Computation, UMC'2K, Proceedings of the Second International Conference, pp. 289–301. Springer (2001), https://doi.org/10.1007/978-1-4471-0313-4_21