Computational Complexity Aspects in Membrane Computing

Claudio Zandron

Dipartimento di Informatica, Sistemistica e Comunicazione
Università degli Studi di Milano-Bicocca

16th International Conference on Membrane Computing
17-21 August 2015, Valencia, Spain
Summary

- Membrane Systems with Active Membranes: definitions
- Time-efficient solutions for computationally hard problems
- Time Complexity Classes for Membrane systems
- Space Complexity Classes for Membrane systems
P Systems with Active Membranes
Definition: \( \mathcal{P} \) systems with Active Membranes

\[
\Pi = (V, H, \mu, M_1, \ldots, M_n, R)
\]

- \( V \): Alphabet
- \( H \): set of labels for membranes
- \( \mu \): Membrane structure (Ex. \([ [ ]_2 [ ]_3 [ [ ]_5 [ ]_6 ]_4 ]_1\))
- \( M_i \): String over \( V \), initial multiset of symbols in region \( i \)
- \( R \): Finite sets of evolution rules
- Membranes and objects can be marked using polarization: \( \{+, -, 0\} \)
Developmental Rules

- Assume $a \in V, w \in V^*, h \in H, \alpha_i \in \{+,-,0\}$
- **Object evolution:** $[a \rightarrow w]^\alpha_h$
- **IN communication:** $a [ ]^\alpha_h \rightarrow [b]^\alpha_h$
- **OUT communication:** $[a]^\alpha_h \rightarrow [ ]^\alpha_b$
- **Dissolution:** $[a]^\alpha_h \rightarrow b$
### Division Rules

- **Elementary division:**
  \[ [a]^{\alpha_1}_h \rightarrow [b]^{\alpha_2}_h [c]^{\alpha_3}_h \]

- **Non-elementary division:**
  \[
  \left[ [ ]^{+}_{h_1} \cdots [ ]^{+}_{h_k} [ ]^{-}_{h_{k+1}} \cdots [ ]^{-}_{h_n} \right]^{\alpha_1}_h \rightarrow \\
  \left[ [ ]^{+}_{h_1} \cdots [ ]^{+}_{h_k} \right]^{\alpha_2}_h \\
  \left[ [ ]^{-}_{h_{k+1}} \cdots [ ]^{-}_{h_n} \right]^{\alpha_3}_h
  \]

- **Non-elementary division:** Membranes with neutral polarization are duplicated
Application of the rules

- Maximal parallel semantics
- At each step, a membrane can be the subject of only one rule
- When a membrane divides, its contents is replicated unchanged in the new copy
- OUTPUT: Symbols that exit from the skin in a halting computation
Solving SAT by P Systems with Active Membranes
SAT Problem

- SAT - Satisfiability for boolean formulas: a boolean formula $\Phi$ in CNF, with
  - $n$ boolean variables $x_1, x_2, \ldots x_n$
  - $m$ clauses
- Question: is there a truth assignment for $x_1, x_2, \ldots x_n$ such that $\Phi$ is true?
- Brute force algorithm requires exponential time
- SAT is NP–complete
Solving SAT in linear time

\[ [[z_1 \ a_1 a_2 \ldots a_n]_{21}^{010} \]
Solving SAT in linear time

\[ [[z_1 \ a_1 \ a_2 \ \ldots \ a_n]_2^0]_1^0 \]
\[ [[z_2 \ T_1 \ a_2 \ \ldots \ a_n]_2^0 \ [z_2 \ F_1 \ a_2 \ \ldots \ a_n]_2^0]_1^0 \]
Solving SAT in linear time

- $[[z_1 \ a_1 a_2 \ldots a_n]_2^0]_1^0$
- $[[z_2 \ T_1 a_2 \ldots a_n]_2^0 \ [z_2 \ F_1 a_2 \ldots a_n]_2^0]_1^0$
- $[[z_3 \ T_1 T_2 a_3 \ldots a_n]_2^0 \ [z_3 \ T_1 F_2 a_3 \ldots a_n]_2^0 \ [z_3 \ F_1 T_2 a_3 \ldots a_n]_2^0 \ [z_3 \ F_1 F_2 a_3 \ldots a_n]_2^0]_1^0$
Solving SAT in linear time

\[
\begin{align*}
&> [[z_1 \ a_1 a_2 \ldots a_n]_2^0]_1^0 \\
&> [[z_2 \ T_1 a_2 \ldots a_n]_2^0 \ [z_2 \ F_1 a_2 \ldots a_n]_2^0]_1^0 \\
&> [[z_3 \ T_1 T_2 a_3 \ldots a_n]_2^0 \ [z_3 \ T_1 F_2 a_3 \ldots a_n]_2^0]_1^0 \\
&> [z_3 \ F_1 T_2 a_3 \ldots a_n]_2^0 \ [z_3 \ F_1 F_2 a_3 \ldots a_n]_2^0]_1^0 \\
&> \ldots \\
&> [[z_n \ T_1 T_2 \ldots T_n]_2^0 \ [z_n \ T_1 T_2 \ldots T_{n-1} F_n]_2^0]_1^0 \\
&> \ldots \ [z_n \ F_1 F_2 \ldots F_n]_2^0]_1^0 \\
&> \text{In } n \text{ steps we generate all possible truth assignments}
\end{align*}
\]
- In one step we change the polarization of the membranes using $z_n$
- $[[T_1 T_2 \cdots T_n]_2^+ [T_1 T_2 \cdots T_{n-1} F_n]_2^+ \cdots [F_1 F_2 \cdots F_n]_2^+]_1^0$
Solving SAT in linear time

- In one step we change the polarization of the membranes using $z_n$
- $[[T_1 \ T_2 \ \ldots \ T_n]_2^+ \ [T_1 \ T_2 \ \ldots \ T_{n-1} F_n]_2^+$
  $\ldots \ [F_1 F_2 \ \ldots \ F_n]_2^+ ]_1^0$
- In one step every symbol $T_i$ (resp. $F_i$) is replaced by some symbols $R_{h_i}$
- $1 \leq h_i \leq m$ is the index of a clause satisfied by setting $x_i = \text{TRUE}$ (resp. $x_i = \text{FALSE}$)
- We obtain, for example,
  $[[R_1 R_3 R_1 R_4 \ \ldots \ R_6]_2^+ \ [R_7 R_3 R_2 R_3 \ \ldots \ R_2]_2^+$
  $\ldots \ [R_2 R_5 R_1 R_5 \ \ldots \ R_1]_2^+ ]_1^0$
Solving SAT in linear time

- In $2m$ steps we check whether or not a membrane contains all $R_j$, where $1 \leq j \leq m$
- $[[\ldots]_2^-\quad [[\ldots]_2^+]_2^+\quad \ldots\quad [[\ldots]_2^- T \quad T]_1^0$
Solving SAT in linear time

- In $2m$ steps we check whether or not a membrane contains all $R_j$, where $1 \leq j \leq m$
- $[[\ldots]_2^- [\ldots]_2^+ \ldots [\ldots]_2^- T \ T]_1^0$
- After $n + m + 2$ steps, eventually a symbol $T$ is sent out through the skin membrane
- $[[\ldots]_2^- [\ldots]_2^+ \ldots [\ldots]_2^- T]_1^- T$
- If after exactly $n + 2m + 2$ computation steps we obtain a $T$ in the environment, then the answer is YES; otherwise the answer is NO
Features of the Solution

- The solution requires linear time and exponential space
- The solution proposed is said to be **Semiuniform**:
  - Every input instance requires a specific membrane system to be computed
  - Given an input instance $x$ of length $n$, the membrane system used to solve it can be generated by a deterministic Turing machine in polynomial time w.r.t. $n$
- The proposed solution is easily modified to send out a symbol "yes" at step $n + m + 2$ or the symbol "no" at step $n + m + 3
Recognizer P system
Recognizer P system

- DEFINITION - **Recognizer P system**: a P system $\Pi$ whose alphabet contains two distinct symbols *yes* and *no*.
- Every halting computation of $\Pi$ sends out from the skin membrane one object between *yes* and *no*
- If the object *yes* is emitted, then we have an *accepting* computation
- If the object *no* is emitted, then we have a *rejecting* computation
Recognizer $P$ system

Language accepted

- Consider an alphabet $\Sigma$ and a language $L \subseteq \Sigma^*$.
- $\Pi$ : family of recognizer $P$ systems, s.t. each string $x \in \Sigma^*$ is associated with a member $\Pi_x \in \Pi$.
- $\Pi$ decides $L$ IFF, for all strings $x \in \Sigma^*$ :
  - $\Pi_x$ accepts if $x \in L$
  - $\Pi_x$ rejects if $x \notin L$
Determinism vs Non-determinism

- A (recognizer) P system $\Pi$ is said to be **deterministic** if there is at most one possible transition from a configuration to the following one, for all possible configurations.

- A **non-deterministic** (recognizer) P system $\Pi$ is said to be **confluent** if the computations of $P$ are either all accepting or all rejecting. Such a P system *accepts* in the former case and *rejects* in the latter.

- When not all computations necessarily agree on the result, the P system is called **non-confluent**. Non-confluent P systems are said to *accept* when there exists an accepting computation, and to *reject* otherwise.
Complexity classes for **confluent** P systems

\[ MC_T(f) = \{ L \subseteq \Sigma^* \mid L \text{ is decided by a semi-uniform family of confluent P systems} \}, \text{ such that} \]

- \( T \) is a class of recognizer P systems
- \( f : \mathbb{N} \rightarrow \mathbb{N} \) be a proper complexity function
- for each \( x \in \Sigma^* \), every computation of \( \Pi_x \) halts within \( f(|x|) \) steps

**PMC_T** : languages decided **IN POLYNOMIAL TIME** by confluent recognizer P systems in the class \( T \)
Complexity classes for non–confluent P systems

\[ NMC_T(f) = \{ L \subseteq \Sigma^* | L \text{ is decided by a semi-uniform family of non–confluent P systems} \} \], such that

- \( T \) is a class of recognizer P systems
- \( f : N \to N \) be a proper complexity function
- for each \( x \in \Sigma^* \), every computation of \( \Pi_x \) halts within \( f(|x|) \) steps

\( NPMC_T \) : languages decided **IN POLYNOMIAL TIME** by non–confluent recognizer P systems in the class \( T \)
Standard classes of recogniser P systems

- $\mathcal{AM}$: recognizer P systems with active membranes. Such recognizers can use both division for elementary and non–elementary membranes
- $\mathcal{EAM}$: recognizer P systems with active membranes, which can only use division for elementary membranes
- $\mathcal{NAM}$: recognizer P systems with active membranes, but which cannot use any kind of division
Time Complexity Classes
Basic properties

- $MC_T(f) \subseteq NMC_T(f)$; in particular, $PMC_T \subseteq NPMC_T$
- $MC_{\mathcal{N}AM}(f) \subseteq MC_{\mathcal{E}AM}(f) \subseteq MC_{AM}(f)$
- $PMC_{\mathcal{N}AM} \subseteq PMC_{\mathcal{E}AM} \subseteq PMC_{AM}$
- $NPMC_{\mathcal{N}AM} \subseteq NPMC_{\mathcal{E}AM} \subseteq NPMC_{AM}$
CONFLUENT P systems without division rules

- \( P \subseteq PMC_{N^{AM}} \)
  - "Trick": the DTM deciding \( L \in P \) is used to solve **DIRECTLY** the problem in polynomial time
  - Then, we build a P system with a single membrane containing either an object \( YES \), whenever an input \( x \in L \) is given, or \( NO \), otherwise. This requires polynomial time.
  - The P system send out the object in a single step
CONFLUENT P systems without division rules

The opposite is also true:

- \( PMC_{NAM} \subseteq P \)
  - Idea: simulation of a generic P system \( \Pi \) without membrane division using a DTM \( M \), with a polynomial slowdown
  - We keep track of the NUMBER OF OCCURRENCES of each symbol in each membrane
  - The application of a rule in \( \Pi \) can be simulated by modifying the counters used in \( M \)
Simulating $\Pi \in PMC_{NAM}$: examples

- $[a \rightarrow bcd]_i$ (object evolution rule)
  - $\# b_i := \# b_i + \# a_i$
  - $\# c_i := \# c_i + \# a_i$
  - $\# d_i := \# d_i + \# a_i$
  - $\# a_i := 0$

- $[ [a]_i ]_k \rightarrow [ b ]_k$ (dissolution rule)
  - $\# a_i := \# a_i - 1$
  - $\# b_k := \# b_k + 1$
  - $\forall x \in \Sigma : \# x_k := \# x_k + \# x_i$
Simulating \( \Pi \in PMC_{NAM} \): time required

\[ A = \# \text{ of membranes in } \Pi \]
\[ B = \text{ number of symbols of the alphabet used by } \Pi \]
\[ C = \text{ max size of the rules in } \Pi \]
\[ D = \max \{C, B + 2\} \]

- Total time needed to simulate \( t \) computation steps of \( \Pi \):
  \[ O(t \times A \times B \times D \times \log(A \times B \times C^t)) = O(t^2) \]

As a consequence: \( P = PMC_{NAM} \)
NON CONFLUENT systems without division rules

- Is the similar non–deterministic version of the previous result also true? \( NP = NPMC_{NAM} \)?
- Previously we showed \( P \subseteq PMC_{NAM} \) simply by consider a DTM that solves the input problem itself.
- The constructed P system only output the correct result.
- This construction is not valid anymore for the non–confluent version.
- Reason: the Turing machine constructing a family of non-confluent P systems must still be deterministic!
- It seems unable to solve an NP problem by itself in polynomial time!
NON CONFLUENT systems without division rules

However, the inclusion $NP \subseteq NPMC_{\mathcal{N}AM}$ holds. Idea:

- Consider a non–confluent P systems solving SAT (NP–complete)
- General rules of the form $[x]; \rightarrow [ ]; 0$ and $[x]; \rightarrow [ ]; 1$
- A truth assignments is non–deterministically generated and then checked in polynomial time (as in the previous construction).
- Such a non–confluent P system can be constructed in polynomial time (w.r.t. the input length) by a DETERMINISTIC TM
NON CONFLUENT systems without division rules

\[ NPMC_{\mathcal{N}AM} \subseteq NP \] also holds

- Idea: simulate a non–confluent P systems without division by means of a non–deterministic Turing machine
- Construction similar to the previous case, for confluent system

Thus, \[ NP = NPMC_{\mathcal{N}AM} \]
CONFLUENT systems with elementary division rules only

- We have already seen that SAT is solvable by a family of recognizer P systems that make use only of elementary membrane division.

- SAT is NP-Complete. Then, it follows: \( NP \subseteq PMC_{AM} \)

- Confluent P systems with elementary membrane division can be simulated by Deterministic Turing machines using polynomial space: \( PMC_{AM} \subseteq PSPACE \)

- Hence: \( NP \subseteq PMC_{AM} \subseteq PSPACE \)
Complexity class $PP$ (Probabilistic P): PP is the class of decision problems solvable by a probabilistic Turing machine in polynomial time, with an error probability of less than 1/2 for all instances.

(Alternatively) $PP$: languages $L \subseteq \Sigma^*$ decided by a NDTM $N$ in polynomial time in the following way: $N$ accepts $x \in \Sigma^*$ IFF more than half of the computations on input $x$ are accepting.

$NP \cup coNP \subseteq PP \subseteq PSPACE$
Elementary CONFLUENT systems: SQRT-3SAT

- SQRT-3SAT: given a Boolean formula of \( m \) variables in 3CNF, determine whether the number of truth assignments satisfying it is at least \( \sqrt{2^m} \).
- SQRT-3SAT is \( \mathbf{PP} \)-complete
- SQRT-3SAT can also be solved by CONFLUENT systems with elementary division rules only. Hence: \( PP \subseteq PMC_{EAM} \)
- \( NP \cup coNP \subseteq PP \subseteq PMC_{EAM} \subseteq PSPACE \)
CONFLUENT systems with both types of division rules

- $\text{PMC}_{AM} \subseteq \text{PSPACE}$: Confluent P systems with division for both elementary and non–elementary membranes can be simulated by Deterministic Turing machines using polinomial space.

- Quantified satisfiability (QSAT) - SAT using quantifiers: consider a Boolean expression $\Phi$ in CNF. Question: $\exists x_1 \forall x_2 \exists x_3 \forall x_4 \ldots Q_n x_n \Phi$?
  - QSAT is PSPACE–complete
  - QSAT can be solved in polynomial time by P confluent P systems using division rules, i.e. QSAT $\in \text{PMC}_{AM}$
  - $\text{PSPACE} \subseteq \text{PMC}_{AM}$
CONFLUENT systems with both types of division rules

- From the previous results, it follows that $PSPACE = PMC_{AM}$
- We can also consider weaker systems, removing various features
- E.G. - What if we remove dissolving action and polarizations?
- Surprising result: $P = PMC_{AM}(n\delta, nPol)$!!!
NON CONFLUENT systems with division rules

- $NPMC_{AM} \subseteq NEXP \subseteq EXPSPACE$
- $PSPACE \subseteq PMC_{AM} \subseteq NPMC_{AM}$
Research directions

- Characterize complexity classes defined on the basis of various features considered
- Uniform vs Semi-uniform systems
- (as said) $P = PMC_{AM}(\neg \text{dissolution}, \neg \text{polarization})$
- $NP \subseteq PMC_{AM}(\neg \text{polarization})$
- $NP \subseteq PMC_{AM}(\text{min})$, where $\text{min}$ denotes the use of minimal parallelism semantics
- ...

-
Space Complexity
Space in membrane systems: definitions

- Idea: both objects and membrane need physical space
- Let $C_i$ be a configuration of a P system $\Pi$
- $\text{size}(C_i)$ is the sum of number of membranes in $\mu$ and the total number of objects they contain
- The space required by a halting computation $C = (C_0, C_1, \ldots, C_m)$ of $\Pi$ is
  $\text{size}(C) = \max\{\text{size}(C_0), \ldots, \text{size}(C_m)\}$
- The space required by $\Pi$ itself is
  $\text{size}(\Pi) =\max\{\text{size}(C) : C \text{ is a halting computation of } \Pi\}$
Space complexity classes

- $MCSPACE_T^*[f]$: languages decided by [semi]uniform confluent recognizer P systems (of type T) within space $f(n)$
- $NMCSPACE_T^*[f]$: languages decided by [semi]uniform non–confluent recognizer P systems (of type T) within space $f(n)$
- $PMCS\frac{PSPACE}_T^*[f] = MCSPACE_T^*[p(n)]$
- $NPMCS\frac{SPACE}_T^*[f] = NMCSPACE_T^*[p(n)]$
Some basic result concerning space complexity classes

- \( P \subseteq MCSPACE_{NAM}(O(1)) \)
- \( NP \cup co-NP \subseteq EXPMCSPACE_{EAM} \)
- \( PSPACE \subseteq EXPMCSPACE_{AM} \)
PSPACE upper bound

- **Theorem:** consider nonconfluent P systems with active membranes (and both types of division), running in space $S$ and having a description of length $m$

- Such systems can be simulated by a NDTM in space $O(S \log m)$
PSPACE upper bound

1. For each rule in R, assign to it a nondeterministically chosen set of membranes and objects to which the rule should be applied
2. Check if the assignment of membranes and objects to rules is indeed maximally parallel; if not, reject
3. Apply the rules selected in 1, from the elementary membranes and going up towards the skin membrane
4. Repeat from 1. until yes or no is sent out from the skin membrane in step 3
5. Halt and accept or reject accordingly
PSPACE upper bound

- How much space is required from the previous simulation?
- We only need to store the current configuration and some auxiliary data (assigned rules, assigned objects, etc).
- Space required is $O(S \log m)$
- As $PSPACE = NPSPACE$ we have:
  \[ PMCSPACE^{[*]}_T \subseteq NPMCSPACE^{[*]}_T \subseteq PSPACE \]
PSPACE lower bound

- Can we also find a lower bound for simulating P systems with active membranes?
- Consider \( Q3SAT \), a \( PSPACE \) complete problem
- Quantified satisfiability (\( Q3SAT \)) - 3-SAT using quantifiers: consider a Boolean expression \( \Phi \) in CNF. Question: \( \exists x_1 \forall x_2 \exists x_3 \forall x_4 \ldots Q_n x_n \Phi \)?
- It is possible to solve \( Q3SAT \) using a uniform family of P systems working in (confluent) polynomial space (using only communication, NO EVOLUTION)
PSPACE lower bound

- Design a uniform family of simple programs solving $Q3SAT$ in polynomial space
- Compile the programs into a uniform family of register machines working in polynomial space
- Simulate the register machine with a uniform family of P systems working in polynomial space
- **Corollary:** $PSPACE \subseteq PMCSPACE_{AM}(-evo; -diss)$
Space in membrane systems: PSPACE

- From the previous results we have:
  - $PSPACE \subseteq PMCSpace_{Am}(-evo; -diss)$: deterministic uniform membrane systems with only division and communication rules contain $PSPACE$
  - $NPMCSpace_{Am}^* \subseteq PSPACE$: nondeterministic semiuniform membrane systems with division and all types of rules contain $PSPACE$

- Thus, ALL TYPES OF MEMBRANE SYSTEMS with active membrane (both types of division) and at least communication rules WORKING IN POLYNOMIAL SPACE characterize $PSPACE$
Sublinear Space Membrane Systems
Sublinear Space Membrane Systems

We can also consider sublinear space; to do so, we need to add some further conditions:

- Two distinct alphabets: \textit{INPUT} alphabet and \textit{WORK} alphabet
- Input objects cannot be rewritten and do not contribute to the size of a configuration
- Size of a configuration: number of membranes + total number of working objects
- Weaker uniformity condition: \textit{DLOGTIME}-uniformity
  \textit{(DLOGTIME} Turing machines)
Power of Sublinear Space Membrane Systems

- Idea: compare with logarithmic space Turing machines (or other equivalent models)
- Two problems if we use "standard" techniques:
  - Need for a polynomial number of working objects (violates log-space condition)
  - Need for a polynomial number of rewriting rules (violates uniformity condition)
- Solution: use polarization both to communicate objects and store information
Power of Sublinear Space Membrane Systems

- Each log-space DTM $M$ can be simulated by a $DLOGTIME$-uniform family $\Pi$ of Membrane systems with active membranes in logarithmic space.
- The bits corresponding to the actual position of the INPUT head of $M$ are stored in the polarizations of the membranes.
- Only ONE object among all INPUT objects can reach the skin.
- That object corresponds to the INPUT symbol actually read by the tape head of $M$.
- We identify the symbol under the WORK tape of $M$.
- The transition of $M$ can be simulated.
Power of Sublinear Space Membrane Systems

- Only a logarithmic number of objects and membranes are required (besides the input objects)
- The family \(\Pi\) is \(DLOGTIME\)-uniform
- Thus: \(L\) is contained in the class of problems solved by \(DLOGTIME\)-uniform, log–space Membrane systems with active membranes
Can the result be improved?

- Log-space Turing machines can only generate polynomially many configurations.
- Log-space P systems have *exponentially* many configurations: $n$ distinct input objects can be partitioned in $\log n$ different regions.
- ANSWER: Yes! (Confluent) log-space P systems can simulate polynomial-space Turing machines.
A GAP in the space hierarchy

- Input symbols’ positions are used to store symbols of the simulated TM
- Querying the symbol under the tape head of the TM is done by means of electrical charges
- A GAP in the space hierarchy: First case where the space complexity of P systems and that of Turing machines differ by an exponential amount!
Constant-space P systems
Constant space P systems

Consider P systems using constant number of working objects and constant number of membranes

Surprising result: a constant amount of space is sufficient (and trivially necessary) to solve all problems in \( PSPACE \)!!!
Simulating a polynomial space TM

- Input objects are used to store the contents of the tape of $M$.
- Only a constant number of input symbols can be rewritten (otherwise, the amount of space used would be non-constant).
- Hence, the subscripts and the positions of the input objects are exploited.
- $\sigma_j$ in a membrane $A$ means that the $j$-th cell of the tape contains the symbol $A$.
- The subscript and position of an input object $\sigma_j$ can be read using only a constant number of additional objects and membranes.
Constant space $\mathbb{P}$ systems

- The simulation of one step of $M$ requires a polynomial number of steps
- Only a constant number of membranes and working objects are present at each computation step
- Simulating systems can be constructed by log-space Turing machines

**Theorem**

Let $M$ be a single-tape deterministic Turing machine working in polynomial space $s(n)$. Then, there exists an $L$–uniform family $\Pi$ of $\mathbb{P}$ systems with active membranes that simulates $M$ in space $O(1)$. 
Rethinking the definition of space
New definition of space

- Each non-input object and membrane label encodes $\Theta(\log(n))$ bits of information. E.G.
  - $q_i$ have a subscript $i$ ranging from 1 to $n$
  - Requires unitary space (definition of space)
  - The binary representation of the subscript $i$ requires
    $\log p(n) = \Theta(\log n)$ bits

- We need an alternative definition to capture in a more accurate way the intuition of space

Definition
Let $C$ be a configuration of a P system $\Pi$, $\Lambda$ the set of labels of its membranes, and $\Gamma$ its non-input alphabet.
The size $|C|$ of $C$ is defined as the number of membranes in the current membrane structure multiplied by $\log |\Lambda|$, plus the total number of objects from $\Gamma$ multiplied by $\log |\Gamma|$.
New definition of space

Adopting this stricter definition:

- Does not significantly change space complexity results involving polynomial or larger upper bounds
- The simulation previously using constant-space P systems would require now logarithmic space
- The simulation previously using logarithmic-space P systems would require now $\Theta(\log n \log \log n)$
Research directions

- Characterize SPACE complexity classes defined on the basis of various features considered
- Uniform vs Semi–uniform systems
- ...

66/67
THANK YOU !!!