Counting P systems

David Orellana-Martín

Research Group on Natural Computing
Dept. of Computer Science and Artificial Intelligence
Universidad de Sevilla

Seville, Spain, January 31, 2017

Introduction

- Membrane systems have been used to solve a large number of problems.

Introduction

- Membrane systems have been used to solve a large number of problems.
- Depending on their own essence, we can talk about "**easier**" or "**harder**" problems.

Introduction

- Membrane systems have been used to solve a large number of problems.
- Depending on their own essence, we can talk about "**easier**" or "**harder**" problems.
- We can switch syntactic (and semantic) ingredients...

Introduction

- Membrane systems have been used to solve a large number of problems.
- Depending on their own essence, we can talk about "**easier**" or "**harder**" problems.
- We can switch syntactic (and semantic) ingredients...
- ... in order to solve different sets of problems.

- Several kinds of membrane systems have been created to solve problems of different natures.

Introduction

- Several kinds of membrane systems have been created to solve problems of different natures.
    - Recognizer P systems

- Several kinds of membrane systems have been created to solve problems of different natures.
  - Recognizer P systems
  - Computing P systems

Introduction

- Several kinds of membrane systems have been created to solve problems of different natures.
  - Recognizer P systems
  - Computing P systems
  - Generating P systems

- Several kinds of membrane systems have been created to solve problems of different natures.
  - Recognizer P systems
  - Computing P systems
  - Generating P systems
  - Function P systems

Complexity classes

- Classic computational complexity classes have been characterized in the framework of Membrane Computing.

[1] L. G. Valiant, The Complexity of Computing the Permanent, *Theoretical Computer Science* 8 (1979), 189–201

Complexity classes

- Classic computational complexity classes have been characterized in the framework of Membrane Computing.
- **P**

[1] L. G. Valiant, The Complexity of Computing the Permanent, *Theoretical Computer Science* 8 (1979), 189–201

Complexity classes

- Classic computational complexity classes have been characterized in the framework of Membrane Computing.
- **P**, **NP**

[1] L. G. Valiant, The Complexity of Computing the Permanent, *Theoretical Computer Science* 8 (1979), 189–201

Complexity classes

- Classic computational complexity classes have been characterized in the framework of Membrane Computing.
- **P**, **NP**, **PP**

[1] L. G. Valiant, The Complexity of Computing the Permanent, *Theoretical Computer Science* 8 (1979), 189–201

Complexity classes

- Classic computational complexity classes have been characterized in the framework of Membrane Computing.
- **P**, **NP**, **PP**, **PSPACE**...
- In **1979**, L. G. Valiant introduced a new complexity class [1].

[1] L. G. Valiant, The Complexity of Computing the Permanent, *Theoretical Computer Science* 8 (1979), 189–201

Complexity classes

- Classic computational complexity classes have been characterized in the framework of Membrane Computing.
- **P**, **NP**, **PP**, **PSPACE**...
- In **1979**, L. G. Valiant introduced a new complexity class [1].
- Here, we have decision problems, but the answer of the system must not be `yes` or `no`...
- ... but the **number of positive computations** we obtain!

[1] L. G. Valiant, The Complexity of Computing the Permanent, *Theoretical Computer Science* 8 (1979), 189–201

Counting Turing machines

- Standard non-deterministic Turing Machine with an auxiliary output device that prints in **binary notation** on a special tape the number of accepting computations induced by the input.

Counting Turing machines

- Standard non-deterministic Turing Machine with an auxiliary output device that prints in **binary notation** on a special tape the number of accepting computations induced by the input.
- A counting Turing machine can solve a special kind of problems: **counting** problems.
- The complexity class of these problems is called $\#\mathbf{P}$.

Counting Turing machines

- Standard non-deterministic Turing Machine with an auxiliary output device that prints in **binary notation** on a special tape the number of accepting computations induced by the input.
- A counting Turing machine can solve a special kind of problems: **counting** problems.
- The complexity class of these problems is called $\#\mathbf{P}$.
- An example of this kind of problems is $\#\texttt{SAT}$.

Counting membrane systems

- A new type of membrane systems is created in order to solve this kind of problems.

Counting membrane systems

- A new type of membrane systems is created in order to solve this kind of problems.
- A **counting membrane system** of degree q is a membrane system

$$\Pi = (\Gamma, \Sigma, C, \mathcal{M}_1, \cdots, \mathcal{M}_1, \mathcal{R}, i_{in}, i_{out})$$

where C is a special alphabet called *output alphabet*, that will represent the answer to the problem in binary representation.

Counting membrane systems

- A new type of membrane systems is created in order to solve this kind of problems.
- A **counting membrane system** of degree q is a membrane system

$$\Pi = (\Gamma, \Sigma, C, \mathcal{M}_1, \cdots, \mathcal{M}_1, \mathcal{R}, i_{in}, i_{out})$$

  where C is a special alphabet called *output alphabet*, that will represent the answer to the problem in binary representation.
- We will denote $\mathbf{PMC}_{\mathcal{C}}$ the class of counting membrane systems.

First results

Theorem

$\#\mathtt{SAT} \subseteq \mathbf{PMC}_{\mathcal{DAM}^0_{\mathcal{C}}(mcmp,+c,-d,-n)}$

- Based on the solution given in [2]...

[2] L. Valencia-Cabrera, D. Orellana-Martín, A. Riscos-Núñez, M. J. Pérez Jiménez. Reaching efficiency through complicity in membrane systems: dissolution, polarization and cooperation, *Theoretical Computer Science*, submitted 2016

First results

- Generation stage

$$\begin{array}{c} [\quad]_1 \\ | \\ [\quad]_2 \end{array}$$

First results

- Generation stage

$$[\quad]_1$$

$$[T_1]_2 \qquad [F_1]_2$$

First results

- Generation stage

$$[\quad]_1$$

$$[T_1\,T_2]_2 \quad [T_1\,F_2]_2 \quad [F_1\,T_2]_2 \quad [F_1\,F_2]_2$$

First results

- Generation stage

$$[\quad]_1$$

$$\underbrace{[T_1 \cdots T_n]_2 \; [T_1 \cdots F_n]_2 \qquad \cdots \qquad [F_1 \cdots T_n]_2 \; [F_1 \cdots F_n]_2}_{2^n \text{ membranes}}$$

First results

- First checking stage

$$[\quad]_1$$

$$\underbrace{[c_1 \cdots c_p]_2 \qquad [c_2 c_4]_2 \qquad \cdots \qquad [c_1 c_2 c_4]_2 \qquad [c_1 \cdots c_p]_2}_{2^n \text{ membranes}}$$

First results

- Second checking stage

$$[\quad]_1$$

$$\underbrace{[d_p]_2 \qquad [c_2 c_4]_2 \qquad \cdots \qquad [d_2 c_4]_2 \qquad [d_p]_2}_{2^n \text{ membranes}}$$

First results

- Second checking stage

$$[d_p^k]_1$$

$$\underbrace{[\quad]_2 \qquad [c_2 c_4]_2 \qquad \cdots \qquad [d_2 c_4]_2 \qquad [\quad]_2}_{2^n \text{ membranes}}$$

$k \equiv$ the number of truth assignments that make true the input formula

First results

- At this point, we would send out:
    - `yes` if at least an object d_p is in membrane 1.
    - `no` if there are no objects d_p in membrane 1.
- We change d_p by α_0.

First results

- Second checking stage

$$[\gamma_0^k]_1$$

$$\underbrace{[\ \ ]_2 \qquad [c_2 c_4]_2 \qquad \cdots \qquad [d_2 c_4]_2 \qquad [\ \ ]_2}_{2^n \text{ membranes}}$$

$k \equiv$ the number of truth assignments that make true the input formula

First results

- And we proceed with the next rules:

$$[\gamma_i \gamma_i \rightarrow \gamma_{i+1}]_1$$

- In n steps we will have in membrane 1:

$$\gamma_k \cdots \gamma_0 \rightarrow \alpha_k \cdots \alpha_0, \alpha_i \in C$$

- We send them out **in polynomial time** with respect of the input.

Corollary

$\#\mathbf{P} \subseteq \mathbf{PMC}_{\mathcal{DAM}^0_{\mathcal{C}}(mcmp,+c,-d,-n)}$

Future work

- If we can solve problems from **NP** in $\mathcal{R}$ then we can solve their counting counterparts from $\#\mathbf{P}$ in $\mathcal{R}_{\mathcal{C}}$...

Future work

- If we can solve problems from **NP** in $\mathcal{R}$ then we can solve their counting counterparts from $\#\mathbf{P}$ in $\mathcal{R}_{\mathcal{C}}$...
- ... but what happens with non-efficient membrane systems?

Future work

- If we can solve problems from **NP** in $\mathcal{R}$ then we can solve their counting counterparts from $\#\mathbf{P}$ in $\mathcal{R}_{\mathcal{C}}$...
- ... but what happens with non-efficient membrane systems?
- $\texttt{PerfectMatching} \in \mathbf{P}$...

Future work

- If we can solve problems from **NP** in $\mathcal{R}$ then we can solve their counting counterparts from $\#\mathbf{P}$ in $\mathcal{R}_{\mathcal{C}}$...
- ... but what happens with non-efficient membrane systems?
- $\texttt{PerfectMatching} \in \mathbf{P}$...
- ... $\#\texttt{PerfectMatching} \in \#\mathbf{P}$ ($\texttt{Permanent}$)
- $\mathbf{P} = \mathbf{PMC}_{\mathcal{SAM}^0(mcmp,+c,-d,-n)}\cdots$

Future work

- If we can solve problems from **NP** in $\mathcal{R}$ then we can solve their counting counterparts from $\#\mathbf{P}$ in $\mathcal{R}_{\mathcal{C}}$...
- ... but what happens with non-efficient membrane systems?
- $\texttt{PerfectMatching} \in \mathbf{P}$...
- ... $\#\texttt{PerfectMatching} \in \#\mathbf{P}$ ($\texttt{Permanent}$)
- $\mathbf{P} = \mathbf{PMC}_{\mathcal{SAM}^0(mcmp,+c,-d,-n)}\cdots$
- ... $\#\mathbf{P} \subseteq \mathbf{PMC}_{\mathcal{SAM}^0_{\mathcal{C}}(mcmp,+c,-d,-n)}$?