# Polynomial Complexity Classes in Spiking Neural P Systems

Petr Sosík[1,2] Alfonso Rodríguez-Patón[1]
Lucie Ciencialová[2]

[1]Departamento de Inteligencia Artificial, Facultad de Informática
Universidad Politécnica de Madrid, Campus de Montegancedo s/n
Boadilla del Monte, 28660 Madrid, Spain

[2]Institute of Computer Science, Faculty of Philosophy and Science, Silesian University in Opava, 74601 Opava, Czech Republic

Eleventh International Conference on Membrane Computing,
24-27 August 2010, Jena, Germany

# Outline

# Spiking neural P system

## Definition

A spiking neural membrane system of degree $m \geq 1$ is a construct of the form $\Pi = (O, \sigma_1, \ldots, \sigma_m, syn, in, out)$, where:

1. $O = \{a\}$ is the singleton alphabet ($a$ is called *spike*);
2. $\sigma_1, \ldots, \sigma_m$ are neurons;
3. $syn \subseteq \{1, 2, \ldots, m\} \times \{1, 2, \ldots, m\}$ with $(i, i) \notin syn$ for $1 \leq i \leq m$ (*synapses* between neurons);
4. $in, out \in \{1, 2, \ldots, m\}$ indicate the *input neuron* (resp., *output neuron*).

# Spiking neural P system
## Neurons

### Definition

Neurons are on the form $\sigma_i = (n_i, R_i), 1 \leq i \leq m$, where:

- $n_i \geq 0$ is the *initial number of spikes* contained in $\sigma_i$;
- $R_i$ is a finite set of *rules* of the following two forms:
  - $E/a^c \rightarrow a; d$, where $E$ is a regular expression over $a$, $c \geq 1$, and $d \geq 0$;
  - $a^s \rightarrow \lambda$, for some $s \geq 1$, with the restriction that for each rule $E/a^c \rightarrow a; d$ of type (1) from $R_i$, we have $a^s \notin L(E)$;

# Configuration, transition, computation

## Configuration

- The initial configuration of the system is described by the numbers of spikes $n_1, n_2, \ldots, n_m$ present in each neuron.
- During a computation, the "state" of the system is described by
  1. the number of spikes present in each neuron,
  2. the open/closed condition of each neuron: if a neuron is closed, then we have to specify when it will become open again.

# Configuration, transition, computation

## Configuration

- The initial configuration of the system is described by the numbers of spikes $n_1, n_2, \ldots, n_m$ present in each neuron.
- During a computation, the "state" of the system is described by
  1. the number of spikes present in each neuron,
  2. the open/closed condition of each neuron: if a neuron is closed, then we have to specify when it will become open again.

## Transition

Using the rules, we can define transitions among configurations. A transition between two configurations $C_1, C_2$ is denoted by $C_1 \Longrightarrow C_2$.

## Computation

Any sequence of transitions starting in the initial configuration is called a computation. A computation halts if it reaches a configuration where all neurons are open and no rule can be used.

## Computation

Any sequence of transitions starting in the initial configuration is called a computation. A computation halts if it reaches a configuration where all neurons are open and no rule can be used.

## Spike train

With any computation (halting or not) we associate a spike train, the sequence of zeros and ones describing the behavior of the output neuron: if the output neuron spikes, then we write 1, otherwise we write 0.

# Recognizer SN P systems

**Definition (Recognizer SN P system)**

A recognizer SN P system is an SN P system which has only halting computations, and whose output neuron spikes no more than once during each computation. Its computation is called accepting if the output neuron spikes exactly once, otherwise it is rejecting.

# Recognizer SN P systems

## Definition (Recognizer SN P system)

A recognizer SN P system is an SN P system which has only halting computations, and whose output neuron spikes no more than once during each computation. Its computation is called accepting if the output neuron spikes exactly once, otherwise it is rejecting.

## Lemma (Leporati et al., 2009)

*Given a system $\Pi$ with standard or extended rules, with or without delays, we can construct a system $\Pi'$ with rules of the same kinds as those of $\Pi$ which spikes if and only if $\Pi$ halts.*

# Spiking versus halting

## Lemma

*Given a system $\Pi$ with standard or extended rules, with or without delays, we can construct a system $\Pi'$ with rules of the same kinds as those of $\Pi$ which halts if and only if $\Pi$ spikes.*

## Proof

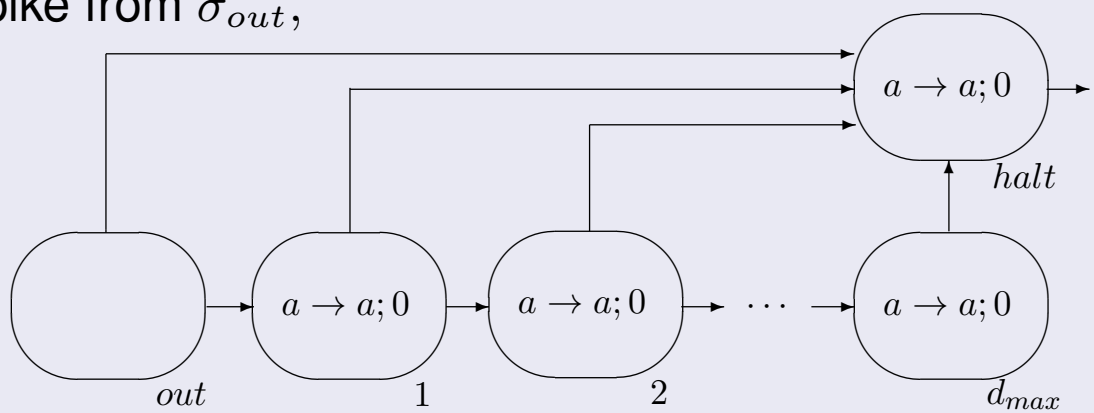
Let us consider an SN P system $\Pi$, and let $\sigma_{out}$ be its output neuron . We construct SN P system $3\Pi$(inspired by [Leporati et al., 2009]) such that we "triple" $\Pi$ by:

- tripling the number of spikes present in the initial configuration in each neuron,
- replacing each rule $E/a^c \to a^p; d$ with $3E/a^{3c} \to a^p; d$, where $3E$ is a regular expression for the set $\{3n \mid n \in L(E)\}$,
- tripling each neuron $\sigma_c$ : adding two identical neurons $\sigma_{c'}$, $\sigma_{c''}$ and adding new synapses: if $\sigma_c$ had originally a synapse to a neuron $\gamma$, now each of $\sigma_c$, $\sigma_{c'}$ and $\sigma_{c''}$ will have synapses to each of $\gamma$, $\gamma'$ and $\gamma''$.

# Spiking versus halting

## Proof

Each neuron in $3\Pi$ spikes if and only if it spikes in $\Pi$. Let $d_{max}$ denote the maximal delay in any neuron of $\Pi$. We construct a module with an incoming synapse from $\sigma_{out}$ which produces $d_{max}+1$ consecutive spikes when obtaining a spike from $\sigma_{out}$,




The neuron $\sigma_{halt}$ will have an outgoing synapse to each neuron of $3\Pi$. During $d_{max}+1$ steps each neuron of $3\Pi$ becomes open and hence it receives a spike from $\sigma_{halt}$. At this moment it will contain

- $3n+1$ spikes, $n \geq 0$, and no rules can be applied.
- $3n+2$ spikes, we add to each neuron of $3\Pi$ the rule $(aaa)^*aa/a \to \lambda$. The system eventually halts after $d_{max}+2$ steps.

# Spiking versus halting

## Proof

Finally, let us add to $3\Pi$ a "perpetuum mobile" circuit of two mutually interconnected neurons with a single initial spike and the rule $a \to a; 0$ in each, and with an incoming synapse from $\sigma_{halt}$. In this way we ensure that $3\Pi$ halts *only* if $\Pi$ spikes.

# Families of recognizer SN P systems

## Definition

A family $\mathbf{\Pi} = \{\Pi(w) : w \in I_X\}$ (respectively, $\mathbf{\Pi} = \{\Pi(n) : n \in \mathbb{N}\}$) of recognizer SN P without input (resp., with input) is polynomially uniform by Turing machines if there exists a deterministic Turing machine working in polynomial time which constructs the system $\Pi(w)$ (resp., $\Pi(n)$) from the instance $w \in I_X$ (resp., from $n \in \mathbb{N}$).

# Polynomial encoding of decision problem

## Definition

Let $X = (I_X, \theta_X)$ be a decision problem, and $\mathbf{\Pi} = \{\Pi(n) : n \in \mathbb{N}\}$ a family of recognizer SN P systems with input membrane. A polynomial encoding of $X$ in $\mathbf{\Pi}$ is a pair $(cod; s)$ of polynomial-time computable functions over $I_X$ such that for each instance $w \in I_X$, $s(w)$ is a natural number (obtained by means of a reasonable encoding scheme) and $cod(w)$ is a binary string — an input of the system $\Pi(s(w))$.

# Polynomial encoding of decision problem

## Definition

Let $X = (I_X, \theta_X)$ be a decision problem, and $\mathbf{\Pi} = \{\Pi(n) : n \in \mathbb{N}\}$ a family of recognizer SN P systems with input membrane. A polynomial encoding of $X$ in $\mathbf{\Pi}$ is a pair $(cod; s)$ of polynomial-time computable functions over $I_X$ such that for each instance $w \in I_X$, $s(w)$ is a natural number (obtained by means of a reasonable encoding scheme) and $cod(w)$ is a binary string — an input of the system $\Pi(s(w))$.

## Lemma (Pérez-Jiménez et al. 2006)

*Let $X_1, X_2$ be decision problems, $r$ a polynomial-time reduction from $X_1$ to $X_2$, and $(cod; s)$ a polynomial encoding of $X_2$ in $\mathbf{\Pi}$. Then, $(cod \circ r; s \circ r)$ is a polynomial encoding of $X_1$ in $\mathbf{\Pi}$.*

# A decision problem is solvable by family of recognizer SN P systems

Let $\mathcal{R}$ denote an arbitrary type of recognizer SN P systems.

## Definition

Let $f : \mathbb{N} \to \mathbb{N}$ be a constructible function. A decision problem $X$ is solvable by a family $\mathbf{\Pi} = \{\Pi(w) : w \in I_X\}$ of recognizer SN P systems of type $\mathcal{R}$ without input in time bounded by $f$, denoted by $X \in \mathbf{SN}^*_{\mathcal{R}}(f)$, if the following holds:

- The family $\mathbf{\Pi}$ is polynomially uniform by Turing machines.
- The family $\mathbf{\Pi}$ is $f$-bounded with respect to $X$; that is, for each instance $w \in I_X$, every computation of $\Pi(w)$ performs at most $f(|w|)$ steps.
- The family $\mathbf{\Pi}$ is sound with respect to $X$; that is, for each $w \in I_X$, if there exists an accepting computation of $\Pi(w)$, then $\theta_X(w) = 1$.
- The family $\mathbf{\Pi}$ is complete with respect to $X$; that is, for each $w \in I_X$, if $\theta_X(w) = 1$, then every computation of $\Pi(w)$ is an accepting computation.

# Families of recognizer SN P systems

The SN P system solving an instance $w$ can be generally nondeterministic, i.e, it may have different possible computations, but with the same result. Such a P system is also called confluent.
The family $\mathbf{\Pi}$ is said to provide a semi-uniform solution to the problem $X$. In this case, for each instance of $X$ we have a special P system. Specifically, we denote by

$$\mathbf{PSN}^*_{\mathcal{R}} = \bigcup_{f \text{ polynomial}} \mathbf{SN}^*_{\mathcal{R}}(f)$$

the class of problems to which uniform families of SN P systems of type $\mathcal{R}$ without input provide semi-uniform solution in polynomial time.

# Families of recognizer SN P systems

## Definition

Let $f : \mathbb{N} \to \mathbb{N}$ be a constructible function. A decision problem $X$ is solvable by a family $\mathbf{\Pi} = \{\Pi(n) : n \in \mathbb{N}\}$ of recognizer SN P systems of type $\mathcal{R}$ with input in time bounded by $f$, denoted by $X \in \mathbf{SN}_{\mathcal{R}}(f)$, if the following holds:

- The family $\mathbf{\Pi}$ is polynomially uniform by Turing machines.
- There exists a polynomial encoding $(cod, s)$ of $X$ in $\Pi$ such that:
  - The family $\mathbf{\Pi}$ is $f$-bounded with respect to $X$; that is, for each instance $w \in I_X$, every computation of $\Pi(s(w))$ with input $cod(w)$ performs at most $f(|w|)$ steps.
  - The family $\mathbf{\Pi}$ is sound with respect to $(X, cod, s)$; that is, for each $w \in I_X$, if there exists an accepting computation of $\Pi(s(w))$ with input $cod(w)$, then $\theta_X(w) = 1$.
  - The family $\mathbf{\Pi}$ is complete with respect to $(X, cod, s)$; that is, for each $w \in I_X$, if $\theta_X(w) = 1$, then every computation of $\Pi(s(w))$ with input $cod(w)$ is an accepting computation.

# Families of recognizer SN P systems

The family $\mathbf{\Pi}$ is said to provide a uniform solution to the problem $X$. We denote by

$$\mathbf{PSN}_{\mathcal{R}} = \bigcup_{f \text{ polynomial}} \mathbf{SN}_{\mathcal{R}}(f)$$

the class of problems to which uniform families of SN P systems of type $\mathcal{R}$ with input provide uniform solution in polynomial time. Obviously, for any constructible function $f$ and a class of SN P systems $\mathcal{R}$ we have

$$\mathbf{SN}_{\mathcal{R}}(f) \subseteq \mathbf{SN}^*_{\mathcal{R}}(f) \quad \text{and} \quad \mathbf{PSN}_{\mathcal{R}} \subseteq \mathbf{PSN}^*_{\mathcal{R}}.$$

# About $\mathbf{SN}_{\mathcal{R}}(f)$ and $\mathbf{SN}^{*}_{\mathcal{R}}(f)$

**Theorem**

*The classes $\mathbf{SN}_{\mathcal{R}}(f)$ and $\mathbf{SN}^{*}_{\mathcal{R}}(f)$ are closed under the operation of complement.*

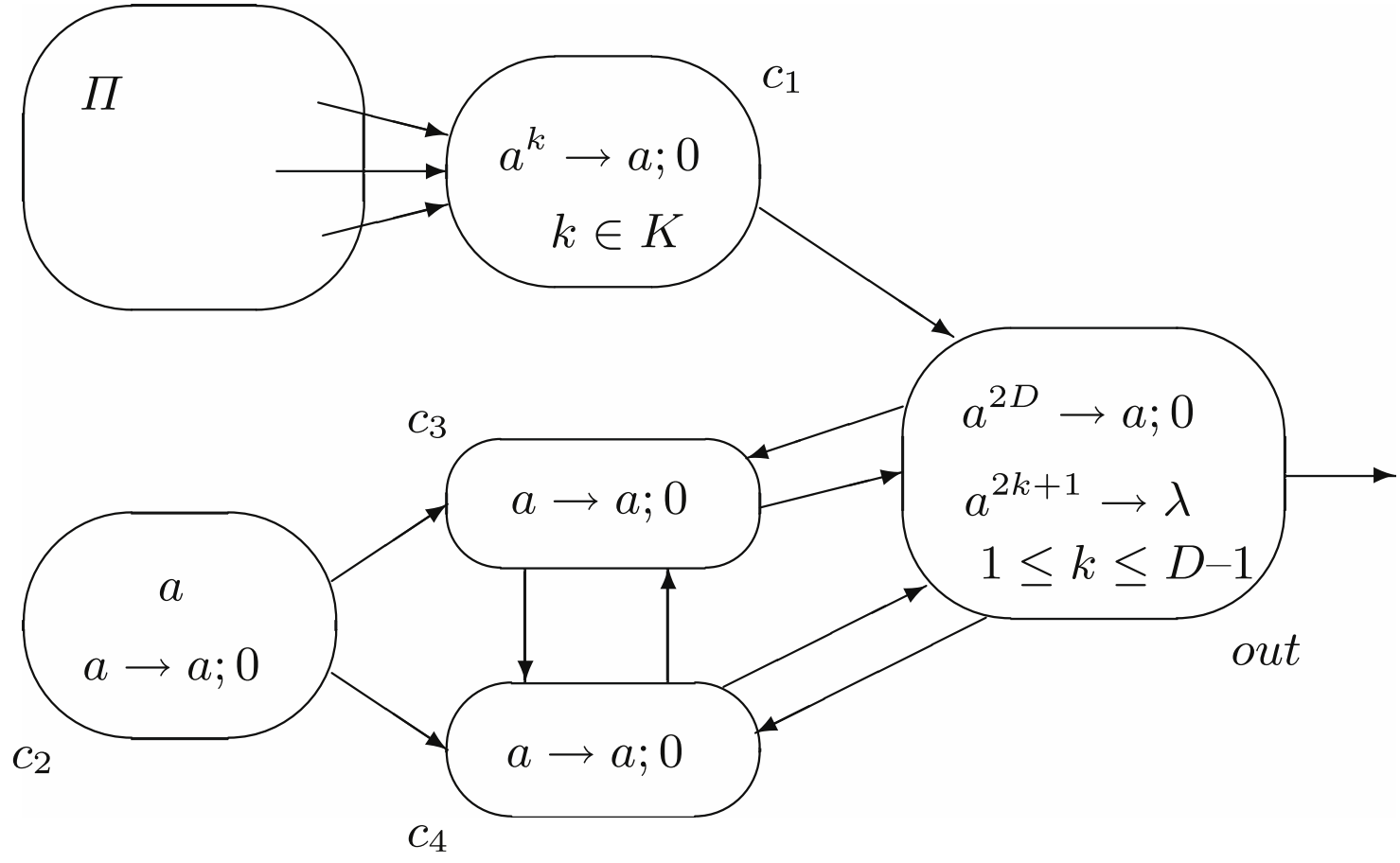
# About $\mathbf{SN}_{\mathcal{R}}(f)$ and $\mathbf{SN}^{*}_{\mathcal{R}}(f)$

## Theorem

*The classes $\mathbf{SN}_{\mathcal{R}}(f)$ and $\mathbf{SN}^{*}_{\mathcal{R}}(f)$ are closed under the operation of complement.*

## Proof

It is necessary to show that for each confluent SN P system $\Pi$ there exists a system $\Pi'$ whose computation is accepting if and only if the computation of $\Pi$ is rejecting. Assume the construction described in [Leporati et al. 2009]. It presents a module which, when added to any SN P system $\Pi$, emits a spike only after the system $\Pi$ halts.




[Leporati et al. 2009]

**Proof.**

Let us extend this module as follows. Let $\sigma_{out}$ be the output neuron of this module. Let a spike emitted from $\sigma_{out}$ after halting of the system $\Pi$ feed two new neurons, each with a rule $a \to a; 0$. Finally, add a new neuron $\sigma_{out'}$ with incoming synapses from these two neurons, another synapse from the original output neuron of $\Pi$, and with a rule $a^2 \to a; 0$. Let $\sigma_{out'}$ be the output neuron of $\Pi'$. Neuron $\sigma_{out'}$ spikes if and only if $\Pi$ halts and its output neuron $\sigma_{out}$ does not spike which concludes the proof.

**Theorem**

*Let $\mathcal{R}$ be a class of SN P systems. Let $X$ and $Y$ be decision problems such that $X$ is reducible to $Y$ in polynomial time. If $Y \in \mathbf{PSN}_{\mathcal{R}}$ (respectively, $Y \in \mathbf{PSN}^{*}_{\mathcal{R}}$), then $X \in \mathbf{PSN}_{\mathcal{R}}$ (resp., $X \in \mathbf{PSN}^{*}_{\mathcal{R}}$).*

**Theorem**

*Let $\mathcal{R}$ be a class of SN P systems. Let $X$ and $Y$ be decision problems such that $X$ is reducible to $Y$ in polynomial time. If $Y \in \mathbf{PSN}_{\mathcal{R}}$ (respectively, $Y \in \mathbf{PSN}^{*}_{\mathcal{R}}$), then $X \in \mathbf{PSN}_{\mathcal{R}}$ (resp., $X \in \mathbf{PSN}^{*}_{\mathcal{R}}$).*

**Proof.**

Let $\mathbf{\Pi}$ by a family providing uniform solution to the problem $Y$. By its definition, let $p$ be a polynomial and $(cod, s)$ a polynomial encoding of $Y$ in $\mathbf{\Pi}$ such that $\mathbf{\Pi}$ is $p$-bounded with respect to $Y$ and sound and complete with respect to $(Y, cod, s)$.
Let further $r : I_X \to I_Y$ be a polynomial time reduction from $X$ to $Y$, and hence there exists a polynomial $q$ such that for each $w \in I_X$, $|r(w)| \leq q(|w|)$.

**Proof.**

Observe that:

- $(cod \circ r; s \circ r)$ is a polynomial encoding of $X$ in $\mathbf{\Pi}$.
- $\mathbf{\Pi}$ is $(p \circ q)$-bounded with respect to $X$ since for each $w \in I_X$, every computation of $\Pi(s(r(w)))$ with input $cod(r(w))$ performs at most $p(|r(w)|) \leq p(q(|w|))$ steps.
- $\mathbf{\Pi}$ is sound and complete with respect to $(X, cod \circ r, s \circ r)$ since for each $w \in I_X$,
  - if there exists an accepting computation of $\Pi(s(r(w)))$ with input $cod(r(w))$, then $\theta_Y(r(w)) = 1$ and, by reduction, also $\theta_X(w) = 1$,
  - if $\theta_X(w) = 1$, then also $\theta_Y(r(w)) = 1$ and hence every computation of $\Pi(s(r(w)))$ with input $cod(r(w))$ is an accepting computation.

Consequently, $X \in \mathbf{SN}_{\mathcal{R}}(p \circ q)$ and hence also in $\mathbf{PSN}_{\mathcal{R}}$.

# SN P systems with restrictions imposed on their regular expressions

It was shown already in [García-Arnau et al. 2009] that SN P systems without delays and with all regular expressions of the form $a^n$, $n \geq 1$, are computationally universal. Results in [Leporati et al. 2007] together with Theorems 1 and 4 in [Rodríguez-Patón et al. 2010] imply the following statement.

**Theorem**

$$\mathbf{PSN}_{-reg,-del} = \mathbf{PSN}^*_{-reg,-del} = \mathbf{PSN}_{ssnf} = \mathbf{PSN}^*_{ssnf} = \mathbf{P}$$

These results show that families of standard confluent SN P systems can reach the computational power beyond **P** only with the aid of complex regular expressions. Whenever we release the condition of single star normal forms in regular expressions, the computational power of SN P systems reaches the class **NP**.

# PSN* ⊆ PSPACE

## Lemma

*Matching of a regular expression $E$ of size $s$ in succinct form over a singleton alphabet with a string $a^k$ can be done on a RAM or a Turing machine in non-deterministic polynomial time with respect to $s \log k$.*

## Proof.

We have the syntactic tree of the expression $E$ at our disposal.
Its parsing can be done in deterministic polynomial time.
We treat the sub-expressions of the form $a^n$ as constants and assign them a leaf node of the tree with the value $n$.
The matching algorithm works as follows:

**Proof.**

- Produce non-deterministically a random element of $L(E)$ in succinct form by a depth-first search traversal of its syntactic tree. Start with the value 0 and evaluate recursively each node depending on its type as follows:
  - leaf node containing a constant: return the value of the node;
  - catenation: evaluate both subtrees of this node and add the results;
  - union: choose non-deterministically one of the subtrees of this node and evaluate it;
  - star: draw a random number of iterations $x$ within the range $\langle 0, k \rangle$, and if $x > 0$, evaluate the subtree starting in this node and multiply the result by $x$, otherwise return 0.
- Compare the drawn element of $L(E)$ with $a^k$ whether they are equal.

**Proof.**

Whenever during the evaluation the computed value exceeds $k$, the algorithm halts immediately and reports that $a^k$ does not match $L(E)$. This guarantees that the number of bits processed in each operation is always $O(\log k)$.

Each of the elementary operations described above can be performed in constant time on RAM with unit instruction price, except the multiplication which requires $O(\log k)$ time. Total number of tree-traversal steps is $\mathcal{O}(s)$. If we implement the algorithm on Turing machine, the time increases only polynomially.

## Theorem

$\mathbf{PSN}^* \subseteq \mathbf{PSPACE}$

## Proof.

- It has been shown in [Leporati et al. 2007] that any confluent SN P system with simple regular expressions can be simulated by a deterministic Turing machine in polynomial time.
- Concerning general regular expressions, by previous lemma their matching can be done in non-deterministic polynomial time, and since $\mathbf{NP} \subseteq \mathbf{PSPACE}$, also in deterministic polynomial space.
- Indeed, if one replaces the random selection in the proof of previous lemma by dept-first-search of all configurations reachable by making nondeterministic choices, one gets a deterministic algorithm performing matching in polynomial space and exponential time.

**Proof.**

Denote by $s$ the size of description of a SN P system $\Pi$. Observe that the total number of bits to describe spikes in all neurons after $t$ steps of computation is $\mathcal{O}(s+t)$ even in the case of maximal parallelism or exhaustive rules. The total size of all regular expressions in $\Pi$ is $\mathcal{O}(s)$. Hence, by previous lemma, the simulation of $\Pi$ performs in polynomial space with respect to $s+t$.

Thank you for your attention!