Semantics of deductive databases with spiking neural P systems

Daniel Díaz-Pernil [a], Miguel A. Gutiérrez-Naranjo [b,*]

[a] CATAM Research Group - Department of Applied Mathematics I, University of Seville, Spain
[b] Department of Computer Science and Artificial Intelligence, University of Seville, Spain

A B S T R A C T

The integration of symbolic reasoning systems based on logic and connectionist systems based on the functioning of living neurons is a vivid research area in computer science. In the literature, one can find many efforts where different reasoning systems based on different logics are linked to classic artificial neural networks. In this paper, we study the relation between the semantics of reasoning systems based on propositional logic and the connectionist model in the framework of membrane computing, namely, spiking neural P systems. We prove that the fixed point semantics of deductive databases without negation can be implemented in the spiking neural P systems model and such a model can also deal with negation if it is endowed with anti-spikes and annihilation rules.

Keywords:
Membrane computing
P systems
Neural-symbolic integration

1. Introduction

Searching new ways for automating reasoning is one of the engines in computer science. In the last decades, the two main research areas devoted to this aim, namely, connectionist systems ans logic-based systems, have explored very different techniques, approaches and representation systems.

On the one hand, logic-based systems are based on a symbolic representation of the knowledge and derivation rules which allow to obtain new formulae from previous ones. Many efforts have done in order to prove the correctness and soundness of the rules depending on the expressiveness of the language and the chosen semantics.

On the other hand, the traditional connectionist systems used to handle automatic reasoning are based on artificial neural nets inspired in the network of biological neurons in a human brain. These nets have been also widely studied and provide a point of view where the data is encoded in real numbers and the synapses between neurons determine the flow of information.

The integration of both paradigms is a vivid area in artificial intelligence (see, e.g., [1–3]). In the framework of membrane computing, several studies have been presented where P systems are used for representing logic-based information and performing reasoning by the application of bio-inspired rules (see [4,5]). These papers study approaches based on cell-like models, as P systems with active membranes, and deal with procedural aspects of the computation. The approach in this paper is different in both senses.

On the one hand, the connectionist model of P systems is considered, i.e, the model of P system inspired by the neurophysiological behavior of neurons sending electrical impulses along axons to other neurons (the so-called spiking neural P systems or SN P systems for short). On the second hand, we consider the semantics of propositional deductive databases in order to show how SN P systems can deal with logic-based representing and reasoning systems.

One of the key features of the integrate-and-fire formal spiking neuron models [6] (and, in particular, of the SN P systems) is the use of the *spikes* as support of the information. Such spikes are short electrical pulses (also called action potentials) between neurons and can be observed by placing a fine electrode close to the soma or axon of a neuron. From the theoretical side, it is crucial to consider that all the biological spikes of an alive biological neuron look alike. This means that we can consider a bio-inspired binary code which can be used to formalize logic-based semantics: the emission of one spike will be interpreted as *true* and the absence of spikes will be interpreted as *false*.

As we will show below, SN P systems provide a natural way for dealing with this binary behavior in a connectionist model. SN P systems suffice for dealing with the semantics of propositional logic systems which do not use negation. The semantics of reasoning systems with negated information, even in the propositional case, needs to add new elements for dealing with the difference between negated information and absence of information (see [7]).

The literature of SN P system provides an efficient tool for dealing with such negated information: the use of anti-spikes and

* Corresponding author.
E-mail addresses: sbdani@us.es (D. Díaz-Pernil), magutier@us.es (M.A. Gutiérrez-Naranjo).

annihilation rules. SN P systems with anti-spikes were presented in [8] as a formalization of the idea of inhibiting in some way the communication between neurons. Such extended model has been widely studied (see, e.g., [9–11]) and it has inspired the use of negative information (see, e.g., [12–14]). Recently, the study of SN P systems has been extended with different features, see e.g., SN P systems *with thresholds* [15], SN P systems *with request rules* [16], SN P systems *with structural plasticity* [17] or *cell-like* SN P systems [18]. Among the recent contributions, the approaches shown in [19–21] deserve to be cited. In such papers, the interaction between reasoning systems and membrane computing is also studied. Instead of dealing with propositional logic, the authors consider fuzzy logic and the applications focus on fuzzy representation of the knowledge and fault diagnosis.

The main result of this paper[1] is to prove that given a reasoning system based on propositional logic it is possible to find an SN P system with the same declarative semantics. We prove it in both cases: when the reasoning system involves negation and when it does not. A declarative semantics for a rule-based propositional system is usually given by selecting models which satisfy certain properties. This choice is often described by an operator mapping interpretations to interpretations. In this paper we consider the so-called *immediate consequence operator* due to van Emden and Kowalski [22]. It is well-known that for a rule based system without negation *KB*, such operator is order continuous and its least fix point coincides with the least model of *KB*. We adapt the definition of the immediate consequence operator to a restricted form of SN P system and we prove that a least fix point, and hence a least model is obtained for the given reasoning system. The monotonicity is lost if negation is allowed, but even in this case, the immediate consequence operator can be computed with membrane computing techniques.

The paper is organized as follows: firstly, we recall some aspects about SN P systems and the semantics of deductive databases. In Section 3 we prove that standard SN P systems can deal with the semantics of deductive databases if they do not involve negation. In Section 4 it is shown that endowing SN P systems with anti-spikes and annihilation, then such devices can deal with the semantics of deductive databases with negation. Finally, some conclusions and topics for further discussion are provided in the last section.

2. Preliminaries

We assume the reader to be familiar with basic elements about membrane computing and the semantics of rule-based systems. Next, we briefly recall some definitions. We refer to [23] for a comprehensive presentation of the former and [7,24,25] for the latter.

2.1. Spiking neural P systems

SN P systems were introduced in [26] with the aim of incorporating membrane computing ideas with spike-based neuron models. It is a class of distributed and parallel computing devices, inspired by the neurophysiological behavior of neurons sending electrical impulses (*spikes*) along axons to other neurons.

In SN P systems the cells (also called *neurons*) are placed in the nodes of a directed graph, called the *synapse graph*. The contents of each neuron consist of a number of copies of a single object type, called the *spike*. Every cell may also contain a number of *firing* and *forgetting* rules. Firing rules allow a neuron to send information to other neurons in the form of *spikes* which are accumulated at the target cell. The applicability of each rule is determined by checking the contents of the neuron against a regular set associated with the rule. In each time unit, if a neuron can use one of its rules, then one of such rules must be used. If two or more rules could be applied, then only one of them is non-deterministically chosen. Thus, the rules are used in the sequential manner in each neuron, but neurons function in parallel with each other. As usually happens in membrane computing, a global clock is assumed, marking the time for the whole system, and hence the functioning of the system is synchronized.

Formally, an SN P system of the degree $m \geq 1$ is a construct[2]

$$\Pi = (O, \sigma_1, \sigma_2, \ldots, \sigma_m, syn)$$

where:

1. $O = \{a\}$ is the singleton alphabet (a is called *spike*);
2. $\sigma_1, \sigma_2, \ldots, \sigma_m$ are *neurons*, of the form $\sigma_i = (n_i, R_i)$, $1 \leq i \leq m$, where:
   (a) $n_i \geq 0$ is the *initial number of spikes* contained in σ_i;
   (b) R_i is a finite set of *rules* of the following two forms:
      (1) *firing* rules $E/a^p \to a^q$, where E is a regular expression over a and p, $q \geq 1$ are integer numbers[3];
      (2) *forgetting* rules $a^s \to \lambda$, with s an integer number such that $s \geq 1$;
3. $syn \subseteq \{1, 2, \ldots, m\} \times \{1, 2, \ldots, m\}$, with $(i, i) \notin syn$ for $1 \leq i \leq m$, is the directed graph of *synapses* between neurons.

Rules of type (1) are firing rules, and they are applied as follows. If the neuron σ_i contains k spikes, $k \geq p$, and a^k belongs to the language $L(E)$ associated to the regular expression E, then the rule $E/a^p \to a^q$ can be applied. The application of this rule means removing p spikes (thus only $k - p$ remain in σ_i), the neuron is fired, and it produces q spikes which are sent immediately to all neurons σ_j such that $(i, j) \in syn$. The rules of type (2) are forgetting rules and they are applied as follows: if the neuron σ_i contains exactly s spikes, then the rule $a^s \to \lambda$ from R_i can be used, meaning that all s spikes are removed from σ_i. If a rule $E/a^p \to a^q$ of type (1) has $E = a^p$, then we will write it in the simplified form $a^p \to a^q$. In each time unit, if a neuron σ_i can use one of its rules, then a rule from R_i must be used. Let us remark that it is possible that two or more rules can be applied in a neuron, and in that case only one of them is non-deterministically chosen regardless its type.. The jth configuration of the system is described by a vector $\mathbb{C}_j = (t_1, \ldots, t_m)$ where t_k represents the number of spikes at the neuron σ_k in such configuration. The initial configuration is $\mathbb{C}_0 = (n_1, n_2, \ldots, n_m)$. Using the rules as described above, one can define transitions among configurations. Any sequence of transitions starting in the initial configuration is called a computation. A computation halts if it reaches a configuration where no rule can be used. Generally, a computation may not halt. If it halts, the last configuration is called a *halting* configuration.

A useful extension to the model presented above was introduced by adding anti-spikes as a formalization of the idea of inhibiting in some way the communication between neurons (see [8]). This extension leads to the definition of *SN P systems with anti-spikes*. In this extension a further object, $\overline{a}$, is added to the alphabet O, and the spiking and forgetting rules are of the forms $E/b^p \to b'^q$ and $b^p \to \lambda$ where E is a regular expression over a or over $\overline{a}$, while $b, b' \in \{a, \overline{a}\}$ and p, $q \geq 1$. As above, if $L(E) = b^p$, then we write the first rule as $b^p \to b'^q$. The rules are used as in a usual SN P system, with the additional fact that a and $\overline{a}$ cannot stay

[1] A preliminary version of this paper appeared in the Proceedings of the 14th Brainstorming Week on Membrane Computing (BWMC2016).

[2] We provide a definition without delays, input or output neurons because these features are not used in this paper.

[3] In the general case, the restriction $p \geq q$ is imposed. In this paper, we will also consider rules $E/a \to a^2$. A more complex design of our solutions satisfying $p \geq q$ is possible, but we prefer to include rules $E/a \to a^2$ for the sake of a simpler design.

together: if in a neuron there are either objects a or objects $\bar{a}$, and further objects of either type (maybe both) arrive from other neurons, such that we end with a^r and $\bar{a}^s$ inside, then immediately an annihilating rule of the form $a\bar{a} \to \lambda$ is applied in a maximal manner, so that either a^{r-s} s or $\bar{a}^{s-r}$ remain, provided that $r \geq s$ or $s \geq r$, respectively. The mutual annihilation of spikes and anti-spikes takes no time, so that the neuron always contains either only spikes or anti-spikes. The jth configuration of the system is described by a vector $\mathbb{C}_j = (t_1, \ldots, t_m)$. If $t_i \geq 0$, it denotes that the neuron σ_i has t_i spikes at the configuration $\mathbb{C}_j$. If $t_i = -s_i$ with $s_i \geq 1$, it denotes that the neuron σ_i has s_i anti-spikes at the configuration $\mathbb{C}_j$.

2.2. Semantics of rule-based deductive databases

Given two pieces of knowledge V and W, expressed in some language, the rule $V \to W$ is usually considered as a causal relation between V and W. In this paper, we only consider propositional logic for representing the knowledge. Given a set of propositional variables $\{p_1, \ldots, p_n\}$, a literal is a variable or a negated variable and a rule is a formula $L_1 \wedge \cdots \wedge L_n \to A$, where $n \geq 0$, A is a variable and $L_1, \ldots, L_n$ are literals. The variable A is called the *head* of the rule and the conjunction of literals $L_1 \wedge \cdots \wedge L_n$ is the *body* of the rule. If $L_i = p_i$, it is said that p_i occurs positively in the body of the rule. If $L_i = \neg p_i$, it is said that p_i occurs negatively in the body of the rule. If $n = 0$, it is said that the body of the rule is empty. If there do not exist negated variables in the body of a rule, the rule is called *definite*. A finite set of rules KB is a deductive database and it is said that it is a definite deductive database if all the rules in KB are definite. An interpretation I is a mapping from the set of variables $\{p_1, \ldots, p_n\}$ to the set $\{0, 1\}$. As usual, we will represent an interpretation I as a vector $(i_1, \ldots, i_n)$ with $I(p_k) = i_k \in \{0, 1\}$ for $k \in \{1, \ldots, n\}$. The set of all the possible interpretations for a set of n variables will be denoted by 2^n. Given two interpretations I_1 and I_2, $I_1 \subseteq I_2$ if for all $k \in \{1, \ldots, n\}$, $I_1(p_k) = 1$ implies $I_2(p_k) = 1$. We will denote by $I_\emptyset$ the interpretation that maps to 0 every variable, $I_\emptyset = (0, \ldots, 0)$. The interpretation I is extended in the usual way, $I(\neg p_i) = 1 - I(p_i)$ for a variable p_i; $I(L_1 \wedge \cdots \wedge L_n) = \min\{I(L_1), \ldots, I(L_n)\}$ and for a rule[4]

$$I(L_1 \wedge \cdots \wedge L_n \to A) = \begin{cases} 0 & \text{if } I(L_1 \wedge \cdots \wedge L_n) = 1 \text{ and } I(A) = 0 \\ 1 & \text{otherwise} \end{cases}$$

An interpretation I is a model for a deductive database KB if $I(R) = 1$ for all $R \in KB$. Next, we recall the propositional version of the immediate consequence operator which was introduced by van Emden and Kowalski [22].

Definition 1. Let KB be a deductive database on a set of variables $\{p_1, \ldots, p_n\}$. The immediate consequence operator of KB is the mapping T_{KB}: $2^n \to 2^n$ such that for all interpretation I, $T_{KB}(I)$ is an interpretation

$$T_{KB}(I) : \{p_1, \ldots, p_n\} \to \{0, 1\}$$

such that, for $k \in \{1, \ldots, n\}$, $T_{KB}(I)(p_k) = 1$ if there exists a rule $L_1 \wedge \cdots \wedge L_n \to p_k$ in KB such that $I(L_1 \wedge \cdots \wedge L_n) = 1$; otherwise, $T_{KB}(I)(p_k) = 0$.

The importance of the immediate consequence operator is shown in the following theorem (see [27]).

Theorem 1. *An interpretation I is a model of KB if and only if $T_{KB}(I) \subseteq I$.*

Since the image of an interpretation is an interpretation, the immediate consequence operator can be iteratively applied.

[4] Let us remark that, from the definition, if $n = 0$, $I(L_1 \wedge \cdots \wedge L_n) = 1$ and, hence, for a rule with an empty body, we have $I(\to A) = 1$ if and only if $I(A) = 1$.

Definition 2. Let KB be a deductive database and T_{KB} its immediate consequence operator. The mapping $T_{KB}\uparrow$: $\mathbb{N} \to 2^n$ is defined as follows: $T_{KB} \uparrow 0 = I_\emptyset$ and $T_{KB} \uparrow n = T_{KB} \uparrow (T_{KB} \uparrow (n-1))$ if $n > 0$. In the limit, it is also considered

$$T_{KB} \uparrow \omega = \bigcup_{k \geq 0} T_{KB} \uparrow k$$

The next theorem is a well-known result which relates the immediate consequence operator of a definite deductive database with its least model of a definite deductive database (see [25]).

Theorem 2. *Let KB be a definite deductive database. The following result hold*

- *$T_{KB}\uparrow\omega$ is a model of KB*
- *If I is a model of KB, then $T_{KB}\uparrow\omega \subseteq I$*

Example 1. Let us consider the following deductive database KB on the set of variables $\Gamma = \{p_1, p_2, p_3, p_4\}$

$R_1 \equiv \to p_1$
$R_2 \equiv p_1 \to p_2$
$R_3 \equiv p_1 \wedge p_2 \to p_3$
$R_4 \equiv p_3 \to p_4$
$R_5 \equiv p_2 \to p_4$

and let us consider the interpretation I: $\Gamma \to \{0, 1\}$ such that $I(p_1) = 1$, $I(p_2) = 0$, $I(p_3) = 0$ and $I(p_4) = 0$. Such interpretation can be represented as $I = (1, 0, 0, 0)$. The truth assignment of this interpretation to the rules is $I(R_1) = 1$, $I(R_2) = 0$, $I(R_3) = 1$, $I(R_4) = 1$, $I(R_5) = 1$. Since $I(R_2) = 0$, the interpretation I is not a model for KB. The application of the immediate consequence operator produces $T_{KB}(I) = (1, 1, 0, 0)$. We observe that $T_{KB}(I) \not\subseteq I$ and hence, by Theorem 1, we can also conclude that I is not a model for KB. Finally, if we consider $I_\emptyset = (0, 0, 0, 0)$, the following interpretations are obtained by the iterative application of the immediate consequence operator

$T_{KB} \uparrow 0 = I\emptyset = (0, 0, 0, 0)$
$T_{KB} \uparrow 1 = T_{KB}(T_{KB} \uparrow 0) = (1, 0, 0, 0)$
$T_{KB} \uparrow 2 = T_{KB}(T_{KB} \uparrow 1) = (1, 1, 0, 0)$
$T_{KB} \uparrow 3 = T_{KB}(T_{KB} \uparrow 2) = (1, 1, 1, 1)$

In this case $T_{KB}\uparrow 3$ is a fix point for the immediate consequence operator and a model for the definite deductive database KB.

3. Semantics of deductive databases with SN P systems

The semantics of deductive databases deals with interpretations, i.e., with mappings from the set of variables into the set $\{0, 1\}$ (which stand for *false and true*) and try to characterize which of these interpretations make true a whole deductive database which, from the practical side, may contain hundreds of variables and thousand of rules. The immediate consequence operator provides a tool for dealing with this problem and provides a way to characterize such models. In this section we will explore how this problem can be studied in the framework of SN P systems and prove that the immediate consequence operator can be implemented in this model and therefore, membrane computing provides a new theoretical framework for dealing with the semantics of deductive databases.

Our main result for definite deductive databases claims that SN P systems can compute the immediate consequence operator and hence, its least model.

Theorem 3. *Given a definite deductive database KB and an interpretation I, an SN P system can be constructed such that*

(a) *It computes the immediate consequence operator $T_{KB}(I)$.*
(b) *It computes the least model for KB in a finite number of steps.*

Proof. Let us consider a deductive database *KB*, let $\{p_1 \ldots, p_n\}$ be the propositional variables and $\{r_1, \ldots, r_k\}$ be the rules of *KB*. Given a variable p_i, we will denote by h_i the number of rules which have p_i in the head and given a rule r_j, we will denote by b_j the number of variables in its body. The SN P systems of degree $2n+k+3$

$$\Pi_{KB} = (O, \sigma_1, \sigma_2, \ldots, \sigma_{2n+k+2}, syn)$$

can be constructed as follows:

- $O = \{a\}$;
- $\sigma_j = (0, \{a \to \lambda\})$ for $j \in \{1, \ldots n\}$
- $\sigma_{n+j} = (i_j, R_j)$, $j \in \{1, \ldots, n\}$, where $i_j = I(p_j)$ and R_j is the set of h_j rules $R_j = \{a^k \to a | k \in \{1, \ldots\ldots, h_j\}\}$
- $\sigma_{2n+j} = (0, R_j)$, $j \in \{1, \ldots\ldots, k\}$, where R_j is one of the following set of rules
  - $R_j = \{a^{b_j} \to a\} \cup \{a^l \to \lambda | l \in \{1, \ldots, b_j - 1\}\}$ if $b_j > 0$
  - $R_j = \{a \to a\}$ if $b_j = 0$.

For a better understanding, the neurons σ_{2n+k+1} and σ_{2n+k+2} will be denoted by σ_G and σ_T, respectively.

- $\sigma_G = (0, \{a \to a\})$
- $\sigma_T = (1, \{a \to a\})$
- $syn = \{(n+i, i) | i \in \{1, \ldots, n\}\}$
  $\cup \left\{ (n+i, 2n+j) \middle| \begin{array}{l} i \in \{1, \ldots, n\}, j \in \{1, \ldots, k\} \\ \text{and } p_i \text{ is a variable in the body of } r_j \end{array} \right\}$
  $\cup \left\{ (2n+j, n+i) \middle| \begin{array}{l} i \in \{1, \ldots, n\}, j \in \{1, \ldots, k\} \\ \text{and } p_i \text{ is the variable in the head of } r_j \end{array} \right\}$
  $\cup \{(G, T), (T, G)\}$
  $\cup \left\{ (T, 2n+j) \middle| \begin{array}{l} j \in \{1, \ldots, k\} \\ \text{and } r_j \text{ is a rule with empty body} \end{array} \right\}$

□

Before going on with the proof, let us note that the construction of this SN P system is illustrated in the Example 2. The next remarks will be useful:

Remark 1. For all $t \geq 0$, in the $2t$th configuration $\mathbb{C}_{2t}$ the neuron σ_T contains exactly one spike and the neuron σ_G does not contain spikes.

Proof. In the initial configuration $\mathbb{C}_0$, σ_T contains 1 spike and σ_G does not contain spikes. By induction, let us suppose that in the $\mathbb{C}_{2t}$ the neuron σ_T contains exactly one spike and σ_G does not contain spikes. Since the unique incoming synapse in σ_T comes from σ_G and the unique incoming synapse in σ_G comes from σ_T and in both neurons occurs the rule $a \to a$, then in $\mathbb{C}_{2t+1}$ the neuron σ_G contains exactly spike and σ_T does not contain spikes and finally, in $\mathbb{C}_{2t+2}$ the neuron σ_T contains exactly spike and σ_G does not contain spikes. □

Remark 2. For all $t \geq 0$ the following results hold:

- For all $p \in \{1, \ldots, k\}$ the neuron σ_{2n+p} does not contain spikes in the configuration $\mathbb{C}_{2t}$
- For all $q \in \{1, \ldots, n\}$, the neuron σ_{n+q} does not contain spikes in the configuration $\mathbb{C}_{2t+1}$

Proof. In the initial configuration $\mathbb{C}_0$, for all $p \in \{1, \ldots, k\}$, the neuron σ_{2n+p} does not contain spikes and each neuron σ_{n+q} contain, at most, one spike. Such spike is consumed by the application of the rule $a \to a$ and, since all the neurons with synapse to σ_{n+q} do not contain spikes at $\mathbb{C}_0$, we conclude that at the configuration $\mathbb{C}_1$, the neurons σ_{n+q} do not contain spikes.

By induction, let us suppose that in $\mathbb{C}_{2t}$, for all $p \in \{1, \ldots, k\}$, the neuron σ_{2n+p} does not contain spikes and for all $q \in \{1, \ldots, n\}$, the neuron σ_{n+q} does not contain spikes in the configuration $\mathbb{C}_{2t+1}$. According to the construction, the number of incoming synapses in each neuron σ_{2n+j} is b_j if $b_j > 1$ and 1 if $b_j = 0$. Such synapses come from neurons that send (at most) one spike in each computational step, so in $\mathbb{C}_{2t+1}$, the number of spikes in the neuron σ_{2n+j} is, at most, b_j if $b_j > 1$ and 1 if $b_j = 0$. All these spikes are consumed by the corresponding rules. Moreover, at $\mathbb{C}_{2t+1}$, all the neurons with outgoing synapses to σ_{2n+p} do not contain spikes, so we conclude that at $\mathbb{C}_{2t+2}$, for all $j \in \{1, \ldots, k\}$, the neuron σ_{2n+j} does not contain spikes. We focus now on the neurons σ_{n+q} with $q \in \{1, \ldots, n\}$. By induction, we assume that they do not contain spikes in the configuration $\mathbb{C}_{2t+1}$. Each neuron σ_{n+q} can receive at most h_q, since there are h_q incoming synapses and the corresponding neuron sends, at most, one spike. Hence, at $\mathbb{C}_{2t+2}$, σ_{n+q} has, at most, h_q spikes. All of them are consumed by the corresponding rule and, since all the neurons which can send spikes to σ_{n+q} do not contain spikes at $\mathbb{C}_{2t+2}$, we conclude that, for all $q \in \{1, \ldots, n\}$, the neuron σ_{n+q} does not contain spikes in the configuration $\mathbb{C}_{2t+3}$. □

Remark 3. For all $q \in \{1, \ldots, n\}$, the neuron σ_q does not contain spikes in the configuration $\mathbb{C}_{2t}$.

Proof. The result holds in the initial configuration. For $\mathbb{C}_{2t}$ with $t > 0$ it suffices to check that, as claimed in Remark 2, for all $q \in \{1, \ldots, n\}$, the neuron σ_{n+q} does not contain spikes in the configuration $\mathbb{C}_{2t+1}$ and each σ_q receives at most one spike in each computation step from the corresponding σ_{n+q}. Therefore, in each configuration $\mathbb{C}_{2t+1}$, each neuron σ_q contains, at most, one spike. Since such spike is consumed by the rule $a \to \lambda$ and no new spike arrives, then the neuron σ_q does not contain spikes in the configuration $\mathbb{C}_{2t}$.

Before going on with the proof, it is necessary to formalize what means that the SN P system computes the immediate consequence operator T_{KB}. Given a deductive database *KB* on a set of variables $\{p_1, \ldots, p_n\}$, an interpretation on *KB* can be represented as a vector $I = (i_1, \ldots, i_n)$ with $i_j \in \{0, 1\}$ for $j \in \{1, \ldots, n\}$. Let us consider that such values $i_j \in \{0, 1\}$ represent the number of spikes placed in the corresponding neuron σ_{n+j} at the initial[5] configuration $\mathbb{C}_0$. We will consider that the computed output for such interpretation is encoded in the number of spikes in the neurons $\sigma_1, \ldots, \sigma_n$ in the configuration $\mathbb{C}_3$.

The main results of the theorem can be obtained from the following technical remark. □

Remark 4. Let $I = (i_1, \ldots, i_n)$ be an interpretation for *KB* and let $S = (s_1, \ldots, s_n)$ be a vector with the following properties. For all $j \in \{1, \ldots, n\}$

- If $i_j = 0$, then $s_j = 0$.
- If $i_j \neq 0$, then $s_j \in \{1, \ldots, h_j\}$

Let us suppose that at the configuration $\mathbb{C}_{2t}$ the neuron σ_{n+j} contains exactly s_j spikes. Then, the interpretation obtained by applying the immediate consequence operator T_{KB} to the interpretation *I*, $T_{KB}(I)$ is $(q_1, \ldots, q_n)$ where q_j, $j \in \{1, \ldots, n\}$, is the number of spikes of the neuron σ_j in the configuration $\mathbb{C}_{2t+3}$.

Proof. Firstly, let us consider $k \in \{1, \ldots, n\}$ and $T_{KB}(I)(p_k) = 1$. Let us prove that at the configuration $\mathbb{C}_{2t+3}$ there is exactly one spike in the neuron σ_k.

If $T_{KB}(I)(p_k) = 1$, then there exists at least one rule $r_l \equiv L_{d_1} \wedge \cdots \wedge L_{d_l} \to p_k$ in *KB* such that $I(L_{d_1} \wedge \cdots \wedge L_{d_l}) = 1$. □

[5] With a more complex design of the SN P system, it may be considered that these neurons do not contain spikes at the initial configuration and the vector $I = (i_1, \ldots, i_n)$ is provided as a spike train via an input neuron, but in this paper we have chosen a simpler design and focus on the computation of the immediate consequence operator. An analogous comment fits for the computed output.

Case 1: Let us consider that there is only one such rule r_l and the body of r_l is empty. By construction, the neuron σ_{2n+l} has only one incoming synapse from neuron σ_T; the neuron σ_{n+j} contains exactly s_j spikes, $j \in \{1,\dots,n\}$ and $s_j \in \{1,\dots,h_j\}$; and according to the previous remarks:

- In $\mathbb{C}_{2t}$ the neuron σ_T contains exactly one spike.
- For all $p \in \{1,\dots,k\}$ the neuron σ_{2n+p} does not contain spikes in the configuration $\mathbb{C}_{2t}$
- For all $q \in \{1,\dots,n\}$, the neuron σ_q does not contain spikes in the configuration $\mathbb{C}_{2t}$.

In these conditions, the corresponding rules in σ_T and σ_{n+k} are fired and in $\mathbb{C}_{2t+1}$, the neuron σ_{2n+k} contains one spike. In $\mathbb{C}_{2t+2}$, the neuron σ_{n+k} contains one spike and σ_k does not contain spikes. Finally, in the next step σ_{n+k} sends one spike to σ_k, so, in $\mathbb{C}_{2t+3}$, σ_k contain one spike.

Case 2: Let us now consider that there exists $r_l \equiv L_{d_1} \wedge \dots L_{d_l} \to p_k$ in KB such that $I(L_{d_1} \wedge \dots \wedge L_{d_l}) = 1$ and $d_l > 0$. We suppose that $I(L_{d_1} \wedge \dots \wedge L_{d_l}) = 1$ and, since KB is definite, this means that $I(L_{d_1}) = \dots = I(L_{d_l}) = 1$ and therefore, in $\mathbb{C}_{2t}$, the neuron σ_{n+d_j} contains s_{d_j} spikes, with $s_{d_j} \in \{1,\dots,h_{d_j}\}$. All these neurons fire the corresponding rule, and σ_{2n+k} has at $\mathbb{C}_{2t+1}$ exactly b_k spikes (since all the incoming synapses send the corresponding spike). The rule $a^{b_k} \to a$ is fired and in $\mathbb{C}_{2t+2}$ the neuron σ_{n+k} contains at least one spike. It may have more spikes depending on the existence of other rules with p_k in the head, but in any case, the number of spikes is between 1 and h_k. The corresponding rule fires and the neuron σ_k contains one spike in $\mathbb{C}_{2t+3}$.

Finally, we prove the statements claimed by the theorem:

(a) The SN P system computes the immediate consequence operator $T_{KB}(I)$.

Proof. It is directly obtained from *Remark 4*. Let us note that one of the possible vectors $S = (s_1,\dots,s_n)$ obtained from the interpretation I is exactly the same interpretation $I = (i_1,\dots,i_n)$. If we also consider the case when $t = 0$, we have proved that from the initial configuration $\mathbb{C}_0$ where i_k represents the number of spikes in the neuron σ_{n+k}, then the configuration $\mathbb{C}_3$ encodes $T_{KB}(I)$. □

(b) The SN P system computes the least model for KB in a finite number of steps.

Proof. Let us consider the empty interpretation as the initial one, i.e., $T_{KB} \uparrow 0 = I_\emptyset$. We will prove that

$$(\forall z \geq 1)\, T_{KB} \uparrow z = \mathbb{C}_{2z+1}[1,\dots,n]$$

where $\mathbb{C}_{2z+1}[1,\dots,n]$ is the vector whose components are the spikes on the neurons $\sigma_1,\dots,\sigma_n$ in the configuration $\mathbb{C}_{2z+1}$. We will prove it by induction. □

For $z = 1$, we have to prove that $T_{KB} \uparrow 1 = T_{KB}(T_{KB} \uparrow 0) = T_{KB}(I_\emptyset)$ is the vector whose components are the spikes on the neurons $\sigma_1,\dots,\sigma_n$ in the configuration $\mathbb{C}_3$. The result holds from *Remark 4* and it has been proved in the statement *(a)* of the theorem. By induction, let us consider now that $T_{KB} \uparrow z = \mathbb{C}_{2z+1}[1,\dots,n]$ holds. As previously stated, this means that in the previous configuration $\mathbb{C}_{2z}$ the spikes in the neurons $\sigma_{n+1},\dots,\sigma_{2n}$ can be represented as a vector $S = (s_1,\dots,s_n)$ be a vector with the properties claimed in *Remark 4*, namely, if the neuron σ_j has no spikes in $\mathbb{C}_{2z+1}$, then $s_j = 0$ and, if the neuron σ_j has spikes in $\mathbb{C}_{2z+1}$, then $s_j \in \{1,\dots,h_j\}$. Hence, according to *Remark 4*, three computational steps after $\mathbb{C}_{2z}$, $T_{KB}(\mathbb{C}_{2z+1}[1,\dots,n])$ is computed

$$T_{KB} \uparrow z+1 = T_{KB}(T_{KB} \uparrow z) = T_{KB}(\mathbb{C}_{2z+1}[1,\dots,n]) = \mathbb{C}_{2z+3}[1,\dots,n]$$

Finally, it is well-known that for a definite database KB, $T_{KB} \uparrow z \subseteq T_{KB} \uparrow z+1$ and, since the KB has a finite number of variables and a finite number of rules, then there exist $n \in \mathbb{N}$ such that $T_{KB}\uparrow n \subseteq T_{KB}\uparrow \omega$ and hence, $T_{KB}\uparrow n$ is a model for KB. □

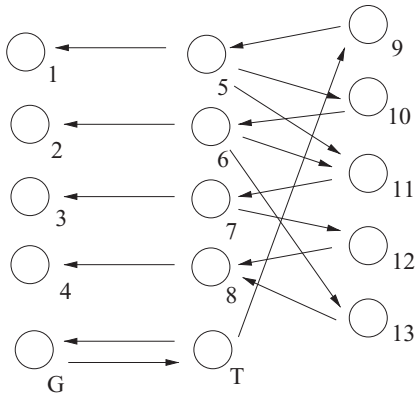

Fig. 1. Graphical representation of the synapses of the SN P system of Example 1.

Example 2. Let us consider the deductive database from Example 1. The SN P system associated with this KB and the interpretation $I_\emptyset$ is

$$\Pi = (O, \sigma_1, \sigma_2, \dots, \sigma_{13}, \sigma_G, \sigma_T, syn)$$

and its graphical representation is shown in Fig. 1. In such SN P system $O = \{a\}$,

$$\sigma_1 = (0, \{r_{1,1} \equiv a \to a\}) \quad \sigma_8 = \left(0, \begin{Bmatrix} r_{8,1} \equiv a \to a \\ r_{8,2} \equiv a^2 \to a \end{Bmatrix}\right)$$
$$\sigma_2 = (0, \{r_{2,1} \equiv a \to a\}) \quad \sigma_9 = (0, \{r_{9,1} \equiv a \to a\})$$
$$\sigma_3 = (0, \{r_{3,1} \equiv a \to a\}) \quad \sigma_{10} = (0, \{r_{10,1} \equiv a \to a\})$$
$$\sigma_4 = (0, \{r_{4,1} \equiv a \to a\}) \quad \sigma_{11} = \left(0, \begin{Bmatrix} r_{11,1} \equiv a \to \lambda \\ r_{11,2} \equiv a^2 \to a \end{Bmatrix}\right)$$
$$\sigma_5 = (0, \{r_{5,1} \equiv a \to a\}) \quad \sigma_{12} = (0, \{r_{12,1} \equiv a \to a\})$$
$$\sigma_6 = (0, \{r_{6,1} \equiv a \to a\}) \quad \sigma_{13} = (0, \{r_{13,1} \equiv a \to a\})$$
$$\sigma_7 = (0, \{r_{7,1} \equiv a \to a\})$$

$\sigma_G = (0, \{r_{G,1} \equiv a \to a\}$ and $\sigma_T = (0, \{r_{T,1} \equiv a \to a\}$ with the synapses

$$syn = \begin{Bmatrix} (5,1), (6,2), (7,3), (8,4), (5,10), (5,11), \\ (6,11), (6,13), (7,12), (9,5), (10,6), (11,7), \\ (12,8), (13,8), (G,T), (T,G), (T,9) \end{Bmatrix}$$

The first steps of the computation are shown in Table 1.

Let us remark that

$T_{KB} \uparrow 0 = \mathbb{C}_1[1,\dots,4] = (0,0,0,0)$
$T_{KB} \uparrow 1 = \mathbb{C}_3[1,\dots,4] = (1,0,0,0)$
$T_{KB} \uparrow 2 = \mathbb{C}_5[1,\dots,4] = (1,1,0,0)$
$T_{KB} \uparrow 3 = \mathbb{C}_7[1,\dots,4] = (1,1,1,1)$

4. Dealing with negation

The use of negation in reasoning systems is a complex task (see [28]) and a detailed discussion is out of the scope of this paper. In order to study the semantics of deductive databases with negation in the body of the rules, we will use the immediate consequence operator as defined in Def. 1. In this case, Theorem 1 still holds, but the immediate consequence operator is not monotonic (i.e., $I_1 \subseteq I_2$ does not implies $T_{KB}(I_1) \subseteq T_{KB}(I_2)$), as the following example illustrates[6].

Example 3. Let us consider a set of variables $V = \{p_1, p_2, p_3, p_4\}$ and the deductive database $KB = \{R_1, R_2, R_3, R_4\}$ with

$$R_1 \equiv \to p1 \quad R_2 \equiv p_1 \wedge \neg p_2 \to p_3$$

[6] In this section we will use -1 (instead of 0) to denote the value *false*. In this way, an interpretation will be represented by a vector $I = (i_1,\dots,i_n)$ with $i_j \in \{1,-1\}$.

Table 1
First steps in the computation of the SNPS in Example 2.

	σ_1	σ_2	σ_3	σ_4	σ_5	σ_6	σ_7	σ_8	σ_9	σ_{10}	σ_{11}	σ_{12}	σ_{13}	σ_G	σ_T
$\mathbb{C}_0$	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
$\mathbb{C}_1$	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0
$\mathbb{C}_2$	0	0	0	0	1	0	0	0	0	0	0	0	0	0	1
$\mathbb{C}_3$	1	0	0	0	0	0	0	0	1	1	1	0	0	1	0
$\mathbb{C}_4$	0	0	0	0	1	1	0	0	0	0	0	0	0	0	1
$\mathbb{C}_5$	1	1	0	0	0	0	0	0	1	1	2	0	1	1	0
$\mathbb{C}_6$	0	0	0	0	1	1	1	1	0	0	0	0	0	0	1
$\mathbb{C}_7$	1	1	1	1	0	0	0	0	1	1	1	1	1	1	0

$R_3 \equiv \neg p_3 \wedge \neg p_4 \rightarrow p_3 \qquad R_4 \equiv \neg p_1 \rightarrow p_4$

Let us consider the interpretation $T_{KB} \uparrow 0 = I_\emptyset = (-1,-1,-1,-1)$, then

$T_{KB} \uparrow 1 = T_{KB}(T_{KB} \uparrow 0) = (1,1,-1,1)$
$T_{KB} \uparrow 2 = T_{KB}(T_{KB} \uparrow 1) = (1,-1,-1,-1)$
$T_{KB} \uparrow 3 = T_{KB}(T_{KB} \uparrow 2) = (1,1,1,-1)$
$T_{KB} \uparrow 4 = T_{KB}(T_{KB} \uparrow 3) = (1,-1,-1,-1)$

For $k \geq 1$,

- $T_{KB} \uparrow 2k = T_{KB} \uparrow 2k+2$
- $T_{KB} \uparrow 2k+1 = T_{KB} \uparrow 2k+3$.

As we will show below, the use of anti-spikes allows to deal with negation. Theorem 4 claims that the immediate consequence operator can be implemented with SN P systems with anti-spikes. In such way, membrane computing also provides an efficient tool for handling with the semantics of deductive databases even if they include negative information.

Theorem 4. *For each deductive database KB on a set of propositional variables $\{p_1, -, p_n\}$ and an interpretation $I = (i_1, \ldots, i_n)$ an SN P system with anti-spikes can be constructed such that*

- *It computes the immediate consequence operator $T_{KB}(I)$.*
- *If I is the empty interpretation, it computes iteratively $T_{KB} \uparrow k$ for $k \geq 0$.*

Proof. Let us consider a deductive database KB and let $\{p_1, \ldots, p_n\}$ be the propositional variables and $\{r_1, \ldots, r_k\}$ be the rules of the KB. Given a variable p_i, we will denote as h_i the number of rules which have p_i in the head and given a rule r_j, we will denote as b_j the number of literals in its body. We will also denote by R^0 the number of rules with empty body. Let us define

$$b_{KB} = R^0 + \sum_{j=1}^{j=k} b_j$$

The SN P systems of degree $2n + k + 3 + b_{KB}$

$$\Pi_{KB} = (O, \sigma_1, \sigma_2, \ldots, \sigma_{2n+k+3+b_{KB}}, syn)$$

can be constructed as follows:

- $O = \{a\}$;
- $\sigma_j = (0, \{a \rightarrow \lambda, \overline{a} \rightarrow \lambda\})$ for $j \in \{1, \ldots, n\}$
- $\sigma_{n+j} = (i_j, R_j)$, $j \in \{1, \ldots, n\}$, where R_j is the set of rules $R_j = \{\overline{a} \rightarrow \overline{a}\} \cup \{a^k \rightarrow a | k \in \{1, \ldots, 2 \times h_j - 1\}\}$
- $\sigma_{2n+j} = (0, R_j)$, $j \in \{1, \ldots, k\}$, where $R_j$is the following rules
  - $R_j = \{a^{b_j} \rightarrow a^2\} \cup \{a^l \rightarrow \lambda | l \in \{1, \ldots, b_j - 1\}\}$ if $b_j > 0$
  - $R_j = \{a \rightarrow a^2\}$ if $b_j = 0$.

For a better understanding, the neurons σ_{2n+k+1}, σ_{2n+k+2} and σ_{2n+k+3} will be denoted by σ_T, σ_G and σ_H.

- $\sigma_G = (0, \{\overline{a} \rightarrow \overline{a}\})$
- $\sigma_T = (-1, \{\overline{a} \rightarrow \overline{a}\})$
- $\sigma_H = (0, \{\overline{a} \rightarrow \overline{a}\})$

As it will be shown below, the neurons $\sigma_{2n+k+3+1}, \ldots, \sigma_{2n+k+3+b_{KB}}$ have only an incoming synapse from a neuron σ_{n+i}, with $i \in \{1, \ldots, n\}$ (or σ_T), and only one outgoing synapse to a neuron σ_{2n+j}, with $j \in \{1, \ldots, k\}$. For a better understanding, we will denote by $\sigma_{i,j}$ the neuron from $\sigma_{2n+k+3+1}, \ldots, \sigma_{2n+k+3+b_{KB}}$ which has a synapse from a neuron σ_{n+i} and an outgoing synapse to a neuron σ_{2n+j}. The label of such neurons will be written as $\langle i, j \rangle$ and it will be said that such neurons are *double-labelled*.

- $\sigma_{T,j} = (0, \{\overline{a} \rightarrow a\})$ for $j \in \{1, \ldots, k\}$ and r_j is a rule with empty body.
- $\sigma_{i,j} = (0, \{a \rightarrow a, \overline{a} \rightarrow \overline{a}\})$ for $i \in \{1, \ldots, n\}$, $j \in \{1, \ldots, k\}$ and the variable p_i occurs positively in the body of the rule r_j.
- $\sigma_{i,j} = (0, \{a \rightarrow \overline{a}, \overline{a} \rightarrow a\})$ for $i \in \{1, \ldots, n\}$, $j \in \{1, \ldots, k\}$ and the variable p_i occurs negatively in the body of the rule r_j.

Finally, the synapses of the SN P systems are

- $syn = \{(n+i, i) | i \in \{1, \ldots, n\}\}$
  $\cup$ $\{(T, G), (G, H), (H, T)\}$ $\cup$ $\{(H, n+i) | i \in \{1, \ldots, n\}\}$
  $\cup$ $\{(T, \langle T, j \rangle) | r_j$ is a rule with empty body$\}$
  $\cup$ $\{(\langle T, j \rangle, 2n+j) | r_j$ is a rule with empty body$\}$
  $\cup$ $\left\{(n+i, \langle i, j \rangle) | \begin{array}{l} i \in \{1, \ldots, n\}, j \in \{1, \ldots, k\} \\ \text{and } l_i \text{ is a literal in the body of } r_j \end{array}\right\}$
  $\cup$ $\left\{(\langle i, j \rangle, 2n+j) | \begin{array}{l} i \in \{1, \ldots, n\}, j \in \{1, \ldots, k\} \\ \text{and } l_i \text{ is a literal in the body of } r_j \end{array}\right\}$
  $\cup$ $\left\{(2n+j, n+i) | \begin{array}{l} i \in \{1, \ldots, n\}, j \in \{1, \ldots, k\} \\ \text{and } p_i \text{ is the variable in the head of } r_j \end{array}\right\}$

Before going on with the proof, let us note that the construction of this SN P system is illustrated in the Example 4. As in Theorem 1, we will prove different remarks which will be useful in the proof of the theorem. □

Remark 1. For each $k \geq 0$

- In the configuration $\mathbb{C}_{3k}$, the neuron σ_T has only one anti-spike and the neurons σ_G and σ_H have no spikes nor anti-spikes.

- In the configuration $\mathbb{C}_{3k+1}$, the neuron σ_G has only one anti-spike and the neurons σ_T and σ_H have no spikes nor anti-spikes.
- In the configuration $\mathbb{C}_{3k+2}$, the neuron σ_H has only one anti-spike and the neurons σ_T and σ_G have no spikes nor anti-spikes.

Proof. This remark can be easily checked since σ_G has only an incoming synapse from σ_T, σ_H has only an incoming synapse from σ_G and σ_T has only an incoming synapse from σ_H. By construction, in $\mathbb{C}_0$, σ_T has one anti-spike in the initial configuration, σ_G and σ_H are empty and in the three neurons the unique rule is $\overline{a} \to \overline{a}$. So in, $\mathbb{C}_1$, σ_G has one anti-spike in the initial configuration, σ_T and σ_H are empty and in $\mathbb{C}_2$, σ_H has one anti-spike in the initial configuration, σ_G and σ_T are empty. This situation is cyclic and it is repeated each three computational steps. Let us also note that the neurons σ_{n+i} with $i \in \{1, \ldots, n\}$ has an incoming synapse from σ_H, so these neurons receive one anti-spike at $\mathbb{C}_{3k+3}$ from σ_H. □

Remark 2. Let $I = (i_1, \ldots, i_n)$ an interpretation for *KB* and let $S = (s_1, \ldots, s_n)$ be a vector with the following properties. For all $j \in \{1, \ldots, n\}$

- If $i_j = -1$, then $s_j = -1$.
- If $i_j = 1$, then $s_j \in \{1, \ldots, 2 \times h_j - 1\}$

Let us suppose that at the configuration $\mathbb{C}_{3t}$ the neuron σ_{n+j} contains exactly s_j spikes[7]. Then, the interpretation obtained by applying the immediate consequence operator T_{KB} to the interpretation I, $T_{KB}(I)$ is $(q_1, \ldots, q_n)$ where q_j, $j \in \{1, \ldots, n\}$ is the number of spikes of the neuron σ_j in the configuration $\mathbb{C}_{3t+4}$.

Proof. Firstly, let us consider $k \in \{1, \ldots, n\}$ and $T_{KB}(I)(p_k) = 1$. Let us prove that at the configuration $\mathbb{C}_{3t+4}$ there is exactly one spike in the neuron σ_k.

If $T_{KB}(I)(p_k) = 1$, then there exists at least one rule $r_l \equiv L_{d_1} \wedge \cdots \wedge L_{d_l} \to p_k$ in *KB* such that $I(L_{d_1} \wedge \cdots \wedge L_{d_l}) = 1$.

Case 1: Let us consider that there is only one such rule r_l and the body of r_l is empty. By construction, the neuron σ_{2n+l} has only one incoming synapse from neuron $\sigma_{\langle T, l\rangle}$ and $\sigma_{\langle T, l\rangle}$ has only one incoming synapse from σ_T. According to *Remark 1*, at $\mathbb{C}_{3k}$ the neuron σ_T has one anti-spike, so $\sigma_{\langle T, l\rangle}$ has one anti-spike at $\mathbb{C}_{3k+1}$; σ_{2n+l} has one spike at $\mathbb{C}_{3k+2}$; σ_{n+k} has one spike at $\mathbb{C}_{3k+3}$ and σ_k has one spike at $\mathbb{C}_{3k+4}$.

Case 2: Let us now consider that there exists $r_l \equiv L_{d_1} \wedge \cdots \wedge L_{d_l} \to p_k$ in *KB* such that $I(L_{d_1} \wedge \cdots \wedge L_{d_l}) = 1$ and $d_l > 0$. Without losing of generality, we can suppose that $L_{d_1} \wedge \cdots \wedge L_{d_l} = p_{d_1} \wedge \cdots \wedge p_{d_f} \wedge \neg p_{d_{f+1}} \wedge \cdots \wedge \neg p_{d_l}$ and therefore

- $I(p_{d_u}) = 1$ if $u \in \{1, \ldots, f\}$
- $I(p_{d_u}) = -1$ if $u \in \{f+1, \ldots, l\}$

According to the construction of the vector $S = (s_1, \ldots, s_n)$, at the configuration $\mathbb{C}_{3k}$

- σ_{n+d_u} has an amount of spikes between 1 and $2 \times h_{d_u} - 1$ if $u \in \{1, \ldots, f\}$
- σ_{n+d_u} has one anti-spike if $u \in \{f+1, \ldots, l\}$

By application of the corresponding rules, at $\mathbb{C}_{3k+1}$

- $\sigma_{\langle d_u, l\rangle}$ has one spike and the rule $a \to a$ can be applied , if $u \in \{1, \ldots, f\}$
- $\sigma_{\langle d_u, l\rangle}$ has one anti-spike and the rule $\overline{a} \to a$ can be applied, if $u \in \{f+1, \ldots, l\}$

At $\mathbb{C}_{3k+2}$, the neuron σ_{2n+l} has b_l spikes and sends 2 spikes to σ_{n+k}. At $\mathbb{C}_{3k+3}$, the neuron σ_{n+k} has received one anti-spike from σ_H, two spikes from σ_{2n+l}, and eventually, other spikes from neurons σ_{2n+p} if p_k is also the head of a rule r_p and the interpretation I satisfies the literals in their bodies of r_p. Bearing in mind the annihilation of the anti-spike with one of the spikes, at $\mathbb{C}_{3k+3}$, σ_{n+k} has an amount of spikes between 1 and $2 \times h_{p_k} - 1$. Let us remark that if none of the rules with p_k in its head has the literals in its body satisfied by I, then the neuron σ_{n+k} receives only one anti-spike from σ_H which is not annihilated. The spikes at $\mathbb{C}_{3k+3}$ fire the corresponding rule in σ_{n+k} and σ_k has one spike at $\mathbb{C}_{3k+4}$.

Let us consider now $k \in \{1, \ldots, n\}$ and $T_{KB}(I)(p_k) = -1$. In this case, all the rules $r_l \equiv L_{d_1} \wedge \cdots \wedge L_{d_l} \to p_k$ verifies $I(L_{d_1} \wedge \cdots \wedge L_{d_l}) = -1$. Firstly, let us note that the body of r_l cannot be empty. Let us consider one of such rules $r_l \equiv L_{d_1} \wedge \cdots \wedge L_{d_l} \to p_k$. Since $I(L_{d_1} \wedge \cdots \wedge L_{d_l}) = -1$, there exists L_{d_e} in the body of the rule such that $I(L_{d_e}) = -1$. Let us suppose that L_{d_e} is a negative literal (the reasoning in case of positive literals is analogous), $L_{d_e} = \neg p_{d_e}$ and $I(p_{d_e}) = 1$. By construction, in $\mathbb{C}_{3k}$ the neuron σ_{n+d_e} has s_{d_e} spikes with $s_{d_e} \in \{1, \ldots, 2 \times h_{d_e} - 1\}$. The corresponding rule is triggered and at $\mathbb{C}_{3k+1}$, the neuron $\sigma_{\langle d_e, l\rangle}$ has one spike. Since p_{d_e} occurs negatively in the body of r_l, the rule $a \to \overline{a}$ is applied and one anti-spike arrives to σ_{2n+l} at configuration $\mathbb{C}_{3k+2}$. Since σ_{2n+l} has b_l incoming synapses from neurons which send at most one spike, in $\mathbb{C}_{3k+2}$, the number of spikes in such neuron does not reach b_l and hence, no spike is sent in the next step. This happens in all neurons σ_{2n+l} and therefore, at $\mathbb{C}_{3k+3}$, σ_{n+k} only has one anti-spike from σ_H, the rule $\overline{a} \to \overline{a}$ is applied and σ_k has one anti-spike at $\mathbb{C}_{3k+4}$. □

Example 4. Let us consider the *KB* in the Example 3. The SNPS associated with this KB and the interpretation $I_\emptyset$ is

$$\Pi = (O, \sigma_1, \sigma_2, \ldots, \sigma_{12}, \sigma_G, \sigma_T, \sigma_H, \sigma_{T,1}, \sigma_{1,2}, \sigma_{2,2}, \sigma_{3,3}, \sigma_{4,3}, \sigma_{1,4}, syn)$$

and its graphical representation is shown in Fig. 2. In such SN P system $O = \{a\}$ and the neurons[8] are

$$\sigma_1 = (0, R_1)\sigma_2 = (0, R_2)\sigma_3 = (0, R_3)\sigma_4 = (0, R_4)$$
$$\sigma_5 = (-1, R_5)\sigma_6 = (-1, R_6)\sigma_7 = (-1, R_7)\sigma_8 = (-1, R_8)$$
$$\sigma_9 = (0, R_9)\sigma_{10} = (0, R_{10})\sigma_{11} = (0, R_{11})\sigma_{12} = (0, R_{12})$$
$$\sigma_G = (0, R_G)\sigma_T = (-1, R_T)\sigma_H = (0, R_H)$$
$$\sigma_{T,1} = (0, R_{T,1})\sigma_{1,2} = (0, R_{1,2})\sigma_{2,2} = (0, R_{2,2})\sigma_{3,3} = (0, R_{3,3})$$
$$\sigma_{4,3} = (0, R_{4,3})\sigma_{1,4} = (0, R_{1,4})$$

with the following sets of rules

$R_j = \{a \to a, \overline{a} \to \overline{a} | j \in \{1, \ldots, 8\}\}$
$R_j = \{\overline{a} \to \overline{a} | j \in \{T, G, H\}\}$
$R_9 = R_{12} = \{a \to a^2\}$
$R_{10} = R_{11} = \{a \to \lambda, a^2 \to a^2\}$
$R_{T,1} = \{\overline{a} \to a\}$
$R_{1,2} = \{a \to a, \overline{a} \to \overline{a}\}$
$R_{2,2} = R_{3,3} = R_{4,3} = R_{1,4} = \{a \to \overline{a}, \overline{a} \to a\}$ with the synapses

$$syn = \left\{ \begin{array}{l} (5,1), (6,2), (7,3), (8,4), \\ (T,G), (G,H), (H,T), (H,5), (H,6), (H,7), (H,8), \\ (T, \langle T,1\rangle), (5, \langle 1,2\rangle), (5, \langle 1,4\rangle), (6, \langle 2,2\rangle), (7, \langle 3,3\rangle), (8, \langle 4,3\rangle) \\ (\langle T,1\rangle, 9), (\langle 1,2\rangle, 10), (\langle 2,2\rangle, 10), (\langle 3,3\rangle, 11), (\langle 4,3\rangle, 11), (\langle 1,4\rangle, 12) \\ (9,5), (10,7), (11,6), (12,8) \end{array} \right\}$$

Let us remark that

$$T_{KB} \uparrow 0 = \mathbb{C}_0[1, \ldots, 4] = (-1, -1, -1, -1)$$
$$T_{KB} \uparrow 1 = \mathbb{C}_4[1, \ldots, 4] = (1, 1, -1, 1)$$

[7] If $s_j = -1$, then σ_{n+j} will contain one anti-spike.

[8] We will represent by t that there are t spikes in the neuron and by $-t$ that there are t anti-spikes

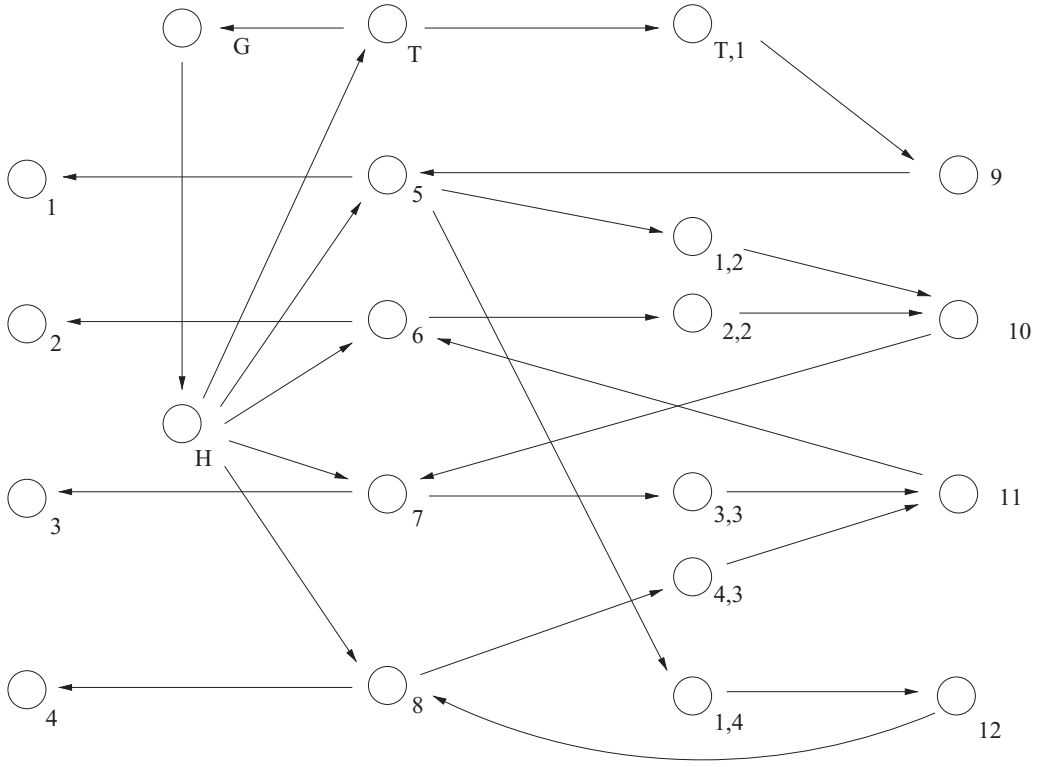

Fig. 2. Graphical representation of the synapses of the SN P system of Example 3.

Table 2
First steps in the computation of the SNPS in Example 4.

	σ_1	σ_2	σ_3	σ_4	σ_5	σ_6	σ_7	σ_8	σ_9	σ_{10}	σ_{11}	σ_{12}	σ_G	σ_T	σ_H	$\sigma_{T,1}$	$\sigma_{1,2}$	$\sigma_{2,2}$	$\sigma_{3,3}$	$\sigma_{4,3}$	$\sigma_{1,4}$
$\mathbb{C}_0$	0	0	0	0	−1	−1	−1	−1	0	0	0	0	0	−1	0	0	0	0	0	0	0
$\mathbb{C}_1$	−1	−1	−1	−1	0	0	0	0	0	0	0	0	−1	0	0	−1	−1	−1	−1	−1	−1
$\mathbb{C}_2$	0	0	0	0	0	0	0	0	1	0	2	−1	0	0	−1	0	0	0	0	0	0
$\mathbb{C}_3$	0	0	0	0	1	1	−1	1	0	0	0	0	0	−1	0	0	0	0	0	0	0
$\mathbb{C}_4$	1	1	−1	1	0	0	0	0	0	0	0	0	−1	0	0	−1	1	1	−1	1	1
$\mathbb{C}_5$	0	0	0	0	0	0	0	0	1	0	0	0	0	0	−1	0	0	0	0	0	0
$\mathbb{C}_6$	0	0	0	0	1	−1	−1	−1	0	0	0	0	0	−1	0	0	0	0	0	0	0
$\mathbb{C}_7$	1	−1	−1	−1	0	0	0	0	0	0	0	0	−1	0	0	−1	1	−1	−1	−1	1
$\mathbb{C}_8$	0	0	0	0	0	0	0	0	1	2	2	−1	0	0	−1	0	0	0	0	0	0
$\mathbb{C}_9$	0	0	0	0	1	1	1	−1	0	0	0	0	0	−1	0	0	0	0	0	0	0
$\mathbb{C}_{10}$	<1	1	1	−1	0	0	0	0	0	0	0	0	−1	0	0	−1	1	1	1	−1	1
$\mathbb{C}_{11}$	0	0	0	0	0	0	0	0	1	0	0	−1	0	0	−1	0	0	0	0	0	0
$\mathbb{C}_{12}$	0	0	0	0	1	−1	−1	−1	0	0	0	0	0	−1	0	0	0	0	0	0	0
$\mathbb{C}_{13}$	1	−1	−1	−1	0	0	0	0	0	0	0	0	−1	0	0	−1	1	−1	−1	−1	1

$T_{KB} \uparrow 2 = \mathbb{C}_7[1, \ldots, 4] = (1, -1, -1, -1)$
$T_{KB} \uparrow 3 = \mathbb{C}_{10}[1, \ldots, 4] = (1, 1, 1, -1)$
$T_{KB} \uparrow 4 = \mathbb{C}_{13}[1, \ldots, 4] = (1, -1, -1, -1)$

The first steps of the computation are shown in Table 2.

5. Conclusions

Biological neurons have a binary behavior depending on a threshold. If the threshold is reached, the neuron is triggered and it sends a spike to the next neurons. If it is not reached, nothing is sent. This binary behavior can be exploited in order to design connectionist systems which are able to deal with

two-valued logic-based reasoning systems. The current efforts for bridging both methods try to one approach to the other one by the adaptation of some of the features. In this sense, SN P system is a bio-inspired model whose features seem to fit to the target of mixing both approaches.

In this paper, we have proved that SN P systems are able to deal with the semantics of deductive databases even when such systems use negation. Namely, we have proved that the immediate consequence operator can be iteratively computed with such devices by using an appropriate representation. In the case of deductive databases without negation, it is suffices to characterize the fix point semantics, since the immediate consequence operator is monotonic and the least Herbrand model can be computed by a SN P system in a finite number of steps.

In the case of databases which involve negation, the semantics is much more complex. In this paper we have shown that the immediate consequence operator can be computed with SN P systems, but it is not enough for dealing with a complex semantics that involves, e.g., supported, stable and perfect models (see [27]).

Considering the semantics of deductive databases in this bio-inspired model open a new door to further research possibilities. In particular, it should be explored if all the new SN P system variants can offer improvements to the study of database semantics. From a more general point of view, the possibility of using other P system models (not only SN P systems) can be explored. Finally, the development of new efficient membrane computing simulators, based on GPU architectures [29] can be also considered. In such case, the theoretical point of view shown in this paper, where the semantics of databases is explore, would have a practical counterpart.

References

[1] S. Bader, P. Hitzler, Dimensions of neural-symbolic integration - a structured survey, in: S. Artemov, H. Barringer, A.S. d'Avila Garcez, L. Lamb, J. Woods (Eds.), We Will Show Them: Essays in Honour of Dov Gabbay, 1, King's College Publications, 2005, pp. 167–194.

[2] T.R. Besold, K.-U. Kühnberger, Towards integrated neural-symbolic systems for human-level AI: two research programs helping to bridge the gaps, Biol. Inspired Cogn. Archit. 14 (2015) 97–110.

[3] B. Hammer, P. Hitzler (Eds.), Perspectives of neural-symbolic integration, Studies in Computational Intelligence, 77, Springer, 2007.

[4] M.A. Gutiérrez-Naranjo, V. Rogojin, Deductive databases and P systems, Comput. Sci. J. Moldova 12 (1) (2004) 80–88.

[5] S. Ivanov, A. Alhazov, V. Rogojin, M.A. Gutiérrez-Naranjo, Forward and backward chaining with P systems, Int. J. Nat. Comput. Res. 2 (2) (2011) 56–66.

[6] W. Gerstner, W. Kistler, Spiking Neuron Models: Single Neurons, Populations, Plasticity, Cambridge University Press, 2002.

[7] K. Doets, From logic to logic programming, Foundations of computing, MIT Press, Cambridge (Mass.), 1994.

[8] L. Pan, Gh. Păun, Spiking neural P systems with anti-spikes, Int. J. Comput. Commun. Control IV (3) (2009) 273–282.

[9] V.P. Metta, K. Krithivasan, D. Garg, Computability of spiking neural P systems with anti-spikes, New Math. Nat. Comput. (NMNC) 08 (03) (2012) 283–295.

[10] T. Song, Y. Jiang, X. Shi, X. Zeng, Small universal spiking neural P systems with anti-spikes, J. Comput. Theor. Nanosci. 10 (4) (2013) 999–1006, doi:10.1166/jctn.2013.2799.

[11] G. Tan, T. Song, Z. Chen, X. Zeng, Spiking neural P systems with anti-spikes and without annihilating priority working in a 'flip-flop' way, Int. J. Comput. Sci. Math. 4 (2) (2013) 152–162, doi:10.1504/IJCSM.2013.055208.

[12] A. Alhazov, B. Aman, R. Freund, P systems with anti-matter, in: M. Gheorghe, G. Rozenberg, A. Salomaa, P. Sosík, C. Zandron (Eds.), Proceedings of the 15th International Conference on Membrane Computing - CMC 2014, Lecture Notes in Computer Science, Revised Selected Papers, vol. 8961, Springer, Prague, Czech Republic, 2014, pp. 66–85.

[13] A. Alhazov, B. Aman, R. Freund, Gh. Păun, Matter and anti-matter in membrane systems, in: H. Jürgensen, J. Karhumäki, A. Okhotin (Eds.), Proceedings of the 16th International Workshop on Descriptional Complexity of Formal Systems DCFS, Lecture Notes in Computer Science, vol. 8614, Springer, Turku, Finland, 2014, pp. 65–76.

[14] D. Díaz-Pernil, F. Peña-Cantillana, A. Alhazov, R. Freund, M.A. Gutiérrez–Naranjo, Antimatter as a frontier of tractability in membrane computing, Fundam. Inform. 134 (2014) 83–96.

[15] X. Zeng, X. Zhang, T. Song, L. Pan, Spiking neural P systems with thresholds, Neural Comput. 26 (7) (2014) 1340–1361.

[16] T. Song, L. Pan, Spiking neural P systems with request rules, Neurocomputing 193 (2016) 193–200.

[17] F.G.C. Cabarle, H.N. Adorna, M.J. Pérez-Jiménez, T. Song, Spiking neural P systems with structural plasticity, Neural Comput. Appl. 26 (8) (2015) 1905–1917.

[18] T. Wu, Z. Zhang, Gh. Păun, L. Pan, Cell-like spiking neural P systems, Theor. Comput. Sci. 623 (2016) 180–189.

[19] T. Wang, G. Zhang, M.J. Pérez-Jiménez, Fuzzy membrane computing: theory and applications, Int. J. Comput. Commun. Control 10 (6) (2015) 904–935.

[20] T. Wang, G. Zhang, J. Zhao, Z. He, J. Wang, M.J. Pérez-Jiménez, Fault diagnosis of electric power systems based on fuzzy reasoning spiking neural P systems, IEEE Trans. Power Syst. 30 (3) (2015) 1182–1194.

[21] G. Zhang, H. Rong, F. Neri, M.J. Pérez-Jiménez, An optimization spiking neural P system for approximately solving combinatorial optimization problems, Int. J. Neural Syst. 24 (5) (2014).

[22] M.H. van Emden, R.A. Kowalski, The semantics of predicate logic as a programming language, J. ACM 23 (4) (1976) 733–742.

[23] Gh. Păun, G. Rozenberg, A. Salomaa (Eds.), The Oxford Handbook of Membrane Computing, Oxford University Press, Oxford, England, 2010.

[24] K.R. Apt, Logic programming, in: Handbook of Theoretical Computer Science, Volume B: Formal Models and Sematics (B), The MIT Press, 1990, pp. 493–574.

[25] J. Lloyd, Foundations of Logic Programming, Symbolic computation: Artificial intelligence, Springer, 1987.

[26] M. Ionescu, Gh. Păun, T. Yokomori, Spiking neural P systems, Fundam. Inform. 71 (2–3) (2006) 279–308.

[27] P. Hitzler, A.K. Seda, Mathematical Aspects of Logic Programming Semantics, Chapman and Hall /CRC studies in informatics series, CRC Press, 2011.

[28] K.R. Apt, R.N. Bol, Special issue: ten years of logic programming logic programming and negation: a survey, J. Logic Program. 19 (1994) 9–71.

[29] F.G.C. Cabarle, H.N. Adorna, M.A.M. del Amor, M.J. Pérez-Jiménez, Improving GPU simulations of spiking neural P systems, Rom. J. Inf. Sci. Technol. 15 (1) (2012) 5–20.