Semantics of Deductive Databases in a Membrane Computing Connectionist Model

Daniel Díaz-Pernil[1], Miguel A. Gutiérrez-Naranjo[2]

[1]Research Group on Computational Topology and Applied Mathematics
Department of Applied Mathematics - University of Sevilla, 41012, Spain
sbdani@us.es

[2]Research Group on Natural Computing
Department of Computer Science and Artificial Intelligence
University of Sevilla, 41012, Spain
magutier@us.es

Summary. 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 found 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 and the immediate consequence operator can be implemented in the spiking neural P systems model.

1 Introduction

Two of the most well-known paradigms for implementing automated reasoning in machines are, on the one hand, the family of connectionist systems, inspired in the network of biological neurons in a human brain and, on the other hand, logic-based systems, able to represent and reason with well-structured symbolic data. The integration of both paradigms is a vivid area in artificial intelligence (see, e.g., [2, 3, 8]).

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 [7, 11]). 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, 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 points 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 a 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 suffice for dealing with the semantics of propositional logic systems.

The main result of this paper 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. 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 [5]. It is well-know that 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 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. Finally, some conclusions 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 [13] for a comprehensive presentation of the former and [1, 4, 12] for the latter.

2.1 Spiking Neural P Systems

SN P systems were introduced in [10] with the aim of incorporating in membrane computing ideas specific to 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[1]

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

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 $E/a^p \to a$, where E is a regular expression over a and $p \geq 1$ is an integer number;
      (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.

The 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 $E/a^p \to a$ 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 one spike which is 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 $E/a^p \to a$ of type (1) has $E = a^p$, then we will write it in the simplified form $a^p \to a$. In each time unit, if a neuron σ_i can use one of its rules, then a rule from R_i must be used. Since two firing rules, $E_1/a^{p_1} \to a$ and $E_2/a^{p_2} \to a$ can have $L(E_1) \cap L(E_2) \neq \emptyset$, 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.

The j-th 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.

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

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.

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 rule is a formula $B_1 \wedge \cdots \wedge B_m \to A$ where $m \geq 0$, $A, B_1, \ldots B_m$ are variables. The variable A is called the *head* of the rule and the conjunction of variables $B_1 \wedge \cdots \wedge B_m$ is the *body* of the rule. If $m = 0$, it is said that the body of the rule is empty. A finite set of rules KB is a deductive database. 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, $I(B_1 \wedge \cdots \wedge B_m) = \min\{I(B_1), \ldots, I(B_m)\}$ and for a rule[2]

$$I(B_1 \wedge \cdots \wedge B_m \to A) = \begin{cases} 0 \text{ if } I(B_1 \wedge \cdots \wedge B_m) = 1 \text{ and } I(A) = 0 \\ 1 \text{ otherwise} \end{cases}$$

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 [5].

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*

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

such that, for $k \in \{1, \ldots, n\}$, $T_{KB}(I)(p_k) = 1$ if there exists a rule $B_1 \wedge \cdots \wedge B_m \to p_k$ in KB such that $I(B_1 \wedge \cdots \wedge B_m) = 1$; otherwise, $T_{KB}(I)(p_k) = 0$.

The importance of the immediate consequence operator is shown in the following proposition (see [9]).

[2] Let us remark that, from the definition, if $m = 0$, $I(B_1 \wedge \cdots \wedge B_m) = 1$ and, hence, for a rule with an empty body, we have $I(\to A) = 1$ if and only if $I(A) = 1$.

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.

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 with the least model of a deductive database (see [12]).

Theorem 2. *Let KB be a deductive database. The following results 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 knowledge base KB on the set of variables $\Gamma = \{p_1, p_2, p_3, p_4\}$

$$\begin{aligned} R_1 &\equiv \to p_1 \\ R_2 &\equiv p_1 \to p_2 \\ R_3 &\equiv p_1 \wedge p_2 \to p_3 \\ R_4 &\equiv p_3 \to p_4 \\ R_5 &\equiv p_2 \to p_4 \end{aligned}$$

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 Th. 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

$$\begin{aligned} &T_{KB} \uparrow 0 = I_\emptyset = (0, 0, 0, 0) \\ &T_{KB} \uparrow 1 = T_{KB}(T_{KB} \uparrow 0) = (1, 0, 0, 0) \\ &T_{KB} \uparrow 2 = T_{KB}(T_{KB} \uparrow 1) = (1, 1, 0, 0) \\ &T_{KB} \uparrow 3 = T_{KB}(T_{KB} \uparrow 2) = (1, 1, 1, 1) \end{aligned}$$

In this case $T_{KB} \uparrow 3$ is a fix point for the immediate consequence operator and a model for the 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 claims that SN P systems can compute the immediate consequence operator and hence, the least model of a deductive database.

Theorem 3. *Given a deductive database KB and an interpretation I, a 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 knowledge database KB, let $\{p_1,\dots,p_n\}$ be the propositional variables and $\{r_1,\dots,r_k\}$ 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 $2n+k+3$

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

can be constructed as follows:

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

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

$$\cup\ \{(G,T),(T,G)\}$$
$$\cup \left\{ \begin{array}{l} (T, 2n+j) \mid j \in \{1,\dots,k\} \\ \qquad\qquad \text{and } r_j \text{ is a rule with empty body} \end{array} \right\}$$

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 $p \in \{1,\dots,k\}$ 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_{n+q} does not contain spikes in the configuration $\mathbb{C}_{2t+1}$*

Proof. In the initial configuration $\mathbb{C}_0$, for all $p \in \{1,\dots,k\}$, 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 $p \in \{1,\dots,k\}$, the neuron σ_{2n+p} does not contain spikes and for all $q \in \{1,\dots,n\}$, 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 $j \in \{1,\dots,k\}$, the neuron σ_{2n+j} does not contain spikes. We focus now on the neurons σ_{n+q} with $q \in \{1,\dots,n\}$. 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 $q \in \{1,\dots,n\}$, the neuron σ_{n+q} does not contain spikes in the configuration $\mathbb{C}_{2t+3}$.

Remark 3. For all $q \in \{1, \dots, n\}$, 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 $q \in \{1, \dots, n\}$, 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 $\{p_1, \dots, p_n\}$, an interpretation on KB can be represented as a vector $I = (i_1, \dots, i_n)$ with $i_j \in \{0, 1\}$ for $j \in \{1, \dots, n\}$. 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[3] configuration $\mathbb{C}_0$. We will consider that the computed output for such interpretation is encoded in the number of spikes in the neurons $\sigma_1, \dots, \sigma_n$ in the configuration $\mathbb{C}_3$.

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

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

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

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 $(q_1, \dots, q_n)$ where q_j, $j \in \{1, \dots, n\}$, is the number of spikes of the neuron σ_j in the configuration $\mathbb{C}_{2t+3}$.
Proof. Firstly, let us consider $k \in \{1, \dots, n\}$ 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 $r_l \equiv B_{d_1} \wedge \dots \wedge B_{d_l} \to p_k$ in KB such that $I(B_{d_1} \wedge \dots \wedge B_{d_l}) = 1$.

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 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.

[3] 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 $I = (i_1, \dots, i_n)$ 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.

- 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 $r_l \equiv B_{d_1} \wedge \dots B_{d_l} \rightarrow p_k$ in KB such that $I(B_{d_1} \wedge \dots \wedge B_{d_l}) = 1$ and $d_l > 0$. We suppose that $I(B_{d_1} \wedge \dots \wedge B_{d_l}) = 1$ and this means that $I(B_{d_1}) = \dots = I(B_{d_l}) = 1$ 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 $a^{b_k} \rightarrow a$ 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

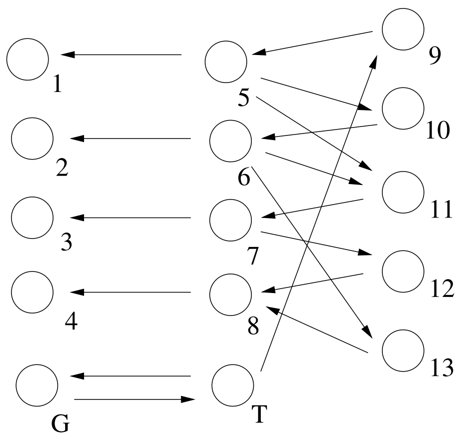

Fig. 1. Graphical representation of the synapses of the SN P system of Example 1.

neuron σ_j has spikes in $\mathbb{C}_{2z+1}$, then $s_j \in \{1, \ldots, h_j\}$. Hence, according to *Remark 4*, three computational steps after $\mathbb{C}_{2z}$, $T_{KB}(\mathbb{C}_{2z+1}[1, \ldots, n])$ is computed

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

Finally, it is well-known that for a 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. $\square$

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

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

where $O = \{a\}$,

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

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

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

Let us consider the first steps of the computation

$Conf.$	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

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

4 Conclusions

Biological neurons have a binary behaviour 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 behaviour can be exploited in order to design connectionist systems which are able to deal with two-valued logic-based reasoning systems. In this paper, we have proved that SN P systems are able to deal with the semantics of deductive databases. Namely, we have proved that the immediate consequence operator can be iteratively computed with such devices by using an appropriate representation. This pioneer work opens a door for future bridges between SN P systems and logic-based reasoning systems.

References

1. Apt, K.R.: Logic Programming. In: Handbook of Theoretical Computer Science, Volume B: Formal Models and Sematics (B), pp. 493–574. The MIT Press (1990)
2. Bader, S., Hitzler, P.: Dimensions of neural-symbolic integration - a structured survey. In: Artemov, S., Barringer, H., d'Avila Garcez, A.S., Lamb, L., Woods, J. (eds.) We Will Show Them: Essays in Honour of Dov Gabbay, vol. 1, pp. 167–194. King's College Publications (2005)

3. Besold, T.R., Kühnberger, K.U.: Towards integrated neural-symbolic systems for human-level AI: Two research programs helping to bridge the gaps. Biologically Inspired Cognitive Architectures 14, 97 – 110 (2015)
4. Doets, K.: From logic to logic programming. Foundations of computing, MIT Press, Cambridge (Mass.) (1994)
5. van Emden, M.H., Kowalski, R.A.: The semantics of predicate logic as a programming language. Journal of the ACM 23(4), 733–742 (1976)
6. Gerstner, W., Kistler, W.: Spiking neuron models: single neurons, populations, plasticity. Cambridge University Press (2002)
7. Gutiérrez-Naranjo, M.A., Rogojin, V.: Deductive databases and P systems. Computer Science Journal of Moldova 12(1), 80–88 (2004)
8. Hammer, B., Hitzler, P. (eds.): Perspectives of Neural-Symbolic Integration, Studies in Computational Intelligence, vol. 77. Springer (2007)
9. Hitzler, P., Seda, A.K.: Mathematical Aspects of Logic Programming Semantics. Chapman and Hall / CRC studies in informatics series, CRC Press (2011)
10. Ionescu, M., Păun, Gh., Yokomori, T.: Spiking neural P systems. Fundamenta Informaticae 71(2-3), 279–308 (2006)
11. Ivanov, S., Alhazov, A., Rogojin, V., Gutiérrez-Naranjo, M.A.: Forward and backward chaining with P systems. International Journal on Natural Computing Research 2(2), 56–66 (2011)
12. Lloyd, J.: Foundations of Logic Programming. Symbolic computation: Artificial intelligence, Springer (1987)
13. Păun, Gh., Rozenberg, G., Salomaa, A. (eds.): The Oxford Handbook of Membrane Computing. Oxford University Press, Oxford, England (2010)