Logic Negation with Spiking Neural P Systems

Daniel Rodríguez-Chavarría[1] · Miguel A. Gutiérrez-Naranjo[1] · Joaquín Borrego-Díaz[1]

Abstract
Nowadays, the success of neural networks as reasoning systems is doubtless. Nonetheless, one of the drawbacks of such reasoning systems is that they work as black-boxes and the acquired knowledge is not human readable. In this paper, we present a new step in order to close the gap between connectionist and logic based reasoning systems. We show that two of the most used inference rules for obtaining negative information in rule based reasoning systems, the so-called *Closed World Assumption* and *Negation as Finite Failure* can be characterized by means of *spiking neural P systems*, a formal model of the third generation of neural networks born in the framework of membrane computing.

Keywords P systems · Logic negation · Membrane computing

1 Introduction

Neural networks are nowadays one of the most promising tools in computer sciences. They have been successfully applied to many real-world domains and the number of application fields is continuously increasing [15]. Beyond this doubtless success, one of the main drawbacks of such systems is that they work as black-boxes, i.e., the learned knowledge through the training process is not human-readable. Learning process in neural networks consists basically of optimizing parameters (usually a huge amount of them) guided by some type of gradient-based method and the resulting model is usually far from having *semantic* sense for a human researcher. In fact, the problem of *explainability* is becoming a new research frontier in artificial intelligence systems, even beyond machine learning [1,14]. Due to this lack of readability, new studies about the integration of neural network models (the so-called

✉ Joaquín Borrego-Díaz
jborrego@us.es

Daniel Rodríguez-Chavarría
danrodcha@gmail.com

Miguel A. Gutiérrez-Naranjo
magutier@us.es

1 Department of Computer Science and Artificial Intelligence, University of Seville, Seville, Spain

connectionist systems) and logic-based systems [4,6,17,27,31,35] can shed a new light on the future development of both research areas.

In this context, the computational framework known as *spiking neural P systems* [20,21] (SN P systems, for short) provides a formal framework for the integration of both disciplines: on the one hand, they use *spikes* (electrical impulses) as discrete units of information as in logic-based methods and, on the other hand, their models consist of graphs where the information flows among nodes as in standard neural network architectures. SN P systems belong to the third generation of neural network models [26], the so-called *integrate-and-fire* spiking neuron models [13]. The integration of logic and neural networks via spikes takes advantage from an important biological fact: all the spikes inside a biological brain look alike. By using this feature, a computational binary code can be considered: sending one spike is considered as a sign for *true* and if no spikes are sent, then it is considered as a sign of *false*. These features were exploited in [10] where SN P systems were used to bridge bioinspired connectionist systems with the semantics of reasoning systems based on logic.

The main contribution of this paper is to add new elements for dealing with negation in the interplay of bridging neural networks and logic. Bridges between both areas can help to enrich each other.

In this study, we go on with the approach started in [10] by focusing on logic negation. Using negation in computational logic systems is often a hard task [3] since pure derivative reasoning systems have no way to derive negative information from a set of facts and rules. This problem is solved by adding to the reasoning system a new inference rule which allows to *derive* negative information. In this paper, two of such inference rules are studied in the framework of SN P systems: *Closed World Assumption (CWA)* and *Negation as Finite Failure* (NFF).

Loosely speaking, given a deductive database KB, CWA considers *false* all the atomic sentences which are not logical consequence of KB. The attempts to check whether a sentence is a logical consequence of KB or not can fall into an infinite loop, and therefore, a different *effective* rule is needed. Such rule is NFF. It considers *false* a sentence if all the attempts to prove it fail (according to some protocol). This is a quite restrictive definition of negation, but it is on the basis of many reasoning systems used in Artificial Intelligence as the Logic Programming paradigms [23] or planning systems [24].

The recent development of SN P systems involves SN P systems with communication on request [28], applications of fuzzy SN P systems [19,37], *cell-like* SN P systems [38], SN P systems *with request rules* [33], SN P systems *with structural plasticity* [8], SN P systems *with thresholds* [40] or SN P systems with rules on synapses [34] among many others. Some applications of Membrane Computing to real-life problems (including SN P systems) can be found in the literature (see, e.g., [16,41]).

The paper is organized as follows: Sect. 2 recalls some basics on SN P systems and the procedural and declarative semantics of deductive databases. The following section shows how the inference rules CWA and NFF can be characterized via SN P systems. Finally, some conclusions are showed in Sect. 4.

2 Preliminaries

In this section, we recall briefly some basic concepts on SN P systems and the declarative and procedural semantics of deductive databases.

2.1 Spiking Neural P Systems

SN P systems were introduced as a model of computational devices inspired by the flow of information between neurons. This model keeps the basic idea of encoding and processing the information via binary events used other spiking neuron models (see, e.g., Ch. 3 in [11]). Such devices are distributed and work in a parallel way. They consist of a directed graph with *neurons* placed on the nodes. Each neuron contains a number of copies of an object called the *spike* and it may contain several *firing* and *forgetting* rules. Firing rules send *spikes* to other neurons. Forgetting rules allow to remove spikes from a neuron. In order to decide if a rule is applicable, the contents of the neuron is checked against a regular set associated with the rule. In each time unit, if several rules can be applied in a neuron, one of them, non-deterministically chosen, must be used. In this way, rules are used in a sequential way in each neuron, but neurons function in parallel with each other. As usual, a global clock with discrete time steps is assumed and the functioning of the whole system is synchronized.

Formally, an SN P system of the degree $m \geq 1$ is a construct[3]

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

where $O = \{a\}$ is the singleton alphabet (a is called *spike*) and $\sigma_1, \sigma_2, \ldots, \sigma_m$ are *neurons*. Each neuron is a pair $\sigma_i = (n_i, R_i)$, $1 \leq i \leq m$, where:

1. $n_i \geq 0$ is the *initial number of spikes* contained in σ_i;
2. R_i is a finite set of *rules* of the following two kinds:

   (1) *firing* rules of type $E/a^p \to a^q$, where E is a regular expression over the spike a and $p, q \geq 1$ are integer numbers ;
   (2) *forgetting* rules of type $a^s \to \lambda$, with s an integer number such that $s \geq 1$;

The set of synapses (edges) syn is a set of pairs $syn \subseteq \{1, 2, \ldots, m\} \times \{1, 2, \ldots, m\}$, verifying that (i, i) does not belong to sys for any $i \in \{1, \ldots, m\}$.

Let us suppose that the neuron σ_i contains k spikes and a rule $E/a^p \to a^q$ with $k \geq p$. Let $L(E)$ be the language generated by the regular expression E. In these conditions, if a^k belongs to $L(E)$, then the rule $E/a^p \to a^q$ can be applied. The application is performed by sending q spikes to all neurons σ_j such that $(i, j) \in syn$ and deleting p spikes from σ_i (thus only $k - p$ spikes remain into the neuron). In this case, it is said that the neuron is *fired*.

Let us now suppose that the neuron σ_i contains exactly s spikes and the forgetting rule $a^s \to \lambda$. In such case, the rule can be fired by removing all the s spikes from σ_i. If the regular expression E in a firing rule $E/a^p \to a^q$ is equal to a^p, then the firing rule can be expressed as $a^p \to a^q$. In each time unit, if a neuron σ_i can use one of its rules, then one of them must be used. If two or more rules can be applied in a neuron, then only one of them is non-deterministically chosen regardless of its type. The SN P system evolves according to these type of rules and reaches different configurations which are represented as vectors $\mathbb{C}_j = (t_1^j, \ldots, t_m^j)$ where t_k^j stands for the number of spikes at the neuron σ_k in the

[3] In the literature, many different SN P systems models have been presented. In this paper, a simple model is considered.

$j-th$ configuration. It will be useful to consider only the first components of a configuration. Let us define $\mathbb{C}_j[1,\ldots,n] = (t_1^j,\ldots,t_n^j)$ as the n-dimensional vector composed by the n first components of $\mathbb{C}_j$. The initial configuration is the vector with the number of spikes in each neuron at the beginning of the computation $\mathbb{C}_0 = (n_1, n_2, \ldots, n_m)$. By using the rules described above, transitions between configurations can be defined. A sequence of transitions which starts at the initial configuration is called a computation.

2.2 Declarative Semantics of Rule-Based Deductive Databases

Reasoning based on rules can be formalized according to different approaches. In this paper, propositional logic is considered for representing knowledge. Different formal representation systems where the number of available *terms* is finite (as those based on pairs attribute-value of first order logic representations without function symbols) can be bijectively mapped onto propositional logic systems and therefore the presented approach covers many real-life cases.

Next, some basics on propositional logic are provided. Let $\{p_1,\ldots,p_n\}$ be a set of variables. A literal is a variable or a negated variable. An expression $L_1 \wedge \cdots \wedge L_n \rightarrow A$, where $n \geq 0$, A is a variable and $L_1,\ldots L_n$ are literals is a rule. The conjunction $L_1 \wedge \cdots \wedge L_n$ is the *body* and the variable A is the *head* of the rule. If $n = 0$, the body of the rule is empty. A finite set of rules KB is called a deductive database. A mapping $I : \{p_1,\ldots,p_n\} \rightarrow \{0,1\}$ is an *interpretation*, which is usually represented as a vector $(i_1,\ldots,i_n)$ with $I(p_k) = i_k \in \{0,1\}$ for $k \in \{1,\ldots,n\}$. The interpretations $I_\downarrow$ and $I_\uparrow$ are defined as $I_\downarrow = (0,\ldots,0)$ and $I_\uparrow = (1,\ldots,1)$. The set of all the interpretations on a set of n variables denote 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 $I_2(p_k) = 1$; $I_1 \cup I_2$ and $I_1 \cap I_2$ are new interpretations such that $(I_1 \cup I_2)(p_k) = max\{I_1(p_k), I_2(p_k)\}$ and $(I_1 \cap I_2)(p_k) = min\{I_1(p_k), I_2(p_k)\}$ for $k \in \{1,\ldots,n\}$. An operator $S : 2^n \rightarrow 2^n$ is *monotone* if for all interpretations I_1 and I_2, if $I_1 \subseteq I_2$, then $S(I_1) \subseteq S(I_2)$. An interpretation I is extended in the following 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]

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

Next, we recall the notions of *model* and *F-model* of a deductive database. The concept of F-model is one of the key ideas in this paper. To the best of our knowledge, it was firstly presented in [39]. The definition used in this paper is adapted from the original one.

Definition 1 Let I be an interpretation for a deductive database KB

- I is a *model* if for all rule $L_1 \wedge \cdots \wedge L_n \rightarrow A$ verifying that $\min_{i\in\{1,\ldots,n\}} I(L_i) = 1$, the equality $I(A) = 1$ holds; in other words, if $I(R) = 1$ for all rule $R \in KB$.
- I is a *F-model* if for all rule $L_1 \wedge \cdots \wedge L_n \rightarrow A$ verifying that $I(A) = 1$, the equality $\max_{i\in\{1,\ldots,n\}} I(L_i) = 1$ holds.

Next example illustrates these concepts.

Example 1 Let KB be the deductive database on the set $\{p_1, p_2, p_3, p_4, p_5, p_6, p_7, p_8, p_9\}$ defined as follows:

$$\begin{array}{llll} R_1 \equiv \quad \rightarrow p_1 & R_4 \equiv p_3 \wedge p_6 \rightarrow p_4 & R_7 \equiv p_6 \rightarrow p_5 & R_9 \equiv p_7 \rightarrow p_2 \\ R_2 \equiv p_1 \rightarrow p_2 & R_5 \equiv p_4 \rightarrow p_5 & R_8 \equiv p_8 \rightarrow p_9 & R_{10} \equiv p_9 \rightarrow p_8 \\ R_3 \equiv p_1 \wedge p_2 \rightarrow p_3 & R_6 \equiv p_7 \rightarrow p_6 & & \end{array}$$

[4] Let us remark that, according to the definition, $I(\rightarrow A) = 1$ if and only if $I(A) = 1$.

then, the interpretation represented by the vector $I_1 = (1, 1, 1, 0, 0, 0, 0, 0, 0)$ is a model of KB.

In this case, the rules verifying

$$\min_{i \in \{1,\dots,n\}} I(L_i) = 1$$

are R_1, R_2 and R_3 and all of them satisfies $I(A) = 1$.

The interpretations $I_2 = (0, 0, 0, 1, 1, 1, 1, 0, 0)$ and $I_3 = (0, 0, 0, 1, 1, 1, 1, 1, 1)$ are *F-models* of KB. In both cases, for all rule $L_1 \wedge \cdots \wedge L_n \to A$ verifying that $I(A) = 1$, the equality

$$\max_{i \in \{1,\dots,n\}} I(L_i) = 1$$

holds. If we consider the interpretation I_2, then $I_2(p_4) = I_2(p_5) = I_2(p_6) = I_2(p_7) = 1$ and all the rules with variables p_4, p_5, p_6 or p_7 in their heads (R_4, R_5 and R_6) verify that there exists a variable q in the body of the rule with $I(q) = 1$. The case of I_3 is analogous.

In a certain sense, *F-models* keep a *duality* with respect the concept of models. If I_A and I_B are models, then $I_A \cap I_B$ is also a model and if they are *F-models*, then $I_A \cup I_B$ is also a *F-model* [39]. Next, the definition of *Failure Operator* F_{KB} of a deductive database FK is recalled. It can be seen as a dual of the Kowalski's immediate consequence operator T_{KB} [36].

Definition 2 Let $\{p_1, \dots, p_n\}$ be a set of variables and KB a deductive database on it. The failure operator of KB is the mapping $F_{KB} : 2^n \to 2^n$ such that for $I \in 2^n$, $F_{KB}(I)$ is an interpretation $F_{KB}(I) : \{p_1, \dots, p_n\} \to \{0, 1\}$ where $F_{KB}(I)(p_k) = 1$ if for each rule $L_1 \wedge \cdots \wedge L_n \to p_k$ in KB, $\max_{i \in \{1,\dots,n\}} I(L_i) = 1$ holds (for $k \in \{1, \dots, n\}$); otherwise, $F_{KB}(I)(p_k) = 0$.

Let us remark that, according to the definition, if there is no rule in KB with p_k in its head, then $F_{KB}(I)(p_k) = 1$, for all interpretation I.

Kowalski's operator T_{KB} allows to characterize the models of KB (see, e.g. [18]) in the sense that an interpretation I is a model of KB if and only if $T_{KB}(I) \subseteq I$. Proposition 1 shows that the failure operator F_{KB} also allows to characterize the *F-models*. The intuition behind the failure operator is to capture the idea of *immediate failure* in a similar way that the operator T_{KB} captures the idea of *immediate consequence*.

Proposition 1 *[39] Let KB be a deductive database.*

- *An interpretation I_F is an F-model of KB if and only if $I_F \subset F_{KB}(I)$.*
- *The failure operator F_{KB} is monotone, over the set of the interpretations of KB.*

Since the image of an interpretation by the F_{KB} operator is an interpretation itself, it can be iteratively applied.

Definition 3 Let KB be a deductive database and F_{KB} its failure operator.

(a) The mapping $F_{KB} \downarrow : \mathbb{N} \to 2^n$ is defined as follows: $F_{KB} \downarrow 0 = I_{\downarrow}$ and $F_{KB} \downarrow n = F_{KB}\,(F_{KB} \downarrow (n-1))$ if $n > 0$. In the limit, it is also considered

$$F_{KB} \downarrow \omega = \bigcup_{k \geq 0} F_{KB} \downarrow k$$

(b) The mapping $F_{KB} \uparrow : \mathbb{N} \to 2^n$ is defined as follows: $F_{KB} \uparrow 0 = I_\uparrow$ and $F_{KB} \uparrow n = F_{KB}(F_{KB} \uparrow (n-1))$ if $n > 0$. In the limit, it is also considered

$$F_{KB} \uparrow \omega = \bigcap_{k \geq 0} F_{KB} \uparrow k$$

Bearing in mind that the number of rules and variables in a deductive database are finite, the next Proposition is immediate.

Proposition 2 *Let KB be a deductive database and F_{KB} its failure operator.*

(a) *There exists $n \in \mathbb{N}$ such that $F_{KB} \uparrow n = F_{KB} \uparrow k$ for all $k \geq n$.*
(b) *There exists $n \in \mathbb{N}$ such that $F_{KB} \downarrow n = F_{KB} \downarrow k$ for all $k \geq n$.*

Let us remark that Proposition 2 implies that the number of computation steps for reaching the above limits is finite.

Example 2 Let us consider again the database KB used in Example 1 and its failure operator. The following interpretations are obtained.

$$\begin{array}{l} F_{KB} \downarrow 0 = I_\downarrow = (0,0,0,0,0,0,0,0,0) \\ F_{KB} \downarrow 1 = F_{KB}(F_{KB} \downarrow 0) = (0,0,0,0,0,0,1,0,0) \\ F_{KB} \downarrow 2 = F_{KB}(F_{KB} \downarrow 1) = (0,0,0,0,0,1,1,0,0) \\ F_{KB} \downarrow 3 = F_{KB}(F_{KB} \downarrow 2) = (0,0,0,1,0,1,1,0,0) \\ F_{KB} \downarrow 4 = F_{KB}(F_{KB} \downarrow 3) = (0,0,0,1,1,1,1,0,0) \end{array}$$

Since $F_{KB} \downarrow 5 = F_{KB} \downarrow 4$, then $F_{KB} \downarrow \omega = (0,0,0,1,1,1,1,0,0)$

$$\begin{array}{l} F_{KB} \uparrow 0 = I_\uparrow = (1,1,1,1,1,1,1,1,1) \\ F_{KB} \uparrow 1 = F_{KB}(F_{KB} \uparrow 0) = (0,1,1,1,1,1,1,1,1) \\ F_{KB} \uparrow 2 = F_{KB}(F_{KB} \uparrow 1) = (0,0,1,1,1,1,1,1,1) \\ F_{KB} \uparrow 3 = F_{KB}(F_{KB} \uparrow 2) = (0,0,0,1,1,1,1,1,1) \end{array}$$

Since $F_{KB} \uparrow 4 = F_{KB} \uparrow 3$, then $F_{KB} \uparrow \omega = (0,0,0,1,1,1,1,1,1)$

2.3 Procedural Semantics of Rule-based Deductive Databases

The main result of this paper is the characterization of the set of variables obtained by the non-monotonic inference rules CWA and NFF via the procedural behaviour of an SN P system. Let us recall that a rule R is said *non-monotonic* if there exist two interpretations I_1 and I_2 with $I_1 \subseteq I_2$ such that $R(I_1) \not\subseteq R(I_2)$.

For the sake of completeness, some basics of the procedural semantics of deductive databases are recalled.[5] A *goal* is a formula $\neg B_1 \vee \cdots \vee \neg B_n$ where B_i are atoms. As usual, the goal $\neg B_1 \vee \cdots \vee \neg B_n$ will be represented as $B_1, \ldots, B_n \to$. We also consider the empty clause $\square$ as a goal. Given a goal $G \equiv A_1, \ldots, A_{k-1}, A_k, A_{k+1}, \ldots, A_n \to$ and a rule $R \equiv B_1, \ldots, B_m \to A_k$, the goal

$$G' \equiv A_1, \ldots, A_{k-1}, B_1, \ldots, B_m, A_{k+1}, \ldots, A_n \to$$

is called the *resolvent* of R and G. It is also said that G' is derived from R and G. Let KB be a deductive database and G a goal. An SLD-derivation of $KB \cup \{G\}$ consists of a (finite or infinite) sequence $G_0, G_1, \ldots$ of goals with $G_0 = G$ and a sequence of rules $R_1, R_2, \ldots$

[5] A detailed description can be found in [25].

from KB such that G_{i+1} is derived from R_{i+1} and G_i. It is said that $KB \cup \{G\}$ has a finite failed tree if all the SLD-derivations are finite and none of them has the empty clause $\square$ as the last goal of the derivation. The *failure set* of KB is the set of all variables A for which there exists a finite failed tree for $KB \cup \{A \rightarrow\}$.

Example 3 Let KB be the same deductive database from Example 1. Next, some SLD derivations are calculated:

$KB \cup \{p_3 \rightarrow\}$

Rule used	Goals
	$p_3 \rightarrow$
R_3	$p_1, p_2 \rightarrow$
R_2	$p_1 \rightarrow$
R_1	$\square$

$KB \cup \{p_9 \rightarrow\}$

Rule used	Goals
	$p_9 \rightarrow$
R_8	$p_8 \rightarrow$
R_{10}	$p_9 \rightarrow$
R_8	$p_8 \rightarrow$
$\vdots$	$\vdots$

$KB \cup \{p_6 \rightarrow\}$

Rule used	Goals
	$p_6 \rightarrow$
R_6	$p_7 \rightarrow$

As shown above, the goals $p_3 \rightarrow$ and $p_9 \rightarrow$ do not have finite failed trees whereas $p_6 \rightarrow$ does. Finally, it is easy to check that the *failure set* of KB is $\{p_4, p_5, p_6, p_7\}$.

We give now a brief recall of the formal definition of both inference rules. A detailed motivation of such rules is out of the scope of this paper. The first inference rule for deriving such negative information considered in this paper is the CWA [30]: *If A is not a logical consequence of KB, then infer* $\neg A$. The second inference rule called NFF [9]: *If $KB \cup \{A \rightarrow\}$ has a finite failed tree, then infer* $\neg A$, or, in other words, if A belongs to the failure set, then infer $\neg A$.

The next Theorem is an adaptation of the Th.13.6 in [25] and provides a procedural characterization of the variables in the failure set of a deductive database KB. It settles the equality of two sets defined with two different approaches: on the one hand, the set of variables such that all the SLD-derivations fail after a finite number of steps and, on the other hand, the set of variables mapped onto 1 by the interpretation $F_{KB} \downarrow \omega$, obtained by the iteration of the failure operator.

Theorem 1 *Let KB be a database on a set of variables $\{p_1, \ldots, p_n\}$ and F_{KB} its failure operator. For all k in $\{1, \ldots, n\}$, p_k is in the failure set of KB if and only if $F_{KB} \downarrow \omega(p_k) = 1$*

By using this theorem, we will prove in the next section that the finite failure set of a database KB can be characterized by means of SN P systems. The next Theorem relates the CWA with the failure operator. A proof of it is out of the scope of this paper. Details can be found in [25,39].

Theorem 2 *Let KB be a database on a set of variables $\{p_1, \ldots, p_n\}$ and F_{KB} its failure operator. For all k in $\{1, \ldots, n\}$, p_k is not a logical consequence of KB if and only if $F_{KB} \uparrow \omega(p_k) = 1$*

3 Logic Negation with SN P Systems

In this section, we bridge the neural model of SN P systems with the inference rules CWA and NFF. The main theorems in this paper claim that the result of both inference rules can be computed in a finite number of steps by an appropriate SN P system. The proof of such results is achieved via some lemmas which link the properties of the SN P systems with the semantics of the deductive databases.

Theorem 3 *Let us consider a set of variables* $\{p_1, \ldots, p_n\}$ *and a deductive database* KB *on it. Let* I *be an interpretation on such set of variables. Let* F_{KB} *be the failure operator of* KB*. An SN P system can be constructed from* KB *such that*

$$F_{KB}(I) = \mathbb{C}_3[1, \ldots, n]$$

where $\mathbb{C}_3$ *is the configuration of the SN P system after the third step of computation.*

Theorem 3 claims the equality of two n-dimensional vectors. The first one is the vector which represents the interpretation $F_{KB}(I) : \{p_1, \ldots, p_n\} \to \{0, 1\}$ obtained by means of the application of the operator F_{KB} to the interpretation I. The second one is the vector which represents the number of spikes in the neurons $\sigma_1, \ldots, \sigma_n$ in the corresponding SN P system in the third configuration. The proof is constructive and it builds explicitly the SN P system.

Proof Let KB be a deductive database such that $\{r_1, \ldots r_k\}$ and $\{p_1, \ldots p_n\}$ are the set of rules and the set of variables. Given a variable p_i, the number of rules which have p_i in the head is denoted by h_i and given a rule r_j, the number of variables in its body is denoted by b_j. The SN P system of degree $2n + k + 2$.

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

□

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

• $i_j = I(p_j)$ if $h_j = 0$
• $i_j = I(p_j) \cdot h_j$ if $h_j > 0$

and R_j is the set of h_j rules

• $R_j = \{a \longrightarrow a\}$ if $h_j = 0$
• $R_j = \{a^{h_j} \longrightarrow a\}$ if $h_j > 0$

- $\sigma_{2n+j} = (0, R_j)$, $j \in \{1, \ldots k\}$, where R_j is one of the following set of rules

• $R_j = \emptyset$ if $b_j = 0$.
• $R_j = \{a^l \to a \,|\, l \in \{1, \ldots, b_j\}\}$ if $b_j > 0$

For the sake of simplicity, the neurons σ_{2n+k+1} and σ_{2n+k+2} will be denoted by σ_G and σ_T, respectively.

- $\sigma_G = (1, \{a \to a\})$
- $\sigma_T = (0, \{a \to a\})$
- $syn = \{(n + i, i) \,|\, i \in \{1, \ldots, n\}\}$

$$\cup \left\{ \begin{array}{l} (n + i, 2n + j) \,|\, i \in \{1, \ldots, n\}, j \in \{1, \ldots, k\} \\ \qquad \text{and } p_i \text{ is a variable in the body of } r_j \end{array} \right\}$$
$$\cup \left\{ \begin{array}{l} (2n + j, n + i) \,|\, i \in \{1, \ldots, n\}, j \in \{1, \ldots, k\} \\ \qquad \text{and } p_i \text{ is the variable in the head of } r_j \end{array} \right\}$$
$$\cup \{(G, T), (T, G)\}$$
$$\cup \left\{ \begin{array}{l} (T, n + i) \,|\, i \in \{1, \ldots, n\} \\ \qquad \text{and } p_i \text{ is a variable such that } h_i = 0 \end{array} \right\}$$

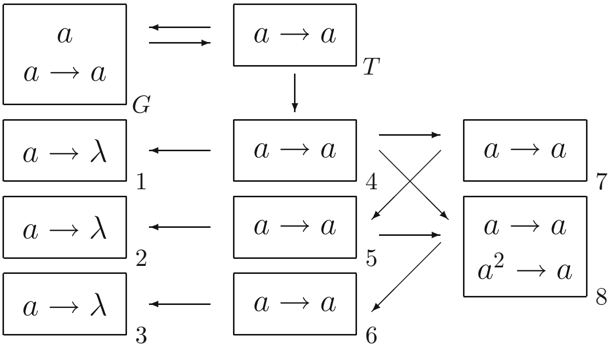

Fig. 1 Initial configuration of the SN P system from Example 4

The proof will be split into four lemmas. Although the result of the theorem only concerns to the third configuration, the lemmas are proved in general.

Before going on with the proof, the building of the SN P system is illustrated with the following toy example.

Example 4 Let us consider the set of three variables $\{p_1, p_2, p_3\}$, a database on it with two rules

$$r_1 \equiv p_1 \rightarrow p_2 \qquad\qquad r_2 \equiv p_1, p_2 \rightarrow p_3$$

and the interpretation $I_\downarrow = (0, 0, 0)$. According to the notation, in this case $n = 3$, $k = 2$, $h_1 = 0, h_2 = 1, h_3 = 1, b_1 = 1$ and $b_2 = 2$. The associated SN P system has $2n+k+2 = 10$ neurons and its initial configuration is depicted in Fig. 1. Since the interpretation is $I_\downarrow$, there is only one spike in this first configuration $\mathbb{C}_0$. It is placed on the neuron σ_G. The rule $a \rightarrow a$ in σ_G is applied and the unique spike in the configuration $\mathbb{C}_1$ is placed in σ_T. Since σ_T has two outgoing synapses, the application of the rule $a \rightarrow a$ in it produces two spikes. Therefore, in the configuration $\mathbb{C}_2$ there are two spikes in the SN P system: one of them in σ_G and the other one in σ_4. The application of the rule $a \rightarrow a$ in σ_G sends one spike to σ_T, so in this neuron there is a spike in the configuration $\mathbb{C}_3$. Since σ_4 has three outgoing synapses, the application of the rule $a \rightarrow a$ sends one spike to σ_1, another to σ_7 and a third one to σ_8. To sum up, in the configuration $\mathbb{C}_3$, there are four spikes in the system, each of them in the neurons σ_T, σ_1, σ_7 and σ_8. According to the theorem, in order to know $F_{KB}(I_\downarrow)$ it suffices to check the number of spikes in the neurons σ_1, σ_2 and σ_3 at the configuration $\mathbb{C}_3$. In other words, this SN P system has computed $F_{KB}(I_\downarrow) = (1, 0, 0)$.

Next, the following lemmas will be proved.

Lemma 1 *For all $t \geq 0$, in the $2t$-th configuration $\mathbb{C}_{2t}$ the neuron σ_G has exactly one spike and σ_T is empty.*

Proof The result will be proved by induction. The lemma holds in the initial configuration. The inductive assumption is that in the configuration $\mathbb{C}_{2t}$, the neuron σ_T does not contain spikes and the neuron σ_G contains exactly one spike. There is only one incoming synapse in σ_G which comes from σ_T, and vice versa. Furthermore, the unique rule that occurs in each neuron is $a \rightarrow a$ so, in $\mathbb{C}_{2t+1}$, σ_G has consumed its spike and does not contain any spike, and the neuron σ_T contains exactly one spike. For the same reasoning, in $\mathbb{C}_{2t+2}$, σ_T has consumed its spike and the neuron σ_G contains exactly one spike. □

Lemma 2 *For all $t \geq 0$ the following results hold:*

- *For all $p \in \{1, \ldots, k\}$ the neuron σ_{2n+p} is empty in the configuration $\mathbb{C}_{2t}$.*
- *For all $q \in \{1, \ldots, n\}$, the neuron σ_{n+q} is empty in the configuration $\mathbb{C}_{2t+1}$.*

Proof In the configuration $\mathbb{C}_0$, for all $p \in \{1, \ldots, k\}$, the neuron σ_{2n+p} is empty and, for all $q \in \{1, \ldots, n\}$, each neuron σ_{n+q} contains, at most, h_q spikes. These spikes are consumed by the application of the rule $a^{h_j} \to a$ (or $a \to a$). Finally, as every neuron with synapse to σ_{n+q} is empty at $\mathbb{C}_0$, it follows that in the configuration $\mathbb{C}_1$, all the neurons σ_{n+q} with $q \in \{1, \ldots, n\}$ are empty.

As induction hypothesis, we state that in $\mathbb{C}_{2t}$, for all $p \in \{1, \ldots, k\}$, the neuron σ_{2n+p} is empty and for all $q \in \{1, \ldots, n\}$, the neuron σ_{n+q} is empty in the configuration $\mathbb{C}_{2t+1}$. As defined before, the number of incoming synapses in each neuron σ_j is b_j. The neurons which are the origin of such synapses send (at most) one spike in one computational step, so in $\mathbb{C}_{2t+1}$, the number of spikes in the neuron σ_{2n+p} is, at most, b_p. The corresponding rules ($a^{h_j} \to a$ or $a \to a$) consume all these spikes so, at $\mathbb{C}_{2t+1}$, all the neurons with outgoing synapses to σ_{2n+p} are empty. In the next step, at most, b_p spikes contained in the neurons σ_{2n+p} were consumed by the corresponding rules. Therefore, we conclude that at $\mathbb{C}_{2t+2}$, for all $p \in \{1, \ldots, k\}$, the neurons σ_{2n+p} are empty.

Focusing on the second part of the lemma, as induction hypothesis we state that the neurons σ_{n+q} with $q \in \{1, \ldots, n\}$ are empty in the configuration $\mathbb{C}_{2t+1}$. Each neuron σ_{n+q} can receive at most h_q if $h_q > 1$ and 1 if $h_q = 0$, since there are h_q or 1 incoming synapses and each of these sends, at most, one spike. Hence, at $\mathbb{C}_{2t+2}$, σ_{n+q} has, at most, h_q if $h_q > 1$ and 1 if $h_q = 0$, spikes. All of them are consumed by the corresponding rule and, since all the neurons which can send spikes to σ_{n+q} are empty at $\mathbb{C}_{2t+2}$, we conclude that, for all $q \in \{1, \ldots, n\}$, the neuron σ_{n+q} is empty in the configuration $\mathbb{C}_{2t+3}$. □

Lemma 3 *For all $q \in \{1, \ldots, n\}$, the neuron σ_q is empty in the configuration $\mathbb{C}_{2t}$.*

Proof In the first configuration ($\mathbb{C}_0$) the lemma holds. For $\mathbb{C}_{2t}$ with $t > 0$ it is enough to check that, as stated in Lemma 2, for all $q \in \{1, \ldots, n\}$, the neuron σ_{n+q} is empty 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 is empty in the configuration $\mathbb{C}_{2t}$. □

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

If in the configuration $\mathbb{C}_{2t}$ the neuron σ_{n+j} contains exactly s_j spikes for all $j \in \{1, \ldots, n\}$ then the interpretation obtained by applying the failure operator F_{KB} to the interpretation I, $F_{KB}(I)$, is $(q_1, \ldots, q_n)$ where q_j, $j \in \{1, \ldots, n\}$, corresponds with the number of spikes contained in the neuron σ_j in the configuration $\mathbb{C}_{2t+3}$.

Proof Let us consider $m \in \{1, \ldots, n\}$ and $F_{KB}(I)(p_m) = 1$. We will prove that in the configuration $\mathbb{C}_{2t+3}$ there is exactly one spike in the neuron σ_m.

If $F_{KB}(I)(p_m) = 1$, then for each rule $r_l \equiv L_{d_1} \wedge \cdots \wedge L_{d_l} \rightarrow p_m$ with p_m in the head, there exists $j \in \{1, \ldots, n\}$ such that $I(L_j) = 1$.

Case 1: Let us consider that there is no such rule r_l. By construction, the neuron σ_{n+m} has only one incoming synapse from neuron σ_T; and according to the previous lemmas:

- In $\mathbb{C}_{2t}$ the neuron σ_G contains exactly one spike.
- For all $q \in \{1, \ldots, n\}$, the neuron σ_{n+q} is empty in the configuration $\mathbb{C}_{2t+1}$
- For all $q \in \{1, \ldots, n\}$, the neuron σ_q is empty in the configuration $\mathbb{C}_{2t}$.

In these conditions, the corresponding rules in σ_G and σ_{n+m} are fired and in $\mathbb{C}_{2t+1}$, the neuron σ_T contains one spike. In $\mathbb{C}_{2t+2}$, the neuron σ_{n+m} contains one spike and σ_m is empty. Finally, in the next step σ_{n+m} sends one spike to σ_m, so, in $\mathbb{C}_{2t+3}$, σ_m contains one spike.

Case 2: Let us now consider that there are h_m rules (with $h_m > 0$) such that $r_l \equiv L_{d_1} \wedge \ldots L_{d_l} \rightarrow p_m$ and for each one there exists $j_l \in \{1, \ldots, n\}$ such that $I(L_{j_l}) = 1$. This means that, in $\mathbb{C}_{2t}$, every neuron σ_{n+j_l} contains 1 or h_{j_l} spikes, as appropriate. All these neurons fire the corresponding rules, and, in $\mathbb{C}_{2t+1}$, every σ_{2n+l} has at least one spike. So one rule from $\{a^q \rightarrow a \mid q \in \{1, \ldots, b_l\}\}$ is fired in every σ_{2n+l} and in $\mathbb{C}_{2t+2}$ the neuron σ_{n+m} contains exactly h_m spikes. The corresponding rule fires and the neuron σ_m contains one spike in $\mathbb{C}_{2t+3}$. □

Finally, the proof of the Theorem 3 is provided. It is immediate from Lemma 4.

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

Theorem 3 is the basis of the two main results of this paper, which are proved in the following theorems.

Theorem 4 *Let KB be a deductive database on the set of variables $\{p_1, \ldots, p_k\}$. An SN P system can be constructed from KB such that it computes the inference rule CWA on the database KB.*

Proof According to Theorem 2, $\neg p_k$ is inferred from KB by using the inference rule CWA if and only is $F_{KB} \uparrow \omega(p_k) = 1$ and from Theorem 3, an SN P system can be constructed from KB such that $F_{KB}(I) = \mathbb{C}_3[1, \ldots, n]$ where $\mathbb{C}_3$ is the configuration of the SN P system after the third step of computation. By combining both results, we will prove

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

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

For $z = 1$, we will see that $F_{KB} \uparrow 1 = F_{KB}(F_{KB} \uparrow 0) = F_{KB}(I_\uparrow)$ is the vector whose components are the spikes on the neurons $\sigma_1, \ldots, \sigma_n$ in the configuration $\mathbb{C}_3$. The result holds from *Lemma 4* in the proof of Theorem 3. By induction, let us consider now that $F_{KB} \uparrow z = \mathbb{C}_{2z+1}[1, \ldots, n]$ holds. As previously stated, this means that in the previous configuration $\mathbb{C}_{2z}$ the spikes in the neurons $\sigma_{n+1}, \ldots, \sigma_{2n}$ can be represented as a vector $S = (s_1, \ldots, s_n)$ with the properties claimed in *Lemma 4*, namely, if the neuron σ_j has no spikes in $\mathbb{C}_{2z+1}$, then $s_j = 0$ or $s_j \in \{0, \ldots, h_j - 1\}$, as corresponds, and, if the neuron σ_j has one spike in $\mathbb{C}_{2z+1}$, then $s_j = 1$ or $s_j = h_j$, as appropriate. Hence, according to *Lemma 4*, three computational steps after $\mathbb{C}_{2z}$, $F_{KB}(\mathbb{C}_{2z+1}[1, \ldots, n])$ is computed:

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

From Corollary 2, there exists $m \in \mathbb{N}$ such that $F_{KB} \uparrow m = F_{KB} \uparrow k$ for all $k \geq m$, i.e. $F_{KB} \uparrow m = F_{KB} \uparrow \omega$. So the vector whose components are the spikes on the neurons $\sigma_1, \ldots, \sigma_n$ in the configuration $\mathbb{C}_{2m+1}$ is the result obtained by applying the inference rule CWA. □

The previous proof can be adapted to prove that the SN P systems also can characterize the inference rule *Negation of Failure Set*.

Theorem 5 *Let KB be a deductive database on the set of variables $\{p_1, \ldots, p_k\}$. An SN P system can be constructed from KB such that it computes the inference rule Negation of Failure Set on the database KB.*

Proof According to Theorem 1, $\neg p_k$ is inferred from KB by using the inference rule *Negation of Failure Set* if and only is $F_{KB} \downarrow \omega(p_k) = 1$ and from Theorem 3, an SN P system can be constructed from KB such that $F_{KB}(I) = \mathbb{C}_3[1, \ldots, n]$ where $\mathbb{C}_3$ is the configuration of the SN P system after the third step of computation. By combining both results, we will prove

$$(\forall z \geq 1)\ F_{KB} \downarrow z = \mathbb{C}_{2z+1}[1, \ldots, n]$$

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

For $z = 1$, we have to prove that $F_{KB} \downarrow 1 = F_{KB}(F_{KB} \downarrow 0) = F_{KB}(I_\downarrow)$ is the vector whose components are the spikes on the neurons $\sigma_1, \ldots, \sigma_n$ in the configuration $\mathbb{C}_3$. The result holds from *Lemma 4* in the proof of Theorem 3. By induction, let us consider now that $F_{KB} \downarrow z = \mathbb{C}_{2z+1}[1, \ldots, n]$ holds. As previously stated, this means that in the previous configuration $\mathbb{C}_{2z}$ the spikes in the neurons $\sigma_{n+1}, \ldots, \sigma_{2n}$ can be represented as a vector $S = (s_1, \ldots, s_n)$ with the properties claimed in *Lemma 4*, namely, if the neuron σ_j has no spikes in $\mathbb{C}_{2z+1}$, then $s_j = 0$ or $s_j \in \{0, \ldots, h_j - 1\}$, as corresponds, and, if the neuron σ_j has one spike in $\mathbb{C}_{2z+1}$, then $s_j = 1$ or $s_j = h_j$, as appropriate. Hence, according to *Lemma 4*, three computation steps after $\mathbb{C}_{2z}$, $F_{KB}(\mathbb{C}_{2z+1}[1, \ldots, n])$ is computed:

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

From Corollary 2, there exists $m \in \mathbb{N}$ such that $F_{KB} \downarrow m = F_{KB} \downarrow k$ for all $k \geq m$, i.e. $F_{KB} \downarrow m = F_{KB} \downarrow \omega$. So the vector whose components are the spikes on the neurons $\sigma_1, \ldots, \sigma_n$ in the configuration $\mathbb{C}_{2m+1}$ is the result obtained by applying the inference rule *Negation of Failure Set*. □

Example 5 Let us consider the deductive database KB from Example 1 and the SN P system associated to KB. Its graphical representation is shown in Fig. 2.

All steps of the computation (downwards and upwards) are shown in Table 1. Note that in Table 1 the solution of applying failure operator at every step is codified on the neurons $\sigma_1, \ldots, \sigma_n$ (*bold values*).

4 Conclusions and Future Work

In the last years, the success of technological devices inspired in the connections on neurons in the brain have is doubtless. Almost each day we read news about new achievements obtained by new models or new architectures. Many of the recent developments on neural networks get new knowledge able to predict or classify with an impressive accuracy, but

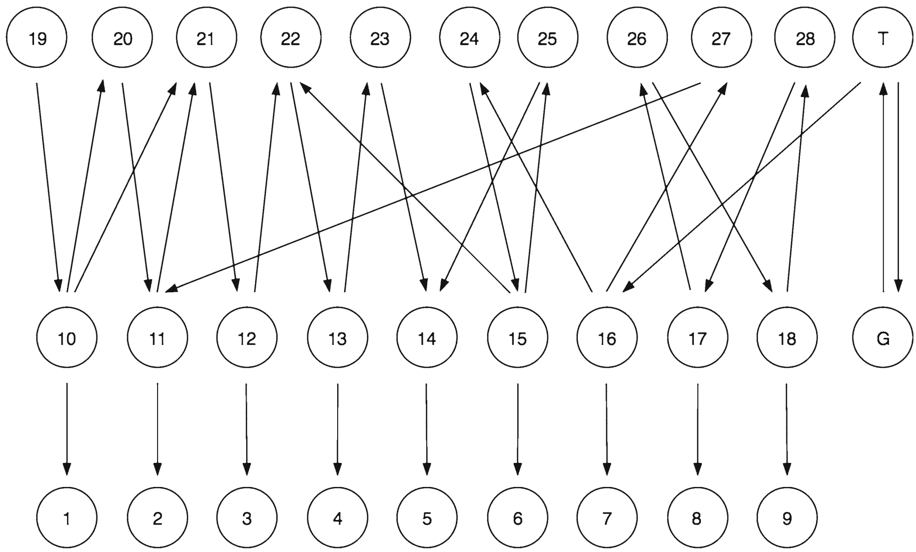

Fig. 2 Graphical representation of the synapses of the SN P system obtained from Example 1

Table 1 $F_{KB} \downarrow \omega$ *(left)* and $F_{KB} \uparrow \omega$ *(right)* of the SN P system of Fig. 2

	$\mathbb{C}_0$	$\mathbb{C}_1$	$\mathbb{C}_2$	$\mathbb{C}_3$	$\mathbb{C}_4$	$\mathbb{C}_5$	$\mathbb{C}_6$	$\mathbb{C}_7$	$\mathbb{C}_8$	$\mathbb{C}_9$
σ_1	0	**0**	0	**0**	0	**0**	0	**0**	0	**0**
σ_2	0	**0**	0	**0**	0	**0**	0	**0**	0	**0**
σ_3	0	**0**	0	**0**	0	**0**	0	**0**	0	**0**
σ_4	0	**0**	0	**0**	0	**0**	0	**1**	0	**1**
σ_5	0	**0**	0	**0**	0	**0**	0	**0**	0	**1**
σ_6	0	**0**	0	**0**	0	**1**	0	**1**	0	**1**
σ_7	0	**0**	0	**1**	0	**1**	0	**1**	0	**1**
σ_8	0	**0**	0	**0**	0	**0**	0	**0**	0	**0**
σ_9	0	**0**	0	**0**	0	**0**	0	**0**	0	**0**
σ_{10}	0	0	0	0	0	0	0	0	0	0
σ_{11}	0	0	0	0	1	0	1	0	1	0
σ_{12}	0	0	0	0	0	0	0	0	0	0
σ_{13}	0	0	0	0	0	0	1	0	1	0
σ_{14}	0	0	0	0	0	0	1	0	2	0
σ_{15}	0	0	0	0	1	0	1	0	1	0
σ_{16}	0	0	1	0	1	0	1	0	1	0
σ_{17}	0	0	0	0	0	0	0	0	0	0
σ_{18}	0	0	0	0	0	0	0	0	0	0
σ_{19}	0	0	0	0	0	0	0	0	0	0
σ_{20}	0	0	0	0	0	0	0	0	0	0
σ_{21}	0	0	0	0	0	0	0	0	0	0
σ_{22}	0	0	0	0	0	1	0	1	0	1

Table 1 continued

	$\mathbb{C}_0$	$\mathbb{C}_1$	$\mathbb{C}_2$	$\mathbb{C}_3$	$\mathbb{C}_4$	$\mathbb{C}_5$	$\mathbb{C}_6$	$\mathbb{C}_7$	$\mathbb{C}_8$	$\mathbb{C}_9$
σ_{23}	0	0	0	0	0	0	0	1	0	1
σ_{24}	0	0	0	1	0	1	0	1	0	1
σ_{25}	0	0	0	0	0	1	0	1	0	1
σ_{26}	0	0	0	0	0	0	0	0	0	0
σ_{27}	0	0	0	1	0	1	0	1	0	1
σ_{28}	0	0	0	0	0	0	0	0	0	0
σ_T	0	1	0	1	0	1	0	1	0	1
σ_G	1	0	1	0	1	0	1	0	1	0

	$\mathbb{C}_0$	$\mathbb{C}_1$	$\mathbb{C}_2$	$\mathbb{C}_3$	$\mathbb{C}_4$	$\mathbb{C}_5$	$\mathbb{C}_6$	$\mathbb{C}_7$
σ_1	0	**1**	0	**0**	0	**0**	0	**0**
σ_2	0	**1**	0	**1**	0	**0**	0	**0**
σ_3	0	**1**	0	**1**	0	**1**	0	**0**
σ_4	0	**1**	0	**1**	0	**1**	0	**1**
σ_5	0	**1**	0	**1**	0	**1**	0	**1**
σ_6	0	**1**	0	**1**	0	**1**	0	**1**
σ_7	0	**1**	0	**1**	0	**1**	0	**1**
σ_8	0	**1**	0	**1**	0	**1**	0	**1**
σ_9	0	**1**	0	**1**	0	**1**	0	**1**
σ_{10}	1	0	0	0	0	0	0	0
σ_{11}	2	0	2	0	0	0	0	0
σ_{12}	1	0	1	0	1	0	0	0
σ_{13}	1	0	1	0	1	0	1	0
σ_{14}	2	0	2	0	2	0	2	0
σ_{15}	1	0	1	0	1	0	1	0
σ_{16}	1	0	1	0	1	0	1	0
σ_{17}	1	0	1	0	1	0	1	0
σ_{18}	1	0	1	0	1	0	1	0
σ_{19}	0	0	0	0	0	0	0	0
σ_{20}	0	1	0	0	0	0	0	0
σ_{21}	0	2	0	1	0	0	0	0
σ_{22}	0	2	0	2	0	2	0	0
σ_{23}	0	1	0	1	0	1	0	1
σ_{24}	0	1	0	1	0	1	0	1
σ_{25}	0	1	0	1	0	1	0	1
σ_{26}	0	1	0	1	0	1	0	1
σ_{27}	0	1	0	1	0	1	0	1
σ_{28}	0	1	0	1	0	1	0	1
σ_T	0	1	0	1	0	1	0	1
σ_G	1	0	1	0	1	0	1	0

such implicit knowledge is not human readable. Recently, many researchers have started to wonder how to *translate* this implicit knowledge into a set of rules in order to be understood by humans and then, to be able to introduce new improvements in the technical designs. In the literature, different approaches by using connectionist models for logic-based representation and reasoning can be found. For example, in [29], a study of the relation between the SAT problem the minimizing energy in several types of neural networks is presented.

Such translation needs bridges and two of them can be, on the one hand, an appropriate logic where a statement has associated a *True* value and some kind of inference rules to acquire more knowledge and, on the other hand, a neural-inspired model able to handle with binary information, as SN P systems do.

In this paper, we propose a possible bridge by studying two non-monotonic logic inference rules into a neural-inspired model. This new point of view could shed a new light to further research possibilities. On the one side, to study if new inference rules can be studied in the framework of SN P systems. On the other side, if other bio-inspired models are also capable of dealing with logic inference rules.

Recently there exist other approaches to model logic-based reasoning with neural models that tackle questions on entailment and satisfiability. In [2], the authors use a type of Hopfield networks to model and solve non-horn 3-SAT, although are models of continuous nature. Moreover, SN P systems can be useful models to both design and verify logic-based tasks. As future work, an interesting research line can be to discretize classical continuous spiking models and to model them via SN P systems. The target is to explore techniques for verifying and validating such models in industrial applications as robotics [7,32].

Acknowledgements Joaquín Borrego-Díaz's work is partially supported by the project PID2019-109152GB-I00 and Miguel A. Gutiérrez-Naranjo's work is partially supported by the project PID2019-107339GB-I00. Both projects are funded by the Spanish Ministry of Science, Innovation and Universities *(Ministerio de Ciencia, Innovación y Universidades)* co-financed by FEDER founds. Daniel Rodríguez Chavarría thanks the partial support by the Youth Employment Operative Program, co-financied with FEDER founds. We also thank reviewers for their valuable comments and suggestions, which have improved the content of this article.

References

1. Adadi A, Berrada M (2018) Peeking inside the black-box: a survey on explainable artificial intelligence (XAI). IEEE Access 6:52138–52160
2. Alzaeemi SAS, Sathasivam S, Adebayo SA, Kasihmuddin MSM, Mansor MA (2017) Kernel machine to doing logic programming in Hopfield network for solve non horn problem-3SAT. MOJ Appl Bion Biomech 1(1):1–6
3. Apt KR, Bol RN (1994) Special issue: ten years of logic programming logic programming and negation: a survey. J Log Program 19:9–71
4. Bader S, Hitzler P (2005) Dimensions of neural-symbolic integration—a structured survey. In: Artemov S, Barringer H, d'Avila Garcez AS, Lamb L, Woods J (eds) We will show them: essays in honour of Dov Gabbay, vol 1. King's College Publications, London, pp 167–194
5. Besold TR, d'Avila Garcez AS, Bader S, Bowman H, Domingos PM, Hitzler P, Kühnberger K, Lamb LC, Lowd D, Lima PMV, de Penning L, Pinkas G, Poon H, Zaverucha G (2017) Neural-symbolic learning and reasoning: a survey and interpretation. arXiv:1711.03902
6. Besold TR, Kühnberger KU (2015) Towards integrated neural-symbolic systems for human-level AI: two research programs helping to bridge the gaps. Biol Inspir Cogn Archit 14:97–110
7. Bing Z, Meschede C, Röhrbein F, Huang K, Knoll AC (2018) A survey of robotics control based on learning-inspired spiking neural networks. Front Neurorobot 12:1–35
8. Cabarle FGC, Adorna HN, Pérez-Jiménez MJ, Song T (2015) Spiking neural P systems with structural plasticity. Neural Comput Appl 26(8):1905–1917
9. Clark KL. Negation as failure. In: Gallaire and Minker [13], pp 293–322

10. Díaz-Pernil D, Gutiérrez-Naranjo MA (2018) Semantics of deductive databases with spiking neural P systems. Neurocomputing 272:365–373
11. Floreano D, Mattiussi C (2008) Bio-inspired artificial intelligence: theories, methods, and technologies. MIT Press, Cambridge
12. Gallaire H, Minker J (eds) (1978) Logic and data bases, symposium on logic and data bases, Centre d'études et de recherches de Toulouse, 1977. Advances in data base theory. Plemum Press, New York
13. Gerstner W, Kistler W (2002) Spiking neuron models: single neurons, populations, plasticity. Cambridge University Press, Cambridge
14. Gilpin LH, Bau D, Yuan BZ, Bajwa A, Specter M, Kagal L (2018) Explaining explanations: an approach to evaluating interpretability of machine learning. arXiv:1806.00069
15. Goodfellow I, Bengio Y, Courville A (2016) Deep learning. MIT Press, Cambridge
16. Graciani C, Riscos-Núñez A, Păun G, Rozenberg G, Salomaa A (2018) Enjoying natural computing: essays dedicated to Mario de Jesús Pérez-Jiménez on the Occasion of His 70th Birthday. Lecture Notes in Computer Science. Springer International Publishing
17. Hammer B, Hitzler P (eds) (2007) Perspectives of neural-symbolic integration, studies in computational intelligence, vol 77. Springer, Berlin
18. Hitzler P, Seda AK (2011) Mathematical aspects of logic programming semantics. Chapman and Hall/CRC studies in informatics series. CRC Press, Boca Raton
19. Huang K, Zhang G, Wei X, Rong H, He Y, Wang T (2016) Fault classification of power transmission lines using fuzzy reasoning spiking neural P systems. In: Gong M, Pan L, Song T, Zhang G (eds) Bio-inspired computing—theories and applications—11th international conference, BIC-TA 2016, Xi'an, China, October 28–30, 2016, Revised selected papers, Part I, Communications in computer and information science, 681st edn. Springer, Berlin, pp 109–117
20. Ibarra OH, Leporati A, Păun A, Woodworth S (2010) Spiking neural P systems. In: Păun Gh, Rozenberg G, Salomaa A (eds) The Oxford handbook of membrane computing. Oxford University Press, Oxford, pp 337–362
21. Ionescu M, Păun Gh, Yokomori T (2006) Spiking neural P systems. Fundamenta Informaticae 71(2–3):279–308
22. Jaffar J, Lassez J, Lloyd JW (1983) Completeness of the negation as failure rule. In: Bundy A (ed) Proceedings of the 8th international joint conference on artificial intelligence. Karlsruhe, FRG, August 1983. William Kaufmann, New York, pp 500–506
23. Kakas AC (1992) Default reasoning via negation as failure. In: Lakemeyer G, Nebel B (eds) Foundation of knowledge representation and reasoning, Lecture notes in computer science, vol 810. Springer, Berlin, pp 160–178
24. Lifschitz V (1999) Answer set planning. In: DD Schreye (ed) Logic programming: the 1999 international conference, Las Cruces, New Mexico, USA, November 29–December 4, 1999. MIT Press, Cambridge, pp 23–37
25. Lloyd J (1987) Foundations of logic programming. Symbolic computation: artificial intelligence. Springer, Berlin
26. Maass W (1997) Networks of spiking neurons: the third generation of neural network models. Neural Netw 10(9):1659–1671
27. Montavon G, Lapuschkin S, Binder A, Samek W, Müller K (2017) Explaining nonlinear classification decisions with deep Taylor decomposition. Pattern Recogn 65:211–222
28. Pan L, Păun Gh, Zhang G, Neri F (2017) Spiking neural P systems with communication on request. Int J Neural Syst 27(8):1–13
29. Pinkas G (1991) Symmetric neural networks and propositional logic satisfiability. Neural Comput 3(2):282–291
30. Reiter R. On closed world data bases. In: Gallaire and Minker [13], pp 55–76
31. Samek W, Binder A, Montavon G, Lapuschkin S, Müller K (2017) Evaluating the visualization of what a deep neural network has learned. IEEE Trans Neural Netw Learn Syst 28(11):2660–2673
32. Sinapayen L, Masumori A, Ikegami T (2019) Reactive, proactive, and inductive agents: an evolutionary path for biological and artificial spiking networks. arXiv:1902.06410
33. Song T, Pan L (2016) Spiking neural P systems with request rules. Neurocomputing 193:193–200
34. Song T, Zou Q, Liu X, Zeng X (2015) Asynchronous spiking neural P systems with rules on synapses. Neurocomputing 151:1439–1445
35. Tran SN, d'Avila Garcez AS (2018) Deep logic networks: inserting and extracting knowledge from deep belief networks. IEEE Trans Neural Netw Learn Syst 29(2):246–258
36. van Emden MH, Kowalski RA (1976) The semantics of predicate logic as a programming language. J ACM 23(4):733–742

37. Wang T, Zhang G, Zhao J, He Z, Wang J, Pérez-Jiménez MJ (2015) Fault diagnosis of electric power systems based on fuzzy reasoning spiking neural P systems. IEEE Trans Power Syst 30(3):1182–1194
38. Wu T, Zhang Z, Păun Gh, Pan L (2016) Cell-like spiking neural P systems. Theor Comput Sci 623:180–189
39. Yang F (1991) Duality in logic programming. Tech. Rep. Paper 119, Electrical Engineering and Computer Science Technical Reports. Syracuse University
40. Zeng X, Zhang X, Song T, Pan L (2014) Spiking neural P systems with thresholds. Neural Comput 26(7):1340–1361
41. Zhang G, Pérez-Jiménez M, Gheorghe M (2017) Real-life Applications with MembraneReal-life applications with membrane computing. Emergence, complexity and computation. Springer, Berlin