Compositional Semantics of Spiking Neural P Systems

Roberto Barbuti[a], Andrea Maggiolo-Schettini[a], Paolo Milazzo[a,*], Simone Tini[b]

[a]*Dipartimento di Informatica, Università di Pisa, Largo B. Pontecorvo 3, 56127 Pisa, Italy*
[b]*Dipartimento di Informatica e Comunicazione, Università dell'Insubria, Via Mazzini 5, 21100 Varese, Italy*

Abstract

The aim of the paper is to give a formal compositional semantics for Spiking Neural P systems (SNP systems) by following the Structural Operational Semantics (SOS) approach. A process algebra is introduced whose terms represent SNP systems. The algebra is equipped with a semantics, given as a labelled transition system. This semantics allows notions of behavioural equivalences over SNP systems to be studied. Some known equivalences are considered and their definition based on the given semantics is provided. Such equivalences are proved to be congruences.

Keywords: Spiking Neural P systems, Structural Operational Semantics, Behavioural equivalence, Congruence.

1. Introduction

Research in membrane computing was initiated by Păun with the definition of membrane systems. A membrane system (or P system, see [1] for an introduction) is a parallel computing device inspired by the structure and the functioning of a living cell. It consists of a hierarchy of membranes, each of them containing a multiset of objects, representing molecules, a set of evolution rules, representing chemical reactions, and possibly other membranes. Evolution rules are applied with maximal parallelism, namely it cannot happen that an evolution rule is not applied when the objects it consumes are available.

Many membrane computing models have been defined (see e.g. [2, 3, 4, 5, 6]) based on abstractions of different biological mechanisms. Recently, a new model called *spiking neural P systems* (SNP systems) was defined inspired by the natural processes of spiking neurons in brain [7]. An SNP system consists of a set of neurons connected by synapses. The structure is represented by a directed graph where nodes represent the neurons and edges represent the

*Corresponding author

Email addresses: `barbuti@di.unipi.it` (Roberto Barbuti), `maggiolo@di.unipi.it` (Andrea Maggiolo-Schettini), `milazzo@di.unipi.it` (Paolo Milazzo), `simone.tini@uninsubria.it` (Simone Tini)

synapses. A neuron may contain a number of *spikes*, each represented by an occurrence of symbol a, and some rules. When a neuron fires (because one of its rules is applied), it sends, after some delay, a spike along each outgoing synapse to neighboring neurons, which then process the received spikes. One of the neurons can also send spikes to the environment, so to produce an output.

The semantics of SNP systems is usually given by means of a precise, but not formal, description of the possible computation steps of a system. The aim of this paper is to provide a formal compositional semantics for SNP systems. Following the SOS approach [8], we introduce a process algebra, called *SNP algebra*, whose terms represent either *neuron contents*, namely multisets of spikes and sets of rules, or *neurons*, or *juxtaposition of neurons*, namely sets of neurons that can be connected to each other by means of synapses. We equip the SNP algebra with a Labelled Transition System (LTS), obtained by means of a set of transition rules. To describe the observable part of neuron behaviour, LTS transition labels contain information on input and output spikes, namely spikes that are received and sent by each neuron in the described computation step.

The given semantics allows us to study some notions of *behavioural equivalences* over SNP systems, by formalizing the idea of neural systems having the same behaviour and being substitutable to each other. We shall prove that the considered equivalences are congruences, meaning that the equivalences are preserved when two equivalent systems are put in the same context. Our semantics could be also used for the construction of formal analysis and verification tools.

As regards related work, to our knowledge there are no papers dealing with operational semantics of SNP systems. Operational semantics have been defined for membrane systems in [9, 10, 11, 12, 13, 14]. The definition of the semantics of SNP systems given in this paper follows the lines of the definition of the semantics of membrane systems given in [13].

2. Spiking Neural P Systems

A spiking neural P system [7] is essentially a directed graph in which nodes represent (and are called) *neurons* and edges represent (and are called) *synapses* among neurons. Neurons may contain a number of *spikes*, each represented by an occurrence of symbol a, and a number of rules of two kinds:

- *spiking rules*, having the form $E/a^n \rightarrow a; t$, where E is a regular expression on the singleton alphabet $\{a\}$, $n \in \mathbb{N}$ with $n > 0$, and $t \in \mathbb{N}$ with $t \geq 0$;
- *forgetting rules*, having the form $a^n \rightarrow \lambda$ where $n \in \mathbb{N}$ with $n > 0$.

In the following, we do not write E in a spiking rule when E equals a^n, and we do not write t when it is 0. Let us denote with $\Re$ the infinite set of all possible spiking and forgetting rules, and let us write $L(E)$ for the language denoted by the regular expression E.

Rules contained in a neuron have to be applied to spikes contained in that neuron. The application of a spiking rule $E/a^n \rightarrow a; t$ may involve more than one computation step. We say that $E/a^n \rightarrow a; t$ *fires* when its application begins, and that it *spikes* in the last step of its application. The number of steps between firing and spiking is given by the value of t. When $t = 0$ we have that firing and spiking are performed simultaneously, when $t = 1$ they are performed in two subsequent steps, and so on. A spiking rule $E/a^n \to a; t$ can be applied only if in the neuron there is a number of spikes that is described by the regular expression E. When this happens, a^n spikes are removed from the neuron content at firing time, whereas at spiking time one spike a is sent to each neuron of the system that can be reached immediately by following the outgoing synapses. The spikes sent at spiking time will be available in the content of the recipients neurons in the computation step immediately after the spiking. During the application of a spiking rule, the neuron containing the rule cannot receive any spike from other neurons, and it is said to be *closed* (i.e. in the refractory period). Actually, the neuron is closed during the firing step (if $t > 0$), but it is not closed during the spiking step, in which spikes can be received. Spikes sent by a neuron to a closed neuron are discarded. A neuron that is not closed is said to be open. At each computation step one rule is applied in each open neuron having some applicable rules. A forgetting rule $a^n \to \lambda$ can be applied only if in the neuron there are exactly n spikes. Application of a forgetting rule is done in a single computation step, and the result of the application is that the n spikes of the neuron are deleted. It is usually assumed (also in [7] and here) that a neuron does not contain a spiking rule and a forgetting rule that can be applied at the same time. This means that the left hand side of each forgetting rule must not be described by any of the regular expressions of the spiking rules of the neuron. This means also that non-determinism can be introduced only by spiking rules of the same neuron whose regular expressions describe languages with non-empty intersection.

A computation of a spiking neural P system is a sequence of steps in which rules are applied, by starting from a given initial configuration. One of the neurons of the system has to be chosen as the output neuron. We consider spiking neural P systems generating natural numbers as in [7]: in a *valid* computation the output neuron applies spiking rules only twice and the number of steps executed between the first and the second spike gives the generated number. All the other computations in which the output neuron does not spike exactly twice are considered invalid and are discarded.

Spiking neural P systems are formally defined as follows.

Definition 1 (SNP System). A *spiking neural P system (SNP system)* is a tuple $\Pi = (V, \sigma_1, \ldots, \sigma_m, syn, i_0)$ where:

- $V = \{a\}$ is the singleton alphabet;
- $\sigma_1, \ldots, \sigma_m$ are pairs $\sigma_i = (n_i, \mathcal{R}_i), 1 \leq i \leq m$, called *neurons*, such that $n_i \in \mathbb{N}$, $n_i \geq 0$, $\mathcal{R}_i \subseteq \Re$, and $\mathcal{R}_i$ is finite;
- syn, called *synapses*, is an irreflexive relation on $\{1, \ldots, m\} \times \{1, \ldots, m\}$;
- $i_0 \in \{1, \ldots, m\}$ indicates the *output neuron*.

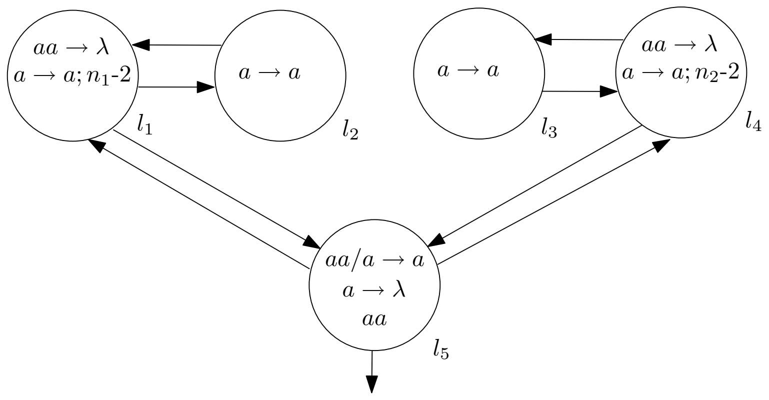

Figure 1: A SNP system computing the l.c.m. of n_1 and n_2, with $n_1 \geq 2$ and $n_2 \geq 2$.

Example 1. In Fig. 1 we give an example of SNP system computing the least common multiple of n_1 and n_2, with $n_1 \geq 2$ and $n_2 \geq 2$. Let m be such a value. The output neuron l_5 is depicted with a dangling outgoing arrow. It spikes at time 1, when aa triggers the spiking rule $aa/a \to a$. Then, neuron l_1 (resp. l_4) receives from l_5 the spike a at time 1 and enters a loop. Starting with $k = 0$, it fires at time $2 + n_1k$ (resp. $2 + n_2k$), spikes at time $2 + n_1k + (n_1 - 2)$ (resp. $2+n_2k+(n_2-2)$) sending a to both l_5 and l_2 (resp. l_3), receives the spike from l_2 (resp. l_3) at time $2+n_1k+(n_1-1)$ (resp. $2+n_2k+(n_2-1)$) and fires once more at time $2 + n_1(k + 1)$ (resp. $2 + n_2(k + 1)$). These loops are kept until l_1 and l_4 do not spike together, so that l_5 receives only a spike, either from l_1 or from l_4, and removes it with forgetting rule $a \to \lambda$. But, at time m, both l_1 and l_4 spike, l_5 receives two spikes, one from l_1 and one from l_4, which are exploited by l_5 to spike at time $m + 1$ with rule $aa/a \to a$. Since l_1 (resp. l_4) receives a spike also from l_2 (resp. l_3), at time $m+1$ it performs the firing rule $aa \to \lambda$ so that all spikes are removed.

3. The SNP Algebra

Let us denote with $\mathcal{L}$ the set of all possible neuron labels. Moreover, let us omit the $\cup$ operator from set and multiset unions, namely given two sets (or multisets) s_1 and s_2 we will write s_1s_2 for $s_1 \cup s_2$. The abstract syntax of the SNP algebra is defined as follows.

Definition 2 (SNP algebra). The abstract syntax of *neuron contents* c and *spiking neural systems sns* is given by the following grammar:

$$\begin{aligned} c \; &::= (\varnothing, \varnothing, 0) \;\big|\; (\varnothing, a, 0) \;\big|\; (E/a^n \to a; t, \varnothing, t') \;\big|\; (a^n \to \lambda, \varnothing, 0) \;\big|\; c \cup c \\ sns \; &::= [\,c\,]_l^L \;\big|\; sns \mid sns \;\big|\; (l)sns \end{aligned}$$

where t and t' range over $\mathbb{N}$, $t' \leq t$, l ranges over $\mathcal{L}$ and L ranges over $2^{\mathcal{L}}$.

A neuron content c represents the computation state of a neuron, which consists of a triple $(\mathcal{R}, u, t)$, where $\mathcal{R}$ is the finite set of spiking and forgetting rules, u is a multiset of spikes (actually, $u = a^n$ for some $n \geq 0$), and $t \in \mathbb{N}$ is a *timer*. The intuition is that u represents the spikes inside the neuron and t is the waiting time for the next spiking. Therefore, the timer is 0 in the initial computation state, the timer becomes t when a rule $E/a^n \to a; t$ with $t > 0$ fires, and a timer $t > 0$ is set to $t - 1$ when the clock ticks.

A neuron content c is obtained through the union operator $\_ \cup \_$ from constants representing single rules and single spikes. Actually, we have that $(\mathcal{R}_1, u_1, t_1) \cup (\mathcal{R}_2, u_2, t_2)$ represents the triple $(\mathcal{R}_1\mathcal{R}_2, u_1u_2, \max(t_1, t_2))$. The semantic rules of the SNP algebra will ensure that by starting from a correct representation of the initial computation state of a spiking neural P system, namely, with all timers equal to zero, at most one of the timers in each neuron content will be greater than zero. Hence, $\max(t_1, t_2)$ is the non-zero timer, if any, between t_1 and t_2. We shall say that a neuron content $(\mathcal{R}, u, t)$ is *open* (i.e. not in refractory period) if $t = 0$, and that it is closed (i.e. in refractory period) if $t > 0$.

A neuron can be obtained from a neuron content c by applying the operation $[\,\_\,]_l^L$. Here, l is the label of the neuron and L is the set of labels of the neurons that might send a spike to l. In other words, L represents the incoming edges of neuron l in the graph representing the spiking neural P system. Operation $\_ \mid \_$ represents the juxtaposition of neurons. Since each neuron contains information on its incoming edges, the result of a juxtaposition operation can be seen as a graph. A spiking neural P system of degree n can hence be represented as a term $[\,c_1\,]_{l_1}^{L_1} \mid \ldots \mid [\,c_n\,]_{l_n}^{L_n}$, where each L_i, with $1 \leq i \leq n$, is a subset of $\{l_1, \ldots, l_n\}$.

Finally, the restriction operation $(l)sns$ makes neuron l of system sns invisible to other neurons in further juxtapositions. This means that neuron l will be allowed to have outgoing synapses only towards other neurons in sns (i.e. in the scope of l). The restriction operator has no corresponding notion in the standard definition of SNP systems, but permits us to develop systems in a modular way. In fact, since outgoing synapses are the only way for SNP systems to send information, they represent the output interfaces of the systems. The restriction operation permits us to select which interfaces we want to keep visible from the external world. Notice that input interfaces are selected by choosing the set L in neuron $[c]_l^L$.

Example 2. Terms of the SNP algebra, namely spiking neural systems, represent SNP systems. For example, let us consider the SNP system in Fig.1. Neuron l_1 can be represented by the following term of the algebra:

$$[\,(aa \to \lambda, \varnothing, 0) \cup (a \to a; n_1 - 2, \varnothing, 0)\,]_{l_1}^{\{l_2, l_5\}}\ .$$

In the examples we shall give, let us slightly simplify the notation as follows:

$$[\,aa \to \lambda\,,\ a \to a; n_1 - 2\,,\ \varnothing\,,\ 0\,]_{l_1}^{\{l_2, l_5\}}$$

where $\varnothing$ is the multiset of available spikes and 0 is the sum of the timers associated with the rules. Neuron l_2 can be represented by the following term:

$$[\, a \to a \,,\, \varnothing \,,\, 0\,]_{l_2}^{\{l_1\}}$$

and neuron l_5 by the following term:

$$[\, aa/a \to a \,,\, a \to \lambda \,,\, aa \,,\, 0\,]_{l_5}^{\{l_1,l_4\}} \,.$$

Neurons l_3 and l_4 are similar to l_2 and l_1, respectively. The whole SNP system can be represented by the following term:

$$\begin{aligned} S_{lcm} \ ::= {} & [\, aa \to \lambda \,,\, a \to a ; n_1 - 2 \,,\, \varnothing \,,\, 0\,]_{l_1}^{\{l_2,l_5\}} \mid [\, a \to a \,,\, \varnothing \,,\, 0\,]_{l_2}^{\{l_1\}} \mid \\ & [\, a \to a \,,\, \varnothing \,,\, 0\,]_{l_3}^{\{l_4\}} \mid [\, aa \to \lambda \,,\, a \to a ; n_2 - 2 \,,\, \varnothing \,,\, 0\,]_{l_4}^{\{l_3,l_5\}} \mid \\ & [\, aa/a \to a \,,\, a \to \lambda \,,\, aa \,,\, 0\,]_{l_5}^{\{l_1,l_4\}} \,. \end{aligned}$$

Note that we might compose S_{lcm} with further neurons which receive spikes from any of the neurons $l_1, \ldots, l_5$. For instance, $S_{lcm} \mid [\varnothing, \varnothing, 0]_{l_6}^{\{l_2\}}$ would represent an SNP system in which l_2 sends spikes also to the new neuron l_6. However, system S_{lcm} might be a component of a larger system we are specifying in a modular way. Hence, we might desire only the output neuron (or a few key neurons) to be visible from other components of the system. Restriction can be used to this purpose, and we might rewrite S_{lcm} as follows:

$$\begin{aligned} S'_{lcm} \ ::= {} & (l_1)(l_4)\big(\\ & (l_2)\big([\, aa \to \lambda \,,\, a \to a ; n_1 - 2 \,,\, \varnothing \,,\, 0\,]_{l_1}^{\{l_2,l_5\}} \mid [\, a \to a \,,\, \varnothing \,,\, 0\,]_{l_2}^{\{l_1\}}\big) \mid \\ & (l_3)\big([\, a \to a \,,\, \varnothing \,,\, 0\,]_{l_3}^{\{l_4\}} \mid [\, aa \to \lambda \,,\, a \to a ; n_2 - 2 \,,\, \varnothing \,,\, 0\,]_{l_4}^{\{l_3,l_5\}}\big) \mid \\ & [\, aa/a \to a \,,\, a \to \lambda \,,\, aa \,,\, 0\,]_{l_5}^{\{l_1,l_4\}} \\ & \big) \,. \end{aligned}$$

We have that only spikes sent by l_5 can be received by neurons in the context.

Restriction allows us to distinguish the labels of the neurons of a system between *bound*, namely in the scope of some restriction operation, and *free*, namely not in the scope of any restriction operation. Moreover, we say that a label l is *pending* if no neuron in the system is labeled l, and there is a neuron in the system willing to receive spikes from some neuron labeled l.

Definition 3 (Bound, free and pending labels). The set of *bound labels* of a spiking neural system sns, denoted $BL(sns) \subset \mathcal{L}$, is recursively defined as follows:

$$BL(sns) = \begin{cases} \varnothing & \text{if } sns = [\, c\,]_l^L \\ BL(sns_1) \cup BL(sns_2) & \text{if } sns = sns_1 \mid sns_2 \\ \{l\} \cup BL(sns') & \text{if } sns = (l)sns' \end{cases}$$

The set of *free labels* of a spiking neural system sns, denoted $FL(sns) \subset \mathcal{L}$, is recursively defined as follows:

$$FL(sns) = \begin{cases} \{l\} & \text{if } sns = [\,c\,]_l^L \\ FL(sns_1) \cup FL(sns_2) & \text{if } sns = sns_1 \mid sns_2 \\ FL(sns') \setminus \{l\} & \text{if } sns = (l)sns' \end{cases}$$

The set of *pending labels* of a spiking neural system sns, denoted $PL(sns) \subseteq \mathcal{L}$, is recursively defined as follows:

$$PL(sns) = \begin{cases} L & \text{if } sns = [\,c\,]_l^L \\ PL(sns_1) \setminus FL(sns_2) \cup PL(sns_2) \setminus FL(sns_1) & \text{if } sns = sns_1 \mid sns_2 \\ PL(sns') & \text{if } sns = (l)sns' \end{cases}$$

We say that a spiking neural system sns is *complete* if and only if $PL(sns) = \varnothing$. Note that only complete spiking neural systems have corresponding SNP systems, since in non-complete ones synapses are not properly specified.

The following constraints are mandatory to represent spiking neural P systems as defined in [7]:

- for a neuron content $(E/a^n \to a; t, \varnothing, t')$ it holds that $n \geq 1$ and $a^n \subseteq u$ for every $u \in L(E)$;
- for a neuron content $(\mathcal{R}, u, t)$, given any $E/a^n \to a; t$ and $a^h \to \lambda$ in $\mathcal{R}$, it holds that $a^h \notin L(E)$;
- for a neuron $[\,c\,]_l^L$, it holds that $l \notin L$;
- for a juxtaposition $sns_1 \mid sns_2$ it holds that $FL(sns_1) \cap FL(sns_2) = \varnothing$.

Now, in order to give the semantics of the SNP algebra, let us recall the model of labeled transition systems [15, 8].

Definition 4 (LTS). A *labeled transition system* (LTS for short) is a triple $(S, Lab, \{\xrightarrow{\ell} \mid \ell \in Lab\})$, where S is a set of *states*, Lab is a set of *labels*, and $\xrightarrow{\ell} \subseteq S \times S$ is a *transition relation* for each $\ell \in Lab$.

As usual, we write $s \xrightarrow{\ell} s'$ for $(s, s') \in \xrightarrow{\ell}$.

The semantics of the SNP algebra is given in terms of an LTS, with a state for each syntactically correct term and a labeled transition for each computation step. LTS labels can be of the following forms:

- (U, E, u, v, v', I, O), describing a computation step performed by an open neuron content c, where:
  - U is a set of regular expressions corresponding to application conditions of the rules in c. Namely, for each spiking rule $E/a^n \to a; t$ in c it holds that $E \in U$. Moreover, for each forgetting rule $a^n \to \lambda$ in c it holds that $a^n \in U$.

  - E is either the regular expression corresponding to the application conditions of the rule fired in the described computation step, or the empty regular expression if no rule is fired in such a step. Note that if E is non-empty then E is in U.
  - u is either the multiset of spikes in c that are consumed by the spiking or forgetting rule fired in the described computation step, or the empty multiset if no rule is fired in such a step. The multiset u will be obtained from the semantic rules describing the behaviour of the individual spiking and forgetting rules in c.
  - v is the same as u, but it is obtained from the semantic rules of the individual spikes in c. When c is inserted into a neuron $[\,c\,]_l^L$, condition $u = v$ will be checked and transitions having a label with $u \neq v$ are discarded, to ensure that spikes consumed by spiking or forgetting rules are available inside $[\,c\,]_l^L$ and are removed from $[\,c\,]_l^L$.
  - v' is the multiset of spikes in c that are not consumed in the described computation step. When c is inserted into a neuron $[\,c\,]_l^L$, if E is empty then it is checked that v' does not satisfy any regular expression in U, namely that the available objects do not trigger any spiking or forgetting rule.
  - I is either the multiset of spikes received from other neurons in the described computation step, or $\perp$, which describes that the neuron is closed and, therefore, it cannot receive objects from other neurons.
  - O is the multiset of spikes sent to other neurons in the described computation step. Since at most one spike (possibly to several neurons) can be sent at each step, we have that either $O = \varnothing$ or $O = \{a\}$.

- $(1, I, O)$, describing a computation step performed by a closed neuron content c, where:
  - 1 is a label used just to emphasize that the transition describes a computation step in which timers are decreased by one.
  - I and O are as in the previous case.

- $(\mathcal{I}, \mathcal{O})$, describing a computation step performed by a spiking neural system sns, where:
  - $\mathcal{I}$ is a set of pairs (l, I) describing the multiset of spikes I (actually, with either $I = \varnothing$ or $I = \{a\}$) that is expected to be received in the described step by some neurons in sns from a neuron l not in sns.
  - $\mathcal{O}$ is a set of pairs (l, O) describing the multiset of spikes O (actually, with either $O = \varnothing$ or $O = \{a\}$) that is sent by neuron l in sns to all neurons willing to receive it.

Components I and O in labels of the first two forms, and components $\mathcal{I}$ and $\mathcal{O}$ in labels of the third form, describe the input/output behaviour of SNP

$$\frac{I \in V^*}{(\varnothing, \varnothing, 0) \xrightarrow[\varnothing,\varnothing,\varnothing,\varnothing,\varnothing]{I,\varnothing} (\varnothing, I, 0)} \qquad (c1)$$

$$\frac{I \in V^*}{(\varnothing, a, 0) \xrightarrow[\varnothing,\varnothing,\varnothing,a,\varnothing]{I,\varnothing} (\varnothing, I, 0)} \qquad (c2) \qquad\qquad \frac{I \in V^*}{(\varnothing, a, 0) \xrightarrow[\varnothing,\varnothing,\varnothing,\varnothing,a]{I,\varnothing} (\varnothing, Ia, 0)} \qquad (c3)$$

$$\frac{I \in V^* \qquad t = 0}{(E/a^n \to a; t, \varnothing, 0) \xrightarrow[\{E\},E,a^n,\varnothing,\varnothing]{I,a} (E/a^n \to a; t, I, 0)} \qquad (c4)$$

$$\frac{t > 0}{(E/a^n \to a; t, \varnothing, 0) \xrightarrow[\{E\},E,a^n,\varnothing,\varnothing]{\perp,\varnothing} (E/a^n \to a; t, \varnothing, t)} \qquad (c5)$$

$$\frac{I \in V^*}{(E/a^n \to a; t, \varnothing, 0) \xrightarrow[\{E\},\varnothing,\varnothing,\varnothing,\varnothing]{I,\varnothing} (E/a^n \to a; t, I, 0)} \qquad (c6)$$

$$\frac{I \in V^*}{(a^n \to \lambda, \varnothing, 0) \xrightarrow[\{a^n\},a^n,a^n,\varnothing,\varnothing]{I,\varnothing} (a^n \to \lambda, I, 0)} \qquad (c7)$$

$$\frac{I \in V^*}{(a^n \to \lambda, \varnothing, 0) \xrightarrow[\{a^n\},\varnothing,\varnothing,\varnothing,\varnothing]{I,\varnothing} (a^n \to \lambda, I, 0)} \qquad (c8)$$

Figure 2: Rules for open neuron contents.

algebra terms, namely what is usually considered to be the observable behaviour. Labels of the first form are more complex since U, E, u, v and v' are needed to infer the behaviour of neuron contents compositionally.

For the sake of legibility, in transitions with labels of the first form we shall write the first five elements of the label (namely, U, E, u, v and v') under the arrow denoting the transition and the other two elements (namely, I, O) over the arrow. Similarly, in transitions with labels of the second form we shall write 1 under the arrow and I, O over the arrow.

Now, LTS transitions are defined through SOS transition rules [8] of the form $\frac{premises}{conclusion}$, where the premises are a set of transitions, and the conclusion is a transition. Intuitively, SOS transition rules permit us to infer moves of SNP algebra terms from moves of their subterms. We assume the standard way to associate a set of transitions with a set of transition rules [16].

In Fig. 2 we introduce the inference rules for open neuron contents. Rule $(c1)$ states that an empty neuron content can only receive a multiset of spikes I from other neurons. These spikes will be available at the next computation step. Also the other inference rules in the figure, but rule $(c5)$, describe the reception of a multiset of spikes I from other neurons.

$$\frac{t' > 1}{(E/a^n \to a; t, \varnothing, t') \xrightarrow[1]{\bot,\varnothing} (E/a^n \to a; t, \varnothing, t'-1)} \quad (t1)$$

$$\frac{I \in V^*}{(E/a^n \to a; t, \varnothing, 1) \xrightarrow[1]{I,a} (E/a^n \to a; t, I, 0)} \quad (t2)$$

Figure 3: Rules for closed neuron contents.

Rules ($c2$) and ($c3$) describe the computation steps that can be performed by a neuron content representing an individual spike. Rule ($c2$) deals with the case in which the spike is consumed by some spiking or forgetting rule, whereas rule ($c3$) deals with the case in which the spike is not consumed. In the former case, the spike is no longer present in the next computation state, and it appears in the fourth component of the label that collects all consumed spikes. In the latter case, the spike is still present in the next computation state, and it also appears in the fifth component of the label that collects all non-consumed spikes.
Rules ($c4$) and ($c5$) describe the firing of a spiking rule $E/a^n \to a; t$. In the first case t is 0, whereas in the second case t is positive and, therefore, firing the rule implies the closure of the neuron. In both cases the regular expression E, representing application requirements on the available spikes, appears in the first two elements of the transition label. Moreover, the multiset a^n of the consumed objects appears in the third element of the label. Notice that in rule ($c4$) the spike a appears as the output of this computation step, whereas in rule ($c5$) no output is produced and the timer is set to t. Moreover, in rule ($c5$) the input is $\bot$, since a neuron cannot receive any input when it is becoming closed.
Rule ($c6$) describes a spiking rule that is not firing. This happens when either the application requirements E are not satisfied by the available spikes, or when another spiking rule is non-deterministically chosen for firing, or when in the current computation step the neuron is closed. In these cases the state is left unchanged and E appears in the first transition label element.
Rules ($c7$) and ($c8$) describe the application and non-application, respectively, of a forgetting rule $a^n \to \lambda$. These rules are analogous to rules ($c4$) and ($c6$), respectively, with E replaced by a^n.

In Fig. 3 we give the inference rules for closed neuron contents consisting of a spiking rule that has already fired but that has not yet spiked, namely a spiking rule whose timer is not 0. If the timer is greater than 1 then the inference rule ($t1$) is applied just to decrement the timer, whereas if the timer is 1 then the neuron content is opening, which is described by rule ($t2$), in which the spike a appears as output of the computation step and an input can be received.

In Fig. 4 we give the inference rules for the union of neuron contents $c_1 \cup c_2$. A computation step of $c_1 \cup c_2$ is obtained from a computation step of each of the two components. Rules ($u1$), ($u2$), ($u3$), ($u4$) deal with the case in which both c_1 and c_2 are open. The inference rule ($u1$) is used when neither in c_1 nor in c_2 any

$$\frac{c_1 \xrightarrow[U_1,\varnothing,\varnothing,v_1,v_1']{I_1,\varnothing} c_1' \qquad c_2 \xrightarrow[U_2,\varnothing,\varnothing,v_2,v_2']{I_2,\varnothing} c_2'}{c_1 \cup c_2 \xrightarrow[U_1U_2,\varnothing,\varnothing,v_1v_2,v_1'v_2']{I_1I_2,\varnothing} c_1' \cup c_2'} \quad (u1)$$

$$\frac{c_1 \xrightarrow[U_1,E_1,u_1,v_1,v_1']{I_1,O_1} c_1' \qquad c_2 \xrightarrow[U_2,\varnothing,\varnothing,v_2,v_2']{I_2,\varnothing} c_2'}{c_1 \cup c_2 \xrightarrow[U_1U_2,E_1,u_1,v_1v_2,v_1'v_2']{I_1I_2,O_1} c_1' \cup c_2'} \quad (u2)$$

$$\frac{c_1 \xrightarrow[U_1,E_1,u_1,v_1,v_1']{\perp,\varnothing} c_1' \qquad c_2 \xrightarrow[U_2,\varnothing,\varnothing,v_2,v_2']{\varnothing,\varnothing} c_2'}{c_1 \cup c_2 \xrightarrow[U_1U_2,E_1,u_1,v_1v_2,v_1'v_2']{\perp,\varnothing} c_1' \cup c_2'} \quad (u3)$$

$$\frac{c_1 \xrightarrow[U_1,\{a^n\},a^n,v_1,v_1']{I_1,\varnothing} c_1' \qquad c_2 \xrightarrow[U_2,\varnothing,\varnothing,v_2,v_2']{I_2,\varnothing} c_2'}{c_1 \cup c_2 \xrightarrow[U_1U_2,\{a^n\},a^n,v_1v_2,v_1'v_2']{I_1I_2,\varnothing} c_1' \cup c_2'} \quad (u4)$$

$$\frac{c_1 \xrightarrow[U,\varnothing,\varnothing,\varnothing,v']{\varnothing,\varnothing} c_1' \qquad c_2 \xrightarrow[1]{\perp,\varnothing} c_2'}{c_1 \cup c_2 \xrightarrow[1]{\perp,\varnothing} c_1' \cup c_2'} \quad (u5)$$

$$\frac{c_1 \xrightarrow[U,\varnothing,\varnothing,\varnothing,v']{I_1,\varnothing} c_1' \qquad c_2 \xrightarrow[1]{I_2,\{a\}} c_2'}{c_1 \cup c_2 \xrightarrow[1]{I_1I_2,\{a\}} c_1' \cup c_2'} \quad (u6)$$

Figure 4: Rules for unions of neuron contents.

spiking or forgetting rule fires. In this case the computation steps by c_1 and c_2 are described by labels with some empty components, namely those containing the regular expression E associated with the firing rule and the consumed and output spikes u and O. Note that there might be some spikes in c_1 and c_2, namely v_1 and v_2, that may be consumed in a neuron content to be further composed. The label of the transition of $c_1 \cup c_2$ is simply the union, element by element, of the labels of the two composed transitions.
The inference rules $(u2)$, $(u3)$, $(u4)$ are exploited when in c_1 there is either a spiking or a forgetting rule that is firing. The dual cases in which the firing or forgetting rule is in c_2 is implicitly assumed. Since it cannot happen that more than one rule, either spiking or forgetting, fires, the case with firing rules in both c_1 and c_2 is not considered. Notice that the rule that fires is a spiking rule that immediatly spikes in $(u2)$, a spiking rules with delay in $(u3)$ and a forgetting rule in $(u4)$.
Rules $(u5)$ and $(u6)$ deal with the case in which c_2 contains a spiking rule that has been already fired in a previous step but has not yet spiked, so that c_2 is

closed, and c_1 is open. Dual rules with c_1 closed and c_2 open are implicitely assumed, whereas rules with both c_1 and c_2 closed do not exist since no more than one rule waiting for spiking is admitted. Notice that also $c_1 \cup c_2$ results to be closed. Since no spiking or forgetting rule can fire in c_1, the components of its transition label containing the regular expression E associated with the firing rule, the consumed and output spikes u, v and O must be empty. If c_2 remains closed, then also $c_1 \cup c_2$ remains closed and, therefore, c_1 cannot receive any input, as it is described by $(u5)$. Otherwise, if c_2 is opening then also $c_1 \cup c_2$ opens and, therefore, c_1 can receive any input I_1.

Let us say that a neuron content $c = (\mathcal{R}, u, t')$ is *well formed* iff either $t' = 0$ or there exist two neuron contents $c_1 = (E/a^n \to a; t, \varnothing, t')$ and $c_2 = (\mathcal{R}', u, 0)$ such that $c = c_1 \cup c_2$.

Proposition 1. *Given a well formed neuron content c, if $c \xrightarrow{\ell} c'$ for some label ℓ and neuron content c', then c' is well formed.*

Proof. By induction over the syntactical structure of c. If c represents either the empty neuron content, or a single spike, or a spiking rule or a forgetting rule then the transition $c \xrightarrow{\ell} c'$ is infered from some rule in Fig. 2 or Fig. 3, and the thesis is immediate. Assume that $c = c_1 \cup c_2$. In this case, since c is well formed then both c_1 and c_2 are well formed and by induction hypothesis we can assume that the thesis holds for them. Transition $c \xrightarrow{\ell} c'$ is inferred from two transitions $c_1 \xrightarrow{\ell_1} c_1'$ and $c_2 \xrightarrow{\ell_2} c_2'$ through some transition rule in Fig. 4, and c' is $c_1' \cup c_2'$. Let $c_i = (\mathcal{R}_i, u_i, t_i)$ and $c_i' = (\mathcal{R}_i', u_i', t_i')$, for $1 \leq i \leq 2$. If both $t_1' = 0$ and $t_2' = 0$ then the thesis is immediate. If either $t_1' = 0$ and $t_2' > 0$ or $t_1' > 0$ and $t_2' = 0$ then the thesis holds from the inductive hypothesis over c_1' and c_2'. The form of the transition rules in Fig. 4 ensures that it cannot happen that both $t_1' > 0$ and $t_2' > 0$.

In Fig. 5 we give the inference rules for spiking neural systems. Rule $(sns1)$ describes a computation step performed by an open neuron that is not closing. The transition of the neuron is obtained from an *acceptable* transition of its content, namely a transition in which the third and the fourth label components are equal. Recall that such components represent the spikes consumed by a firing rule as it results from the transition performed by the individual firing rule and the transitions performed by the individual spikes. The fact that these two label components are equal ensures that individual transitions have been coherently chosen during the composition. Rule $(sns1)$ also checks that the conditions on available spikes that determined the firing or non-firing of rules in c are satisfied. In particular, if in the transition performed by c no rule fires (i.e. $E = \varnothing$, that implies $u = v = \varnothing$), then the multiset of available spikes v' must be such that no regular expression in U is satisfied (to this aim, let $\vdash$ be defined as follows: $u \vdash U \iff \exists E \in U. u \in L(E)$). This means that there was no applicable rule in c. If in the transition performed by c there is a rule firing (i.e. $E \neq \varnothing$), then the multiset of available spikes vv' must satisfy E. This means that the firing rule was actually applicable. The input spikes I expected by the neuron content

$$\frac{c \xrightarrow[U,E,u,v,v']{I,O} c' \qquad u = v \qquad \begin{matrix} I = \bigcup_{l' \in L} I_{l'} \\ I_{l'} \subseteq a \end{matrix} \qquad \begin{matrix} E = \varnothing \implies v' \not\models U \\ E \neq \varnothing \implies vv' \in L(E) \end{matrix}}{[\,c\,]_l^L \xrightarrow{\{(l',I_{l'})|l' \in L\},\{(l,O)\}} [\,c'\,]_l^L} \quad (sns1)$$

$$\frac{c \xrightarrow[U,E,u,v,v']{\bot,\varnothing} c' \qquad u = v \qquad \begin{matrix} E = \varnothing \implies v' \not\models U \\ E \neq \varnothing \implies vv' \in L(E) \end{matrix}}{[\,c\,]_l^L \xrightarrow{\varnothing,\{(l,\varnothing)\}} [\,c'\,]_l^L} \quad (sns2)$$

$$\frac{c \xrightarrow[1]{I,O} c' \qquad \begin{matrix} I = \bigcup_{l' \in L} I_{l'} \\ I_{l'} \subseteq a \end{matrix}}{[\,c\,]_l^L \xrightarrow{\{(l',I_{l'})|l' \in L\},\{(l,O)\}} [\,c'\,]_l^L} \quad (sns3) \qquad \frac{c \xrightarrow[1]{\bot,\varnothing} c'}{[\,c\,]_l^L \xrightarrow{\varnothing,\{(l,\varnothing)\}} [\,c'\,]_l^L} \quad (sns4)$$

$$\frac{\begin{matrix} sns_1 \xrightarrow{\mathcal{I}_1,\mathcal{O}_1} sns_1' \qquad sns_2 \xrightarrow{\mathcal{I}_2,\mathcal{O}_2} sns_2' \\ \big((l', I_1) \in \mathcal{I}_1 \wedge (l', O_2) \in \mathcal{O}_2\big) \implies \big((I_1 = O_2)\big) \\ \big((l', I_2) \in \mathcal{I}_2 \wedge (l', O_1) \in \mathcal{O}_1\big) \implies \big((I_2 = O_1)\big) \\ \mathcal{I}_{1'} = \{(l', I_1) \in \mathcal{I}_1 \mid (l', O_2) \notin \mathcal{O}_2\} \qquad \mathcal{I}_{2'} = \{(l', I_2) \in \mathcal{I}_2 \mid (l', O_1) \notin \mathcal{O}_1\} \end{matrix}}{sns_1 \mid sns_2 \xrightarrow{\mathcal{I}_1'\mathcal{I}_2',\mathcal{O}_1\mathcal{O}_2} sns_1' \mid sns_2'} \quad (sns5)$$

$$\frac{sns \xrightarrow{\mathcal{I},\mathcal{O}} sns'}{(l)sns \xrightarrow{\mathcal{I},\mathcal{O} \setminus \{(l,\varnothing),(l,a)\}} (l)sns'} \quad (sns6)$$

Figure 5: Rules for spiking neural systems.

are non-deterministically partitioned into different multisets ($I_{l'}$ for each $l' \in L$) of spikes to be received from each neuron mentioned in L. Note that the neuron can receive at most one spike from each neuron in L, hence $I_{l'} \subseteq \{a\}$.
Rule $(sns2)$ differs from $(sns1)$ since it describes the case of an open neuron that is closing and, therefore, cannot receive any input.
Rules $(sns3)$ and $(sns4)$ describe a computation step performed by a closed neuron, which is opening in the case of $(sns3)$. In these cases the output sent out by the neuron content c is reported in the label of the neuron transition.
Rule $(sns5)$ describes a computation step performed by a juxtaposition of neurons sns_1 and sns_2. Ouput spikes of the two neurons are simply combined. More complex is the handling of input spikes: first of all it has to be checked whether some neurons in sns_1 (sns_2, respectively) might receive some spikes from neurons in sns_2 (sns_1, respectively). This amounts in checking whether in the input of one component and in the output of the other there are spikes associated with the same label l'. If this is the case, then the input and the output spikes must coincide, namely they are both $\{a\}$ or $\varnothing$. If this condition is satisfied, then the information on the input from l' can be forgotten (since there is no longer the need to look for such a neuron in the context) and therefore it

can be omitted in the label of the transition.
Finally, rule ($sns6$) deals with the restriction operation. In this inference rule the transition performed by a system with a restriction is obtained from the transition performed by the system in which the restriction has been omitted, by removing from the label information on the spikes sent by the neuron whose label is restricted. This avoids other neurons added by further juxtapositions to receive spikes from the neuron whose label is restricted.

Proposition 2. *Given a complete spiking neural system sns and a transition* $sns \xrightarrow{\mathcal{I},\mathcal{O}} sns'$*, it holds that* $\mathcal{I} = \varnothing$ *and that also* sns' *is complete.*

Proof. The fact that sns' is complete is immediate. If, by contraddiction, $\mathcal{I} \neq \varnothing$, then $\mathcal{I}$ contains at least a pair (l, I_l). In this case the rules in Fig. 5 ensure that $\mathcal{O}$ does not contain any pair (l, O), for any O, and that sns has not any spiking neural system with label l as a subterm. Therefore, sns is not complete, which leads to a contraddiction.

Example 3. As an example of transition derivation let us consider again the term S'_{lcm} defined above and corresponding to the SNP system in Fig. 1. Let us show how the transition corresponding to the first computation step of the system is derived. A transition for neuron l_1 is derived by applying inference rules $(c6), (c8), (u1)$ and $(sns1)$, and it is the following:

$$[\, aa \to \lambda,\, a \to a; n_1{-}2,\, \varnothing,\, 0\,]_{l_1}^{\{l_2,l_5\}} \xrightarrow{\{(l_2,\varnothing),(l_5,a)\},\{(l_1,\varnothing)\}} [\, aa \to \lambda,\, a \to a; n_1{-}2,\, a,\, 0\,]_{l_1}^{\{l_2,l_5\}}$$

The transition of neuron l_2 is derived by applying inference rules $(c6)$ and $(sns1)$ and it is the following:

$$[\, a \to a\,,\, \varnothing\,,\, 0\,]_{l_2}^{\{l_1\}} \xrightarrow{\{(l_1,\varnothing)\},\{(l_2,\varnothing)\}} [\, a \to a\,,\, \varnothing\,,\, 0\,]_{l_2}^{\{l_1\}}$$

Transitions of neurons l_3 and l_4 are similar to those of neurons l_2 and l_1, respectively. Now, the transition of $(l_2)([\, aa \to \lambda\,,\, a \to a; n_1 - 2\,,\, \varnothing\,,\, 0\,]_{l_1}^{\{l_2,l_5\}} \mid [\, a \to a\,,\, \varnothing\,,\, 0\,]_{l_2}^{\{l_1\}})$ (let us call this subterm $S_{1,2}$) is obtained from transitions of neurons l_1 and l_2 by applying $(sns5)$, which removes $(l_1, \varnothing)$ and $(l_2, \varnothing)$ from the input, and $(sns6)$, which removes $(l_2, \varnothing)$ from the output. The result is the following transition:

$$S_{1,2} \xrightarrow{\{(l_5,a)\},\{(l_1,\varnothing)\}} (l_2)([\, aa \to \lambda\,,\, a \to a; n_1 - 2\,,\, a\,,\, 0\,]_{l_1}^{\{l_2,l_5\}} \mid [\, a \to a\,,\, \varnothing\,,\, 0\,]_{l_2}^{\{l_1\}})$$

Subterm $(l_3)([\, aa \to \lambda\,,\, a \to a; n_2 - 2\,,\, \varnothing\,,\, 0\,]_{l_4}^{\{l_3,l_5\}} \mid [\, a \to a\,,\, \varnothing\,,\, 0\,]_{l_3}^{\{l_4\}})$ (let us call it $S_{3,4}$) performs a transition similar to that of $S_{1,2}$.

Neuron l_5 performs a transition derived by applying $(c2), (c4), (c8), (u2)$ and $(sns1)$. The transition is as follows:

$$[\, aa/a \to a\,,\, a \to \lambda\,,\, aa\,,\, 0\,]_{l_5}^{\{l_1,l_4\}} \xrightarrow{\{(l_1,\varnothing),(l_4,\varnothing)\},\{(l_5,a)\}} [\, aa/a \to a\,,\, a \to \lambda\,,\, a\,,\, 0\,]_{l_5}^{\{l_1,l_4\}}$$

Finally, the transition of the whole term can be derived from that of neuron l_5 and of subterms $S_{1,2}$ and $S_{3,4}$ by applying inference rules $(sns5)$ and $(sns6)$, with the following result:

$$S'_{lcm} \xrightarrow{\varnothing,\{(l_5,a)\}} S''_{lcm}$$

where S''_{lcm} is the following term:

$$
\begin{aligned}
S''_{lcm} \ ::= \ & (l_1)(l_4)\big(\\
& (l_2)\big([\,aa \to \lambda\,,\, a \to a; n_1 - 2\,,\, a\,,\, 0\,]_{l_1}^{\{l_2,l_5\}} \mid [\,a \to a\,,\, \varnothing\,,\, 0\,]_{l_2}^{\{l_1\}}\big) \mid \\
& (l_3)\big([\,a \to a\,,\, \varnothing\,,\, 0\,]_{l_3}^{\{l_4\}} \mid [\,aa \to \lambda\,,\, a \to a; n_2 - 2\,,\, a\,,\, 0\,]_{l_4}^{\{l_3,l_5\}}\big) \mid \\
& [\,aa/a \to a\,,\, a \to \lambda\,,\, a\,,\, 0\,]_{l_5}^{\{l_1,l_4\}} \\
& \big)\,.
\end{aligned}
$$

4. Behavioural preorders and equivalences

In this section we introduce some notions of behavioural equivalence for SNP algebra terms. The concept of equivalence of systems we consider here is well established in the literature on reactive and interactive systems, and on concurrency theory. In these fields, systems are assumed to be placed in some environment and to be able to interact with such an environment. The equivalence is hence based on the interactions with the environment that are performed step-by-step by the considered systems. Two systems are hence equivalent when they are able to interact in the same way with the environment they are placed in at each step of the computation. Moreover, it is usually required that if two equivalent systems are plugged in the same context then we obtain two bigger systems being, in turn, equivalent.

We remark that all SOS rules in Figs. 2–5 respect the well known *de Simone* format [17], namely that they are in the form:

$$\frac{\{x_i \xrightarrow{\alpha_i} y_i \,|\, i \in I\}}{f(x_1, \ldots, x_{ar(f)}) \xrightarrow{\alpha} t}$$

where f is an operation with arity $ar(f)$, $x_1, \ldots, x_{ar(f)}$ and $\{y_i \mid i \in I\}$ are variables that can be instantiated with any term, I is any subset of $\{1, \ldots, ar(f)\}$, the variables x_i and y_j are all distinct and the only variables that occur in the rule, and, finally, t is any term that does not contain any variable x_i, for $i \in I$, and has no multiple occurrence of variables.

In this section we consider some well known notions of behavioural *preorder* and *equivalence* defined in the literature over the LTS model (see [16] for a survey). Let us recall that a preorder is a reflexive and transitive relation, and an equivalence is a symmetric preorder. Moreover, the largest equivalence contained in a preorder is called the *kernel* of the preorder. As usual, given an LTS, we shall write $s \overset{\ell}{\not\to}$ if $s \xrightarrow{\ell} s'$ holds for no s', and $s \not\to$ if $s \overset{\ell}{\not\to}$ for all $\ell \in \mathcal{L}$. Moreover, for a state $s \in \mathcal{S}$, we denote by $\mathsf{Initials}(s)$ the set $\{\ell \in \mathcal{L} \mid \exists s'.\ s \xrightarrow{\ell} s'\}$.

Definition 5. Let $(\mathcal{S}, \mathcal{L}, \{\xrightarrow{\ell} \mid \ell \in \mathcal{L}\})$ be an LTS. A relation $R \subseteq \mathcal{S} \times \mathcal{S}$

- is a *simulation* if, for each pair $s_1 \, R \, s_2$, if $s_1 \xrightarrow{\ell} s'_1$ then there is a transition $s_2 \xrightarrow{\ell} s'_2$ such that $s'_1 \, R \, s'_2$;

$$\begin{tikzcd}
& \sqsubseteq_S \arrow[rd] & \\
\sqsubseteq_{RS} \arrow[ru] \arrow[rd] & & \sqsubseteq_T \\
& \sqsubseteq_{RT} \arrow[ru] \arrow[r] & \sqsubseteq_F
\end{tikzcd}$$

Figure 6: Relations between preorders.

- is a *ready simulation* if it is a simulation and, for each pair $s_1 \, R \, s_2$, if $s_1 \stackrel{\ell}{\not\rightarrow}$ then $s_2 \stackrel{\ell}{\not\rightarrow}$;
- is a *ready trace preorder* if, for each pair $s_1 R s_2$, any ready trace of s_1 is a ready trace of s_2 (a sequence $L_0 \ell_1 L_1 \dots \ell_n L_n$ with $L_i \subseteq \mathcal{L}$ and $\ell_i \in \mathcal{L}$ is a *ready trace* of a state s_0 if $s_0 \xrightarrow{\ell_1} s_1 \xrightarrow{\ell_2} \dots s_{n-1} \xrightarrow{\ell_n} s_n$ and $\mathsf{Initials}(s_i) = L_i$ for $i = 0, \dots, n$);
- is a *failure preorder* if, for each pair $s_1 R s_2$, any failure of s_1 is a failure of s_2 (a pair $(\ell_1 \dots \ell_n, L)$ with $\ell_1 \dots \ell_n \in \mathcal{L}$ and $L \subseteq \mathcal{L}$ is a *failure* of a state s if $s \xrightarrow{\ell_1} \dots \xrightarrow{\ell_n} s'$ for some state s' such that $\mathsf{Initials}(s') \cap L = \varnothing$);
- is a *trace preorder* if, for each pair $s_1 R s_2$, any trace of s_1 is a trace of s_2 (a sequence $\ell_1 \dots \ell_n$ with $\ell_i \in \mathcal{L}$ is a *trace* of a state s_0 if $s_0 \xrightarrow{\ell_1} \dots \xrightarrow{\ell_n} s_n$ for some state s_n).

The relations in Def. 5 are preorders. Intuitively, two states s and t are related by a preorder (resp. an equivalence that is the kernel of a preorder) if the behaviour of s is step-by-step simulated by (resp. equivalent to) the behaviour of t, provided that some details of the behaviours of s and t are abstracted away. These details depend on the considered preorder (resp. equivalence) and correspond to the part of behaviour that is not visible to an external observer.

As usual, we denote with $\sqsubseteq_{RS}$ (resp.: $\sqsubseteq_S$) the union of all ready simulations (resp.: simulations), which, in turn, is a ready simulation (resp. simulation). The kernel of $\sqsubseteq_{RS}$ and the kernel of $\sqsubseteq_S$ coincide, is called *bisimulation*, and is denoted by $\approx$. We denote by $\sqsubseteq_{RT}$ (resp.: $\sqsubseteq_F$, $\sqsubseteq_T$) the union of all ready trace preorders (resp.: failure preorders, trace preorders), which, in turn, is a ready trace preorder (resp.: failure preorder, trace preorder), and by $\approx_{RT}$ (resp.: $\approx_F$, $\approx_T$) its kernel. It is well known (see, e.g., [16]) that the preorders in Def. 5 are structured by the hierarchy of inclusions shown in Fig. 6 (where $\rightarrow$ stands for $\subseteq$).

A usual requirement for a preorder (resp. equivalence) is to be a precongruence (resp. congruence). This is essential for substitution of programs with equivalent ones and to develop an axiomatic framework.

Definition 6. A preorder (resp. equivalence) $R \subseteq \mathcal{S} \times \mathcal{S}$ is called a *precongruence* (resp. *congruence*) iff, for each operation f with arity n and pairs $s_1 R s'_1, \dots s_n R s'_n$, it holds that $f(s_1, \dots, s_n) R f(s'_1, \dots, s'_n)$.

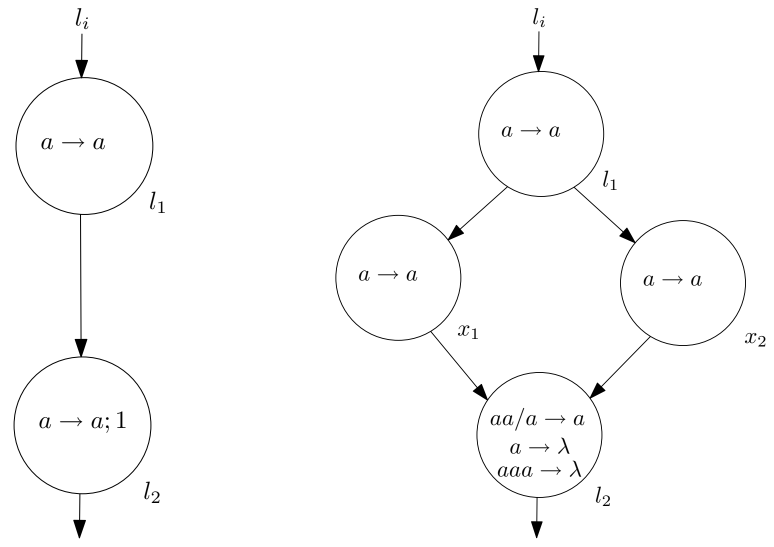

Figure 7: Example of equivalent SNP systems.

We show that SNP algebra operations preserve the relations in Def. 5.

Theorem 3. *All preorders in Def. 5 are precongruences.*

Proof. In [18] it is proved that trace preorder and failure preorder are precongruences for all calculi defined with SOS rules in de Simone format. In [19] some formats are given which ensure that simulation preorder, ready simulation preorder and ready trace preorder are precongruences. All these formats are less restrictive than de Simone format. □

Corollary 4. *The kernels of all preorders in Def. 5 are congruences.*

Let us consider now an example of equivalent spiking neural systems.

Example 4. Take the SNP systems in Figure 7. Actually, they should be intended as portions of a bigger SNP system, with neurons l_1 and l_2 of both systems receiving spikes from an external neuron l_i and sending spikes to external neurons, respectively. When receiving the spike a from neuron l_i at time t, each of the two systems sends out the spike a at time $t+3$, unless a spike leading to an output at time $t+2$ was already received at time $t-1$. In other words, if two spikes are received subsequently, only the first one will lead to an output after three time units; if three (or four) spikes are received subsequently, only the first and the third will lead to an output, and so on.

The two SNP systems can be represented by the following SNP algebra terms: the system on the left corresponds to

$$S_1 \ ::= \ (l_1)([\,a \to a\,,\,\varnothing\,,\,0\,]_{l_1}^{\{l_i\}} \mid [\,a \to a; 1\,,\,\varnothing\,,\,0\,]_{l_2}^{\{l_1\}})\,,$$

whereas that on the right corresponds to

$$\begin{aligned} S_2 \ ::= \ & (l_1)(x_1)(x_2)\big([\, a \to a \, , \, \varnothing \, , \, 0]_{l_1}^{\{l_i\}} \mid [\, a \to a \, , \, \varnothing \, , \, 0]_{x_1}^{\{l_1\}} \mid \\ & \qquad [\, a \to a \, , \, \varnothing \, , \, 0]_{x_2}^{\{l_1\}} \mid [\, aa/a \to a \, , \, a \to \lambda , \, aaa \to \lambda \, , \, \varnothing \, , \, 0]_{l_2}^{\{x_1,x_2\}}\big) \, . \end{aligned}$$

Now, let us introduce the following notations:

$$\begin{aligned} S_1^{X,Y,t} \ & ::= (l_1)([\, a \to a \, , \, X \, , \, 0]_{l_1}^{\{l_i\}} \mid [\, a \to a ; 1 \, , \, Y \, , \, t \,]_{l_2}^{\{l_1\}}) \\ S_2^{X,Y,Z,W} \ & ::= (l_1)(x_1)(x_2)\big([\, a \to a \, , \, X \, , \, 0]_{l_1}^{\{l_i\}} \mid [\, a \to a \, , \, Y \, , \, 0]_{x_1}^{\{l_1\}} \mid \\ & \qquad [\, a \to a \, , \, Z \, , \, 0]_{x_2}^{\{l_1\}} \mid [\, aa/a \to a \, , \, a \to \lambda , \, aaa \to \lambda \, , \, W \, , \, 0]_{l_2}^{\{x_1,x_2\}}\big) \end{aligned}$$

namely $S_1^{X,Y,t}$ denotes a system with the same structure as S_1, but with X and Y as multisets of spikes in l_1 and l_2, respectively, and with t as timer of l_2. Similarly, $S_2^{X,Y,Z,W}$ denotes a system with the same structure as S_2, but with X, Y, Z and W as multisets of spikes in l_1, x_1, x_2 and l_2, respectively. Note that $S_1 = S_1^{\varnothing,\varnothing,0}$ and $S_2 = S_2^{\varnothing,\varnothing,\varnothing,\varnothing}$.

From the semantic rules, we infer the following transitions:

$$S_1^{X,\varnothing,0} \xrightarrow{\{(l_i,I)\},\{(l_2,\varnothing)\}} S_1^{I,X,0}$$

$$S_1^{X,a,0} \xrightarrow{\{(l_i,I)\},\{(l_2,\varnothing)\}} S_1^{XI,a,1}$$

$$S_1^{X,a,1} \xrightarrow{\{(l_i,I)\},\{(l_2,a)\}} S_1^{I,X,0}$$

for every $X, I \in \{\varnothing, a\}$. We infer also the following transitions:

$$S_2^{X,Y,Y,\varnothing} \xrightarrow{\{(l_i,I)\},\{(l_2,\varnothing)\}} S_2^{I,X,X,YY}$$

$$S_2^{X,Y,Y,aa} \xrightarrow{\{(l_i,I)\},\{(l_2,a)\}} S_2^{I,X,X,YYa}$$

$$S_2^{X,Y,Y,a} \xrightarrow{\{(l_i,I)\},\{(l_2,\varnothing)\}} S_2^{I,X,X,YY}$$

$$S_2^{X,Y,Y,aaa} \xrightarrow{\{(l_i,I)\},\{(l_2,\varnothing)\}} S_2^{I,X,X,YY}$$

for every $X, Y, I \in \{\varnothing, a\}$.

Let R be the least relation on terms satisfying the following axioms:

$$S_1^{X,Y,0} \, R \, S_2^{X,Y,Y,\varnothing} \qquad S_1^{X,a,1} \, R \, S_2^{X,Y,Y,aa}$$

$$S_1^{X,Y,0} \, R \, S_2^{X,Y,Y,a} \qquad S_1^{X,Y,0} \, R \, S_2^{X,Y,Y,aaa}$$

for every $X, Y \in \{\varnothing, a\}$. It holds that R is a bisimulation, namely $R \subseteq \approx$.

In principle, the considered equivalences (and preorders) could be used also to compare the behaviour of neuron contents. However, the semantics of neuron contents we have defined does not allow, at the moment, the equivalences to work as expected. For example, let us consider the following neuron contents:

$$\begin{aligned} c_1 &::= (aa + aaa/a \to a\,,\, \varnothing\,,\, 0) \\ c_2 &::= (aaa + aa/a \to a\,,\, \varnothing\,,\, 0) \\ c_3 &::= (aa/a \to a\,,\, aaa/a \to a\,,\, \varnothing\,,\, 0) \end{aligned}$$

Any reasonable behavioural equivalence on neuron contents should consider c_1, c_2 and c_3 as equivalent. However, transitions performed by c_1, c_2 and c_3 have syntactically different labels, since labels contain the regular expressions associated with the rules that are, in this case, syntactically different.

In order to solve this problem we should slightly modify the definition of the semantics as follows: label component U, namely the set of all regular expressions associated with rules, should be replaced by a language that is the union of the languages denoted by such regular expressions. Similarly, label component E, namely the regular expression associated with the applied rule, should be replaced by $L(E)$.

By replacing regular expressions with languages in the transition labels, we obtain that neuron contents with a different number of rules or with rules having syntactically different regular expressions perform the same transitions if the regular expressions denote the same languages. This would be the case, for instance, of c_1, c_2 and c_3. However, the use of languages in place of regular expressions in transition labels is less intuitive and gives some advantages only when dealing with behavioural equivalences. This motivates our choice of not using languages in the definition of the semantics. Moreover, note that the current definition of the semantics is such that neurons $[\,c_1 \cup c\,]_l^L, [\,c_2 \cup c\,]_l^L$ and $[\,c_3 \cup c\,]_l^L$, for any c, l and L, are equivalent with respect to any behavioural equivalence, even if c_1, c_2 and c_3 are not. In other words, the current semantics ensures that c_1, c_2 and c_3, when inserted in the same context constituting a neuron, turn out to be equivalent, as it was the intuition.

5. Conclusions

In the paper we have given a formal compositional semantics for SNP systems by following a Structural Operational Semantics approach. We have introduced a process algebra, the SNP algebra, whose terms represent either neuron contents, or neurons, or juxtaposition of neurons. The algebra is equipped with a semantics, given as a labelled transition system. We have considered some known behavioural equivalences and provided their definition based on the given semantics. We have also proved that such equivalences are congruences.

A further step would be the development of axiomatic semantics characterizing equivalent SNP systems. Moreover, the given semantics could be easily adapted to described the behaviour of variants of SNP systems such as *SNP systems with extended rules* [20] and *asynchronous SNP systems* [21].

References

[1] G. Păun, G. Rozenberg, A guide to membrane computing, Theor. Comput. Sci. 287 (2002) 73–100.

[2] G. Păun, Membrane computing. An introduction, Springer, 2002.

[3] C. Zandron, C. Ferretti, G. Mauri, NP-Complete problems using P systems with active membranes, in: Proc. of Unconventional Models of Computation (UMC'2K), Springer-Verlag London, 2001, pp. 289–301.

[4] P. Bottoni, C. Martin-Víde, G. Păun, G. Rozenberg, Membrane systems with promoters/inhibitors, Acta Informatica 38 (2002) 695–720.

[5] A. Păun, G. Păun, The power of communication: P systems with symport/antiport, New Generation Computing 20 (2002) 295–305.

[6] C. Martín-Vide, J. Pazos, G. Păun, A. Rodríguez-Patón, Tissue P systems, Theor. Comput. Sci. 296 (2003) 295–326.

[7] M. Ionescu, G. Păun, T. Yokomori, Spiking neural P systems, Fund. Inform. 71 (2006) 279–308.

[8] G. Plotkin, A structural approach to operational semantics, J. Log. Algebr. Program. 60-61 (2004) 17–139.

[9] O. Andrei, G. Ciobanu, D. Lucanu, A rewriting logic framework for operational semantics of membrane systems, Theor. Comp. Sci. 373 (2007) 163–181.

[10] N. Busi, Using well–structured transition systems to decide divergence for catalytic P systems, Theor. Comp. Sci. 372 (2007) 125–135.

[11] N. Busi, Causality in membrane systems, in: Proc. of the 8th Workshop on Membrane Computing (WMC 2007), Vol. 4860 of Lecture Notes in Computer Science, Springer, 2007, pp. 169–171.

[12] R. Freund, S. Verlan, A formal framework for static (tissue) P systems, in: Proc. of the 8th Workshop on Membrane Computing (WMC 2007), Vol. 4860 of Lecture Notes in Computer Science, Springer, 2007, pp. 271–284.

[13] R. Barbuti, A. Maggiolo-Schettini, P. Milazzo, S. Tini, Compositional semantics and behavioral equivalences for P systems, Theor. Comput. Sci. 395 (2008) 77–100.

[14] R. Barbuti, A. Maggiolo-Schettini, P. Milazzo, S. Tini, A P systems flat form preserving step-by-step behaviour, Fund. Inform. 87 (2008) 1–34.

[15] R. Keller, Formal verification of parallel programs, CACM 19 (1976) 371–384.

[16] L. Aceto, W. J. Fokkink, C. Verhoef, Handbook of Process Algebra, Elsevier, 2001, Ch. Structural operational semantics, pp. 197–292.

[17] R. de Simone, High level synchronization devices in Meije-SCCS, Theor. Comput. Sci. 37 (1985) 245–267.

[18] F. Vaandrager, On the relationship between process algebra and input/output automata, in: Proc. of the 6th Annual IEEE Symposium on Logic in Computer Science (LICS'91), IEEE Press, 1991, pp. 387–398.

[19] R. J. van Glabbeek, Full abstraction in structural operational semantics, in: Proc. of the 3rd International Conference on Algebraic Methodology and Software Technology (AMAST'93), Springer, 1993, pp. 77–84.

[20] H. Chen, M. Ionescu, T.-O. Ishdorj, A. Păun, G. Păun, M. J. Pérez-Jiménez, Spiking neural P systems with extended rules: Universality and languages, Nat. Comput. 7 (2008) 147–166.

[21] M. Cavaliere, O. H. Ibarra, G. Păun, O. Egecioglu, M. Ionescu, S. Woodworth, Asynchronous spiking neural P systems, Theor. Comput. Sci. 410 (2009) 2352–2364.