Spiking Neural P System Models as Formal Framework Models

Ren Tristan A. de la Cruz

December 14, 2020

1 Background

2 Preliminaries

The following sets will be commonly used throughout the document: $\mathbb{N} = \{0, 1, 2, 3, ...\}$, $\mathbb{N}_\infty = \mathbb{N} \cup \{\infty\}$, $[1..n] = \{1, ..., n\}$, $2^{[1..n]} = \mathcal{P}([1..n])$ (power set of $[1..n]$).

Let V be a set of symbols called an *alphabet*. A *string* or *word* over V is a sequence of symbols from V. The *empty string* ϵ is a string without symbols, an empty sequence. A *string language* over V is a set of strings over V. The set of all strings over V is denoted by V^*.

A *multiset* over V is a function of the form $m : V \to \mathbb{N}_\infty$ while a *finite multiset* over V is a function of the form $m : V \to \mathbb{N}$. If m is a multiset over V and $a \in V$, $m(a)$ denotes the number of occurrences of symbol a in multiset m. If $V = \{v_1, ..., v_k\}$ and m is a finite multiset over V, m can be represented by the string ${v_1}^{m(v_1)} \cdots {v_k}^{m(v_k)}$. The size of a multiset m over V is $|m| = \sum_{v \in V} m(v)$. An *empty multiset* $\emptyset$ is any multiset of size 0. A *multiset language* over V is a set of multisets over V. The set of all multisets over V is denoted by V°.

Let $V = \{v_1, ..., v_k\}$, m be the multiset ${v_1}^{m(v_1)} \cdots {v_k}^{m(v_k)}$ and n be the multiset ${v_1}^{n(v_1)} \cdots {v_k}^{n(v_k)}$. $m \subseteq n$ if and only if for all $v \in V$ $m(v) \leq n(v)$. $m + n$ is the multiset ${v_1}^{m(v_1)+n(v_1)} \cdots {v_k}^{m(v_k)+n(v_k)}$. If $m \subseteq n$, $n - m$ is the multiset ${v_1}^{n(v_1)-m(v_1)} \cdots {v_k}^{n(v_k)-m(v_k)}$

The set of n-vectors whose components are finite multisets over V is denoted by $V^{\circ n}$. Let $X = (x_1, ..., x_n), Y = (y_1, ..., y_n) \in V^{\circ n}$. $X \subseteq Y$ if and only if $x_i \subseteq y_i$ for $1 \leq i \leq n$. $X + Y = (x_1 + y_1, ..., x_n + y_n)$. If $X \subseteq Y$, $Y - X = (y_1 - x_1, ..., y_n - x_n)$. Aside from denoting the empty multiset, $\emptyset$ will also denote a vector of empty multisets. i.e. $\emptyset = (\emptyset, ..., \emptyset)$. If the context is not clear, we will specify if $\emptyset$ means an empty multiset or a vector of empty multisets.

A *family of languages* is a set of languages. It can either be a family of string languages or a family of multiset languages. The notations $\mathscr{F}$ and $\mathscr{F}^\circ$ will be used for a family of string languages and a family of multiset languages, respectively. The notation $\mathscr{F}(V)$ will be used for a family of string languages over alphabet V while $\mathscr{F}(V)^\circ$ will be used for a family of multiset languages over alphabet V. For example, REG is the set regular string languages, $REG(V)$ is the set of regular string languages over V, REG° is the set of regular multiset languages, and $REG(V)^\circ$ is the set of regular multiset languages over V.

3 Formal Framework for Spiking Neural P Systems

Definition 1. **[Configuration]** An n-degree *configuration* $C = (u_1, ..., u_n)$ over alphabet V is an n-vector of multisets over V. A configuration C is called a *finite configuration* if all the components are finite multisets.

A configuration is referred to as a *full configuration* if we specifically want to specify that the configuration can contain infinite multisets.

Definition 2. **[Interaction Rule]** An n-degree *interaction rule* over alphabet V is the construct $(X \to Y; K)$ where $X = (x_1, ..., x_n)$ and $Y = (y_1, ..., y_n)$ are n-vectors of multisets over V and $K = (k_1, ..., k_n)$ is an n-vector of multiset languages.

$$[x_1]_1 \cdots [x_n]_n \to [y_1]_1 \cdots [y_n]_n \ ; \ [k_1]_1 \cdots [k_n]_n$$

The multiset languages $k_1, ..., k_n$ are called *control languages*.

$$(1, x_1) \cdots (n, x_n) \to (1, y_1) \cdots (n, y_n) \ ; \ (1, k_1) \cdots (n, k_n)$$

if x_i or y_i is $\emptyset$ the term (i, x_i) and (i, y_i) can be removed
if x_i (or y_i) is empty, the term $[x_i]_i$ (or $[y_i]_i$) can be removed.

Definition 3. **[Rule Eligibility]** Let $C = (u_1, ..., u_n)$ be an n-degree configuration over V and $r = ((x_1, ..., x_n) \to (y_1, ..., y_n); (k_1, ..., k_n))$ be an n-degree rule over V, rule r is *eligible* with respect to configuration C if the following conditions hold: (1) for all x_i, $x_i \subseteq u_i$ and (2) for all u_i, $u_i \in k_i$.

Definition 4. **[Applicability of a Multiset of Rules]** Let R' be a multiset of n-degree rules over V and $C = (u_1, ..., u_n)$ be an n-degree configuration over V, R' is *applicable* to configuration C if the following conditions hold: (1) each rule $(X_j \to Y_j; K_j) \in R'$ is eligible with respect to configuration C and (2)

$$X = \left(\sum_{(X_j \to Y_j; K_j) \in R'} X_j \right) \subseteq C.$$

Definition 5. **[Application of a Multiset of Rules]** If R' is multiset of rules applicable to configuration $C = (u_1, ..., u_n)$, *applying* R' to configuration C means producing a new configuration C' which is defined as:

$$C' = Apply(R', C) = C - \left(\sum_{(X_j \to Y_j; K_j) \in R'} X_j \right) + \left(\sum_{(X_j \to Y_j; K_j) \in R'} Y_j \right).$$

The *application function* $Apply(R', C)$

Definition 6. **[Network of Cells]** A $\mathscr{F}$-controlled *network of cells* of degree n is the construct

$$\Pi = (n, V, W, c_{in}, c_{out}, R)$$

where

- n is the number of cells;
- V is a finite alphabet;
- $W = (w_1, ..., w_n)$ is the *initial configuration* and $w_i \in V^\circ$ is the multiset associated with cell i.
- $c_{in} \subseteq \{1, ..., n\}$ is the set of *input cells.*
- $c_{out} \subseteq \{1, ..., n\}$ is the set of *output cells.*
- R is the set of interactive rules.

Applicable(Π, C) is the set of multiset of rules (rules from R) applicable to configuration C.

Definition 7. **[Network of Cells with Environment]** An $\mathscr{F}$-controlled *network of cells with environment* of degree n is the construct:

$$\Pi_{inf} = (\Pi, Inf) = ((n, V, W, c_{in}, c_{out}, R), Inf)$$

- The first component Π_{inf} is as defined in Definition 6.
- $Inf = (inf_1, ..., inf_n)$ where $inf_i \subseteq V$ is a *set of symbols occurring infinitely often in cell i.*

Definition 8. **[Derivation Mode]** A *derivation mode* δ is a restriction of the set of applicable rules. For an $\mathscr{F}$-controlled network of cells Π and configuration C, $Appl(\Pi, C, \delta) \subseteq Appl(\Pi, C)$ denotes the set of multisets of rules in Π applicable to configuration C according to derivation mode δ.

Definition 9. **[Network of Cells Working in δ Derivation Mode]** An $\mathscr{F}$-controlled n-degree *network of cells working in δ derivation mode* is the construct:

$$\Pi' = (\Pi, \delta) = ((n, V, W, c_{in}, c_{out}, R), \delta)$$

- The first component Π_{inf} is as defined in Definition 6.
- δ is the derivation mode used.

Applicable(Π, R', δ)
$NC(n, V, \mathscr{F}, \delta)$ is the set of n-degree $\mathscr{F}$-controlled network of cells using alphabet V and working in δ derivation mode.

Definition 10. [Computation of a Network of Cells] A *computation* of a network of cell $\Pi' = ((n, V, W, c_{in}, c_{out}, R), \delta)$ is sequence a $C_0, C_1, C_2, ...$ with the following properties:

- $C_0 = W$
- $C_{i+1} = Apply(\Pi', R', C_i)$ where $R' \in Applicable(\Pi, C_i, \delta)$.

Definition 11. [Input Function] An *input function* for a system $\Pi' = ((n, v, W, c_{in}, c_{out}, R), \delta)$, $\Pi' \in NC(n, V, \mathscr{F}, \delta)$, is a function of the form $Input(\Pi') : \mathbb{N} \to V^{\circ n}$ and fulfills that condition that for all $i \notin c_{in}$ the i-th component of resulting input vector from $V^{\circ n}$ is an empty multiset.

Definition 12. [Computation of a Network of Cells]

- $C_0 = W + Input(\Pi')(0)$
- $C_{i+1} = Apply(\Pi', C_i, R') + Input(\Pi')(i+1)$ where $R' \in Appl(\Pi, C_i, \delta)$.

Definition 13. [Output Function]

4 Spiking Neural P System Models as Formal Framework Network of Cells

4.1 Spiking Neural P System

Definition 14. [Spiking Neural P System] A *spiking neural P sytem* of degree n the construct:

$$\Pi = (O, \sigma_1, ..., \sigma_n, syn, i_o)$$

- $O = \{a\}$ is the singleton alphabet of the system. Symbol a is called a *spike*.
- $\sigma_1, ..., \sigma_n$ are neurons of the form
  - $\sigma_i = (n_i, R_i)$ for $1 \leq i \leq n$.
  - $n_i \in \mathbb{N}$ is initial number of spikes in neuron i.
  - R_i is a finite set of rules of the following two forms:
    * *Spiking Rule:* $E/a^c \to a; d$ where E is regular expression over O, $d \in \mathbb{N}$,$c \in \mathbb{N}\backslash\{0\}$.
    * *Forgetting Rule:* $a^s \to \lambda$ where $s \in \mathbb{N}\backslash\{0\}$ and $\{a^s\} \cap L^\circ(E) = \emptyset$ for all E that are regular expressions of a spiking rules in the same neuron.
- $syn \subseteq [1..n] \times [1..n]$ is the set of *synapses* where $(i, i) \notin syn$ for all $i \in [1..n]$.
- $i_o \in [1..n]$ specifies the *output neuron*.

SNP System' Spiking Rule:

$$[a^c]_i \to [a]_{j_1} \cdots [a]_{j_k}; [L^\circ(E)]_i$$

SNP System's Forgetting Rule

$$[a^c]_i \to \emptyset; [L^\circ(E)]_i$$

SNP Systems with Extended Spiking Rule

$$[a^c]_i \to [a^p]_{j_1} \cdots [a^p]_{j_k}; [L^\circ(E)]_i$$

SNP Systems with Weights

$$[a^c]_i \to [a^{w_{j_1}}]_{j_1} \cdots [a^{w_{j_k}}]_{j_k}; [L^\circ(E)]_i$$

SNP Systems with Extended Rules and Weights

$$[a^c]_i \to [a^{p \cdot w_{j_1}}]_{j_1} \cdots [a^{p \cdot w_{j_k}}]_{j_k}; [L^\circ(E)]_i$$

References