A Semantic Investigation of Spiking Neural P Systems

Gabriel Ciobanu, Eneia Nicolae Todoran

Romanian Academy, TU Cluj-Napoca

19th International Conference on
Membrane Computing
(CMC 19)

Dresden, Germany

September 4–7, 2018

Aim and Contribution

- We present a denotational semantics $[\![\cdot]\!]$ for a language $\mathcal{L}_{SNP}$ inspired by the spiking neural P (SN P) systems [Ionescu, Păun and Yokomori - 2006]
  - At syntactic level $\mathcal{L}_{SNP}$ provides constructions for specifying: neurons and synapses, rules with time delays
  - The denotational semantics $[\![\cdot]\!]$ for $\mathcal{L}_{SNP}$ is designed with metric spaces and continuations
- We provide a Haskell implementation of $[\![\cdot]\!]$

  `http://ftp.utcluj.ro/pub/users/gc/eneia/cmc19`

Aim and Contribution

- SN P systems - a class of P systems inspired from the way neurons communicate by means of spikes [Păun - 2007]
  - Equivalent in computational power to Turing machines
  - Able to solve NP-complete problems in polynomial time

- We investigate the behavior of SN P systems using methods specific of programming languages semantics
  - Syntax of $\mathcal{L}_{SNP}$ is specified in BNF
    - $\mathcal{L}_{SNP}$ constructions are called statements or programs
  - Semantics of $\mathcal{L}_{SNP}$ is described in denotational style

Aim and Contribution

- Our denotational semantics $[\![\cdot]\!]$ describes accurately
  - The structure of SN P systems: neurons, synapses, spikes
  - The behavior of SN P systems:
    - Time delays between firings and spikings
    - Non-deterministic behavior and synchronized functioning
- $[\![\cdot]\!]$ is the first denotational (compositional) semantics for this combination of concepts, specific of SN P systems

Principles of Denotational Sematics (mathematical or Scott-Stratchey semantics)

- *Language constructions denote values from a mathematical domain of interpretation*

$$[\![\cdot]\!] : \mathcal{L} \to \mathbf{D}$$

- *Definitions are compositional*

$$[\![\cdots x_1 \cdots x_2 \cdots]\!] = \cdots [\![x_1]\!] \cdots [\![x_2]\!] \cdots$$

- Various options in designing **D** and $[\![\cdot]\!]$ for a given $\mathcal{L}$
  - Classic (order-theoretic) domains vs metric spaces
  - Direct semantics, continuations

Metric Spaces vs Order-Theoretic Domains

- The purpose of domain theory is to give models for spaces on which to define computable functions [Scott - 1982]

- In classic domains (order-theoretic domains)
  - One works with least fixed points of continuous functions
  - Not all elements are comparable, the order is partial

- Metric spaces employ additional information
  - One can (compare and even) measure the distance between any two elements of a metric space
  - Contracting functions on complete metric spaces have unique fixed points (Banach's theorem)

Metric Spaces vs Order-Theoretic Domains

- Domain theory was initiated by [Scott - 1976, Scott - 1982]
  - Scott's key construction - a solution of the 'equation'

$$\mathbf{D} \cong \mathbf{D} \to \mathbf{D}$$

- We offer a semantic description of SN P systems based on a domain of continuations

$$\mathbf{D} \cong \mathbf{K} \to \mathbf{K} \qquad \mathbf{K} \cong \cdots \mathbf{D} \cdots$$

- Following [De Bakker and De Vink - 1996] we employ the mathematical methodology of metric semantics
  - Traditional (direct) concurrency semantics may not work for the complex interactions specific of MC and SN P systems

Continuation Semantics for Concurrent Languages

- In Continuation-Passing Style (CPS) control is passed explicitly in the form of continuations [Appel - 2007]

- We need a domain of continuations which can store computations (between firings and spikings) in CSC style [Todoran - 2000, Ciobanu & Todoran - 2014]

$$\mathbf{D} \cong \mathbf{K} \to \mathbf{K} \qquad\qquad \mathbf{K} \cong \cdots \mathbf{D} \cdots$$

- In previous work we investigated MC concepts by using a simple domain of continuations

G. Ciobanu and E.N. Todoran, Denotational Semantics of Membrane Systems by using Complete Metric Spaces, Theor. Comput. Sci., 2017.

Syntax of $\mathcal{L}_{SNP}$

Definition (Syntax of $\mathcal{L}_{SNP}$)

(a) *(Statements)* $x(\in X) ::= a \mid \mathrm{send}(y,\xi) \mid x \parallel x$

$y(\in Y) ::= a \mid y \parallel y$ *(obviously, $Y \subseteq X$)*

(b) *(Rules)* $r(\in Rs) ::= r_\epsilon \mid \varrho, r$

$\varrho(\in R) ::= E/w \to x; t \mid w \to \lambda,$

(E is a regular expression over O, $w \neq []$, $t \geq 0, t \in \mathbb{N}$)

(c) *(Neuron declarations)*

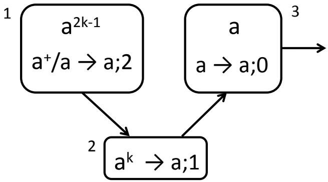
$d(\in ND) ::= \mathrm{neuron}\, N\,\{\, r \mid \xi \,\}$ $D(\in NDs) ::= d \mid d, D$

(d) *(Programs)* $\rho(\in \mathcal{L}_{SNP}) ::= D, x$ *(x executed by first neuron in D)*

- $(a \in)O$ - alphabet of spikes/objects (several types of spikes)
- $(N \in)Nname$ - set of neuron names
- $(w \in)W = [O]$ - set of multisets over O
- $(\xi \in)\Xi = \mathcal{P}_{fin}(Nname)$ - finite sets of neuron names
- Extended rules - a statement x is able to produce more than one spike
- $\mathrm{send}(y,\xi)$ is specific of $\mathcal{L}_{SNP}$
  (Instead of W and Ξ we could use O^* and $Nname^*$)

An $\mathcal{L}_{SNP}$ program and its intuitive behavior

$$\rho_1 = (D_1, x_1)$$
$$x_1 = \mathrm{send}(\langle a^{2k-1}\rangle, \{N_1\}) \parallel \mathrm{send}(a, \{N_3\})$$
$$\begin{aligned} D_1 = \; & \text{neuron } N_0 \{ r_\epsilon \mid \{N_1, N_2, N_3\} \}, \\ & \text{neuron } N_1 \{ a^+/[a] \to a; 2 \mid \{N_2\} \}, \\ & \text{neuron } N_2 \{ [a^k] \to a; 1 \mid \{N_3\} \}, \\ & \text{neuron } N_3 \{ [a] \to a; 0 \mid \{N_0\} \} \end{aligned}$$



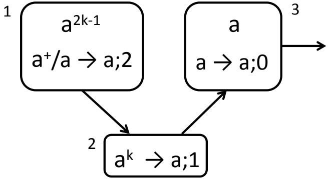
[Ionescu, Păun and Yokomori – 2006]
SN P system Π_1

$\{(N_0, [\,]), (N_1, [a, a, a]), (N_2, [\,]), (N_3, [a])\}$
$\Rightarrow \{(N_0, [a]), (N_1, [a, a, a]), (N_2, [\,]), (N_3, [\,])\}$
$\Rightarrow \{(N_0, [a]), (N_1, [a, a, a]), (N_2, [\,]), (N_3, [\,])\}$
$\Rightarrow \{(N_0, [a]), (N_1, [a, a]), (N_2, [a]), (N_3, [\,])\}$ $\Leftarrow k = 2$
$\Rightarrow \{(N_0, [a]), (N_1, [a, a]), (N_2, [a]), (N_3, [\,])\}$
$\Rightarrow \{(N_0, [a]), (N_1, [a, a]), (N_2, [a]), (N_3, [\,])\}$
$\Rightarrow \{(N_0, [a]), (N_1, [a]), (N_2, [a, a]), (N_3, [\,])\}$
$\Rightarrow \{(N_0, [a]), (N_1, [a]), (N_2, [a, a]), (N_3, [\,])\}$
$\Rightarrow \{(N_0, [a]), (N_1, [a]), (N_2, [\,]), (N_3, [a])\}$
$\Rightarrow \{(N_0, [a, a]), (N_1, [\,]), (N_2, [a]), (N_3, [\,])\}$

The output neuron N_3 spikes in steps 2 and 10
The number computed by this $\mathcal{L}_{SNP}$ program is
$3k + 2 = 8$
(same as in [Ionescu, Păun and Yokomori - 2006])

Two behavioraly equivalent $\mathcal{L}_{SNP}$ programs

$$\begin{aligned}
\rho_1 &= (D_1, x_1)\\
x_1 &= \text{send}(\langle a^{2k-1}\rangle, \{N_1\}) \parallel \text{send}(a, \{N_3\})\\
D_1 &= \ \text{neuron } N_0 \{ r_\epsilon \mid \{N_1, N_2, N_3\} \},\\
&\quad\ \text{neuron } N_1 \{ a^+/[a] \to a; 2 \mid \{N_2\} \},\\
&\quad\ \text{neuron } N_2 \{ [a^k] \to a; 1 \mid \{N_3\} \},\\
&\quad\ \text{neuron } N_3 \{ [a] \to a; 0 \mid \{N_0\} \}
\end{aligned}$$



[Ionescu, Păun and Yokomori – 2006]
SN P system Π_1

$$\begin{aligned}
\rho'_1 &= (D'_1, x'_1)\\
x'_1 &= \text{send}(\langle a^{2k-1}\rangle, \{N_1\}) \parallel \text{send}(a, \{N_3\})\\
D'_1 &= \ \text{neuron } N_0 \{ r_\epsilon \mid \{N_1, N_2, N_3\} \},\\
&\quad\ \text{neuron } N_1 \{ a^+/[a] \to \text{send}(a, \{N_2\}); 2 \mid \{N_2, N_3\} \},\\
&\quad\ \text{neuron } N_2 \{ [a^k] \to \text{send}(a, \{N_3\}); 1 \mid \{N_1, N_3\} \},\\
&\quad\ \text{neuron } N_3 \{ [a] \to a; 0 \mid \{N_0\} \}
\end{aligned}$$

Observables and final semantic domain [De Bakker and De Vink - 1996]

- Final semantic domain **P**, **Q** (linear time)

$$(p \in)\mathbf{P} = \mathcal{P}_{nco}(\mathbf{Q}) \qquad (q \in)\mathbf{Q} \cong \{\epsilon\} + (\Omega \times \frac{1}{2} \cdot \mathbf{Q})$$

- Set of observables Ω

$$(\omega \in)\Omega = \{\omega \mid \omega \in \mathcal{P}_{nfin}(Nname \times W),\ \nu(\omega)\}$$

- Nondeterministic choice operator $\oplus : (\mathbf{P} \times \mathbf{P}) \xrightarrow{1} \mathbf{P}$

$$p_1 \oplus p_2 = \{q \mid q \in p_1 \cup p_2, q \neq \epsilon\} \cup \{\epsilon \mid \epsilon \in p_1 \cap p_2\}$$

Remark

$\oplus$ is associative, commutative and idempotent

Computations and continuations [America and Rutten - 1989]

$$(\varphi \in)\mathbf{D} \cong \mathbf{K} \xrightarrow{1} \mathbf{K}$$

$$(\phi \in)\mathbf{Den} = \{d_0\} + \mathbf{D}$$

$$
\begin{array}{rcl}
(\kappa \in)\mathbf{K} & = & \Gamma \xrightarrow{1} \mathbf{P} \\
(\gamma \in)\Gamma & = & \{\!|\Sigma|\!\} \\
(\sigma \in)\Sigma & = & \mathbf{Open} + \mathbf{Closed} \\
\mathbf{Open} & = & \Xi \times W \\
\mathbf{Closed} & = & \Xi \times W \times \mathbb{N} \times W \times \frac{1}{2} \cdot \mathbf{D}
\end{array}
$$

- Continuations
- Configurations
- States (of neurons)

$\{\!|\Sigma|\!\} \stackrel{not.}{=} \Xi \times (Nname \to \Sigma)$ - Multisets / Bags (of states)

$[\gamma \mid N \mapsto \sigma]$ - Update (state of neuron N in γ with σ)

Continuation semantics for parallel composition

Definition (Semantics of $\|$ in continuation semantics)

We define $\|: (\mathbf{D} \times \mathbf{D}) \xrightarrow{1} \mathbf{D}$, $\lfloor : (\mathbf{D} \times \mathbf{D}) \xrightarrow{1} \mathbf{D}$ *by:*

$$\varphi_1 \| \varphi_2 = \lambda\kappa \,.\, \lambda\gamma \,.\, ((\varphi_1 \lfloor \varphi_2)(\kappa)(\gamma) \oplus (\varphi_2 \lfloor \varphi_1)(\kappa)(\gamma))$$

$$\varphi_1 \lfloor \varphi_2 = \varphi_1 \circ \varphi_2 \quad \text{('}\circ\text{' is function composition operator)}$$

Remarks

- $\|$ *and* $\lfloor$ *are well-defined and nonexpansive in both args*
- $\lfloor$ *is associative*
- $\|$ *is commutative (because* $\oplus$ *is commutative)*

Semantics of $\mathcal{L}_{SNP}$ statements

Definition (Denotational semantics $[\![\cdot]\!] : X \to Alpha \to \mathbf{D}$)

$$
\begin{array}{rcl}
[\![a]\!](\alpha) & = & \lambda\kappa \,.\, \lambda\gamma \,.\, \kappa(send(a, \alpha, \gamma)), \\
[\![\, \text{send}(y, \xi) \,]\!](\alpha) & = & [\![y]\!](\xi \Cap \alpha), \\
[\![x_1 \parallel x_2]\!](\alpha) & = & [\![x_1]\!](\alpha) \parallel [\![x_2]\!](\alpha).
\end{array}
$$

- $(\alpha \in) Alpha = Nname \times \Theta$
- $(\theta \in) \Theta = \{\text{all}\} \cup \Xi$
- $\Cap : (\Xi \times Alpha) \to Alpha$
  $\xi \Cap (N, \text{all}) = (N, \xi)$
  $\xi \Cap (N, \xi') = (N, \xi \cap \xi')$

The operation *send* : $(O \times Alpha \times \Gamma) \to \Gamma$

$$
\begin{aligned}
& send(a, (N, \text{all}), \gamma) = \\
& \qquad \text{let } \{N_1, \ldots, N_i\} = nbs(N, \gamma) \\
& \qquad \text{in } [\gamma \mid N_1 \mapsto add(a, \gamma(N_1)) \mid \cdots \mid N_i \mapsto add(a, \gamma(N_i))], \\
& send(a, (N, \xi), \gamma) = \\
& \qquad \text{let } \{N_1, \ldots, N_i\} = nbs(N, \gamma) \cap \xi \\
& \qquad \text{in } [\gamma \mid N_1 \mapsto add(a, \gamma(N_1)) \mid \cdots \mid N_i \mapsto add(a, \gamma(N_i))]
\end{aligned}
$$

$$
\begin{aligned}
& add(a, (\xi, w)) = (\xi, w \uplus [a]) \\
& add(a, (\xi, w, t, w_r, \varphi)) = (\xi, w, t, w_r, \varphi)
\end{aligned}
$$

Compositional reasoning with continuations

Proposition

(a) $[\![x_1]\!](\alpha_1) \parallel [\![x_2]\!](\alpha_2) = [\![x_1]\!](\alpha_1) \lfloor [\![x_2]\!](\alpha_2) = [\![x_2]\!](\alpha_2) \lfloor [\![x_1]\!](\alpha_1)$

(b) $[\![x_1 \parallel x_2]\!] = [\![x_2 \parallel x_1]\!]$

(c) $[\![x_1 \parallel (x_2 \parallel x_3)]\!] = [\![(x_1 \parallel x_2) \parallel x_3]\!]$

Proof.

Property (a) follows by induction on $size(x_1) + size(x_2)$; note that, in general, $\alpha_1 \neq \alpha_2$.
For property (c), let $x_1, x_2, x_3 \in X$, $\alpha \in Alpha$.

$$
\begin{aligned}
&[\![x_1 \parallel (x_2 \parallel x_3)]\!](\alpha) = [\![x_1]\!](\alpha) \parallel [\![x_2 \parallel x_3]\!](\alpha) \\
&= [\![x_1]\!](\alpha) \lfloor [\![x_2 \parallel x_3]\!](\alpha) = [\![x_1]\!](\alpha) \lfloor ([\![x_2]\!](\alpha) \parallel [\![x_3]\!](\alpha)) && \text{[Property (a)]} \\
&= [\![x_1]\!](\alpha) \lfloor ([\![x_2]\!](\alpha) \lfloor [\![x_3]\!](\alpha)) && [\ \lfloor\ \text{is associative}] \\
&= ([\![x_1]\!](\alpha) \lfloor [\![x_2]\!](\alpha)) \lfloor [\![x_3]\!](\alpha) \\
&= ([\![x_1]\!](\alpha) \parallel [\![x_2]\!](\alpha)) \lfloor [\![x_3]\!](\alpha) = [\![x_1 \parallel x_2]\!](\alpha) \lfloor [\![x_3]\!](\alpha) && \text{[Property (a)]} \\
&= [\![x_1 \parallel x_2]\!](\alpha) \parallel [\![x_3]\!](\alpha) = [\![(x_1 \parallel x_2) \parallel x_3]\!](\alpha)
\end{aligned}
$$

□

Semantics of $\mathcal{L}_{SNP}$ programs

Definition (Initial continuation κ_0)

Let $\Psi_{\mathbf{K}} : NDs \to \mathbf{K} \to \mathbf{K}$ *be given by*

$$\Psi_{\mathbf{K}}(D)(\kappa)(\gamma) = to_{\Omega}(\gamma) \cdot (\text{ if } halt_{NS}(\gamma, D) \text{ then } \{\epsilon\}$$
$$\text{else } \bigoplus\{\varphi(\kappa)(\gamma') \mid (\varphi, \gamma') \in sched(\gamma, D)\} \oplus$$
$$\bigoplus\{\kappa(\gamma') \mid (d_0, \gamma') \in sched(\gamma, D)\})$$

For any $D \in NDs$*, we define* $\kappa_0 \in \mathbf{K}$ *by* $\kappa_0 = \text{fix}(\Psi_{\mathbf{K}}(D))$.

Remark (Time is implicit in our denotational model)

- *In SN P systems functioning is synchronized*
- *A global clock is assumed; in our model the value of the clock is given by the number of* Ω *observable steps in each* $\mathbf{Q}$ *execution trace*
- $to_{\Omega} : \Gamma \to \Omega$ *produces an observable* $\omega \in \Omega$ *from a configuration* $\gamma \in \Gamma$

Semantics of $\mathcal{L}_{SNP}$ programs - scheduler mapping

$$\|: (\textbf{Den} \times \textbf{Den}) \xrightarrow{1} \textbf{Den},\ d_0 \parallel d_0 = d_0,\ d_0 \parallel \varphi = \varphi,\ \varphi \parallel d_0 = \varphi,\ \varphi_1 \parallel \varphi_2 = \varphi_1 \parallel \varphi_2$$

$$\begin{aligned}
&sched : (\Gamma \times NDs) \to \mathcal{P}_{co}(\textbf{Den} \times \Gamma) \\
&sched(\gamma, D) = \quad \text{let } \{N_0, \ldots, N_m\} = id(\gamma) \\
&\qquad \text{in } \{(\|_{i=0}^{m} \phi_i, [\gamma \mid N_0 \mapsto \sigma_0 \mid \cdots \mid N_m \mapsto \sigma_m]) \\
&\qquad \mid (\phi_0, \sigma_0) \in schedN(N_0, \gamma(N_0), D), \ldots, \\
&\qquad (\phi_m, \sigma_m) \in schedN(N_m, \gamma(N_m), D)\}
\end{aligned}$$

$$\begin{aligned}
&schedN : (Nname \times \Sigma \times NDs) \to \mathcal{P}_{co}(\textbf{Den} \times \Sigma) \\
&schedN(N, (\xi, w), D) = \text{ if } halt_N(N, (\xi, w), D) \text{ then } \{(d_0, (\xi, w))\} \\
&\qquad \text{else let } r = rules(D, N) \\
&\qquad \text{in } \{([\![x]\!](N, \text{all}), (\xi, w \setminus w_r)) \\
&\qquad \mid (E/w_r \to x; t) \in r, w \in L(E), w_r \subseteq w, t = 0\} \cup \\
&\qquad \{(d_0, (\xi, w, t-1, w \setminus w_r, [\![x]\!](N, \text{all}))) \\
&\qquad \mid (E/w_r \to x; t) \in r, w \in L(E), w_r \subseteq w, t > 0\} \cup \\
&\qquad \{(d_0, (\xi, [\,])) \mid (w_r \to \lambda) \in r, w_r = w\};
\end{aligned}$$

$$\begin{aligned}
&schedN(N, (\xi, w, t, w_r, \varphi), D) = \\
&\quad \text{if } t = 0 \text{ then } \{(\varphi, (\xi, w_r))\} \text{ else } \{(d_0, (\xi, w, t-1, w_r, \varphi))\}
\end{aligned}$$

Semantics of $\mathcal{L}_{SNP}$ programs: fixed-point (compositional) semantics

Proposition (κ_0 is well-defined (Banach))

$\Psi_{\mathbf{K}}(D) \in \mathbf{K} \xrightarrow{\frac{1}{2}} \mathbf{K}$ *($\Psi_{\mathbf{K}}(D)$ is a contraction), for any $D \in NDs$.*

Proof.

It suffices to prove the following (for any $D \in NDs, \kappa_1, \kappa_2 \in \mathbf{K}, \gamma \in \Gamma$):

$$d(\Psi_{\mathbf{K}}(D)(\kappa_1)(\gamma), \Psi_{\mathbf{K}}(D)(\kappa_2)(\gamma)) \leq \frac{1}{2} \cdot d(\kappa_1, \kappa_2)$$

We have

$$d(\Psi_{\mathbf{K}}(D)(\kappa_1)(\gamma), \Psi_{\mathbf{K}}(D)(\kappa_2)(\gamma))$$

["$to_\Omega(\gamma)$" - step in def. $\Psi_{\mathbf{K}}$, metric on $\mathbf{P}$, $\oplus$ is nonexpansive]

$$\leq \frac{1}{2} \cdot max\{max\{d(\varphi(\kappa_1)(\gamma'), \varphi(\kappa_2)(\gamma')) \mid (\varphi, \gamma') \in sched(\gamma, D)\}, max\{d(\kappa_1(\gamma'), \kappa_2(\gamma')) \mid (d_0, \gamma') \in sched(\gamma, D)\}\}$$

[$\varphi \in \mathbf{D}$, φ is nonexpansive]

$$\leq \frac{1}{2} \cdot d(\kappa_1, \kappa_2)$$

Semantics of $\mathcal{L}_{SNP}$ programs

Definition (Semantics of $\mathcal{L}_{SNP}$ programs)

We define $\mathcal{D}[\![\cdot]\!] : \mathcal{L}_{SNP} \to \mathbf{P}$ *for any* $\rho = (D, x) \in \mathcal{L}_{SNP}$ *by*

$$\mathcal{D}[\![\rho]\!] = \mathcal{D}[\![D, x]\!] = [\![x]\!](\alpha_0)(\kappa_0)(\gamma_0),$$

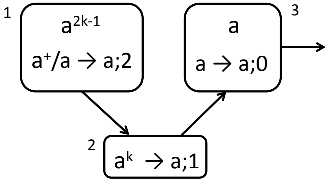
where $\kappa_0 = fix(\Psi_{\mathbf{K}}(D))$, $\gamma_o = init_{\Gamma}(D)$ *and* $\alpha_0 = (N_0, \mathrm{all})$.

- Haskell implementation provided in two variants:
  - `Lsnp.hs` - accurate implementation of $\mathcal{D}[\![\cdot]\!]$; can only run simple $\mathcal{L}_{SNP}$ programs like ρ_1 or ρ_1' (Π_1)
  - `Lsnp-fin.hs` - stops execution (prunes execution traces) after n steps; can run arbitrary $\mathcal{L}_{SNP}$ programs, including nonterminating and nondeterministic programs

Example: let $\rho_1 = (D_1, x_1)$, $\rho'_1 = (D'_1, x'_1)$ be as before

- ρ_1 implements deterministic SN P system Π_1

- $x_1 = \text{send}(\langle a^{2k-1}\rangle, \{N_1\}) \parallel \text{send}(a, \{N_3\})$
- $D_1 = $ neuron N_0 $\{ r_\epsilon \mid \{N_1, N_2, N_3\} \}$,
  neuron N_1 $\{ a^+/[a] \to a; 2 \mid \{N_2\} \}$,
  neuron N_2 $\{ [a^k] \to a; 1 \mid \{N_3\} \}$,
  neuron N_3 $\{ [a] \to a; 0 \mid \{N_0\} \}$
- $\omega_1 = \{(N_0, [\,]), (N_1, [a, a, a]), (N_2, [\,]), (N_3, [a])\}$
  $\omega_2 = \{(N_0, [a]), (N_1, [a, a, a]), (N_2, [\,]), (N_3, [\,])\}$
  $\omega_3 = \{(N_0, [a]), (N_1, [a, a, a]), (N_2, [\,]), (N_3, [\,])\}$
  $\omega_4 = \{(N_0, [a]), (N_1, [a, a]), (N_2, [a]), (N_3, [\,])\}$
  $\omega_5 = \{(N_0, [a]), (N_1, [a, a]), (N_2, [a]), (N_3, [\,])\}$
  $\omega_6 = \{(N_0, [a]), (N_1, [a, a]), (N_2, [a]), (N_3, [\,])\}$
  $\omega_7 = \{(N_0, [a]), (N_1, [a]), (N_2, [a, a]), (N_3, [\,])\}$
  $\omega_8 = \{(N_0, [a]), (N_1, [a]), (N_2, [a, a]), (N_3, [\,])\}$
  $\omega_9 = \{(N_0, [a]), (N_1, [a]), (N_2, [\,]), (N_3, [a])\}$
  $\omega_{10} = \{(N_0, [a, a]), (N_1, [\,]), (N_2, [a]), (N_3, [\,])\}$



[Ionescu, Păun and Yokomori – 2006]
SN P system Π_1

$$\mathcal{D}[\![\rho_1]\!] = [\![x_1]\!](\alpha_0)(\kappa_0)(\gamma_0) = \{\omega_1\omega_2\omega_3\omega_4\omega_5\omega_6\omega_7\omega_8\omega_9\omega_{10}\} = \mathcal{D}[\![\rho'_1]\!]$$

- Our Haskell interpreter `Lsnp-fin.hs` also runs nondeterministic SN P system Π_3 [Ionescu, Păun and Yokomori - 2006]; Π_3 computes all natural numbers (> 1)

Declarative protyping and denotational specification

```
type D = K -> K
data Den = D0 | D D

type K = Gamma -> P
type Gamma = Bag Sigma
data Sigma =
             Open Xi W
           | Closed Xi W Time W D
type P = [Q]
data Q = Q Omega Q | Epsilon

sem (Aspike a) alpha =
  \k gamma -> k (send a alpha gamma)
sem (Send y xi) alpha =
  sem y (xi `dintersect` alpha)
sem (Par x1 x2) alpha =
  (sem x1 alpha) `par` (sem x2 alpha)

k0 = fix (psiK nDs)
fix ::  (a -> a) -> a
fix f = f (fix f)
```

$$\mathbf{D} \cong \mathbf{K} \xrightarrow{1} \mathbf{K}$$
$$\mathbf{Den} = \{d_0\} + \mathbf{D}$$

$$\mathbf{K} = \Gamma \xrightarrow{1} \mathbf{P}$$
$$\Gamma = \{\!|\Sigma|\!\}$$
$$\Sigma = \mathbf{Open} + \mathbf{Closed}$$
$$\mathbf{Open} = \Xi \times W$$
$$\mathbf{Closed} = \Xi \times W \times \mathbb{N} \times W \times \tfrac{1}{2} \cdot \mathbf{D}$$
$$\mathbf{P} = \mathcal{P}_{nco}(\mathbf{Q})$$
$$\mathbf{Q} \cong \{\epsilon\} + (\Omega \times \tfrac{1}{2}\mathbf{Q})$$

$$[\![a]\!](\alpha) = \lambda\kappa\,.\,\lambda\gamma\,.\,\kappa(send(a,\alpha,\gamma))$$
$$[\![\,\text{send}(y,\xi)\,]\!](\alpha) = [\![y]\!](\xi \Cap \alpha)$$
$$[\![x_1 \parallel x_2]\!](\alpha) = [\![x_1]\!](\alpha) \parallel [\![x_2]\!](\alpha)$$

$$\kappa_0 = \text{fix}(\Psi_{\mathbf{K}}(D))$$

Conclusion and future research

- We offer a denotational semantics $[\![\cdot]\!]$ for an experimental concurrent language $\mathcal{L}_{SNP}$ inspired by the SN P systems
  - $[\![\cdot]\!]$ is designed with metric domains and continuations
  - Accurately describes the behavior of SN P systems
    - Including time delays and synchronized functioning
  - Offers support for reasoning about the behavior of $\mathcal{L}_{SNP}$
  - Haskell implementation available from

    `http://ftp.utcluj.ro/pub/users/gc/eneia/cmc19`

- In the future we could
  - Study the abstractness of denotational vs operational semantics of SN P systems [Ciobanu & Todoran - 2018]
  - Develop language support and formal verification tools for SN P systems extended with: dynamical structure, quantitative aspects (stochastic/fuzzy), compositionality

S. Abramsky and A. Jung,
Domain Theory,
Handbook of Logic in Computer Science, Clarendon Press, Oxford, pp. 1–170, 1994.

P. America, J.J.M.M. Rutten,
Solving reflexive domain equations in a category of complete metric spaces,
J. of Comput. System Sci. **39**, 343–375, 1989.

A.W. Appel,
Compiling with Continuations,
Cambridge University Press, 2007.

J.W. de Bakker, E.P. de Vink,
Control Flow Semantics,
MIT Press, 1996.

H. Chen, M. Ionescu, T.O. Isidorj, A. Păun, Gh. Păun, M.J. Pérez-Jiménez,
Spiking neural P systems with extended rules: universality and languages,
Natural Computing **7**, 453–470, 147–166, 2008.

G. Ciobanu, Gh. Păun, M.J. Pérez-Jiménez.
Applications of Membrane Computing,
Natural Computing Series, Springer, 2006.

G. Ciobanu, E.N. Todoran.
Continuation semantics for asynchronous concurrency,
Fundamenta Informaticae **131**, 373–388, 2014.

G. Ciobanu, E.N. Todoran,
Denotational semantics of membrane systems by using complete metric spaces,
Theoretical Computer Science **701**, 85–108, 2017.

G. Ciobanu, E.N. Todoran,
Abstractness of Continuation Semantics for Asynchronous Concurrency,
Federated Logic Conference, Workshop Domains, Oxford, UK, 2018.

M. Ionescu, G. Păun, T. Yokomori,
Spiking neural P systems,
Fundamenta Informaticae **71**, 279–308, 2006.

Gh. Păun,
Membrane Computing. An Introduction.
Springer, 2002.

Gh. Păun,
Spiking Neural P Systems with Astrocyte-Like Control
Journal of Universal Computer Science, vol. 13(11), pp. 1707–1721, 2007.

Gh. Păun, G. Rozenberg, A. Salomaa (Eds.),
Handbook of Membrane Computing,
Oxford University Press, 2010.

S. Peyton Jones, J. Hughes (Eds.),
Report on the Programming Language Haskell 98: A Non-Strict Purely Functional Language, 1999, `http://www.haskell.org`

D.S. Scott,
Data Types as Latices,
SIAM J. Comput., vol. 5, pp. 522-587, 1976.

D.S. Scott,
Domains for Denotational Semantics,
Proc. ICALP'82, LNCS, vol. 140, pp. 577–613, 1982.

E.N. Todoran,
Metric semantics for synchronous and asynchronous communication: a continuation-based approach,
Electronic Notes in Theoretical Computer Science **28**, 101–127, 2000.

WWW:
Haskell implementation of the denotational semantics of $\mathcal{L}_{SNP}$, 2018,
`http://ftp.utcluj.ro/pub/users/gc/eneia/cmc19`