Generating String Languages using Spiking Neural P Systems with Structural Plasticity

MS Thesis Proposal by

Ren Tristan A. de la Cruz

2017 July 20

Membrane Computing &

P Systems

Membrane Computing [1]

Gheorghe Păun's
paper titled

"***Computing with Membranes***",

published in the

<u>Journal of Computer and System Sciences</u>

started the field known as

Membrane Computing.

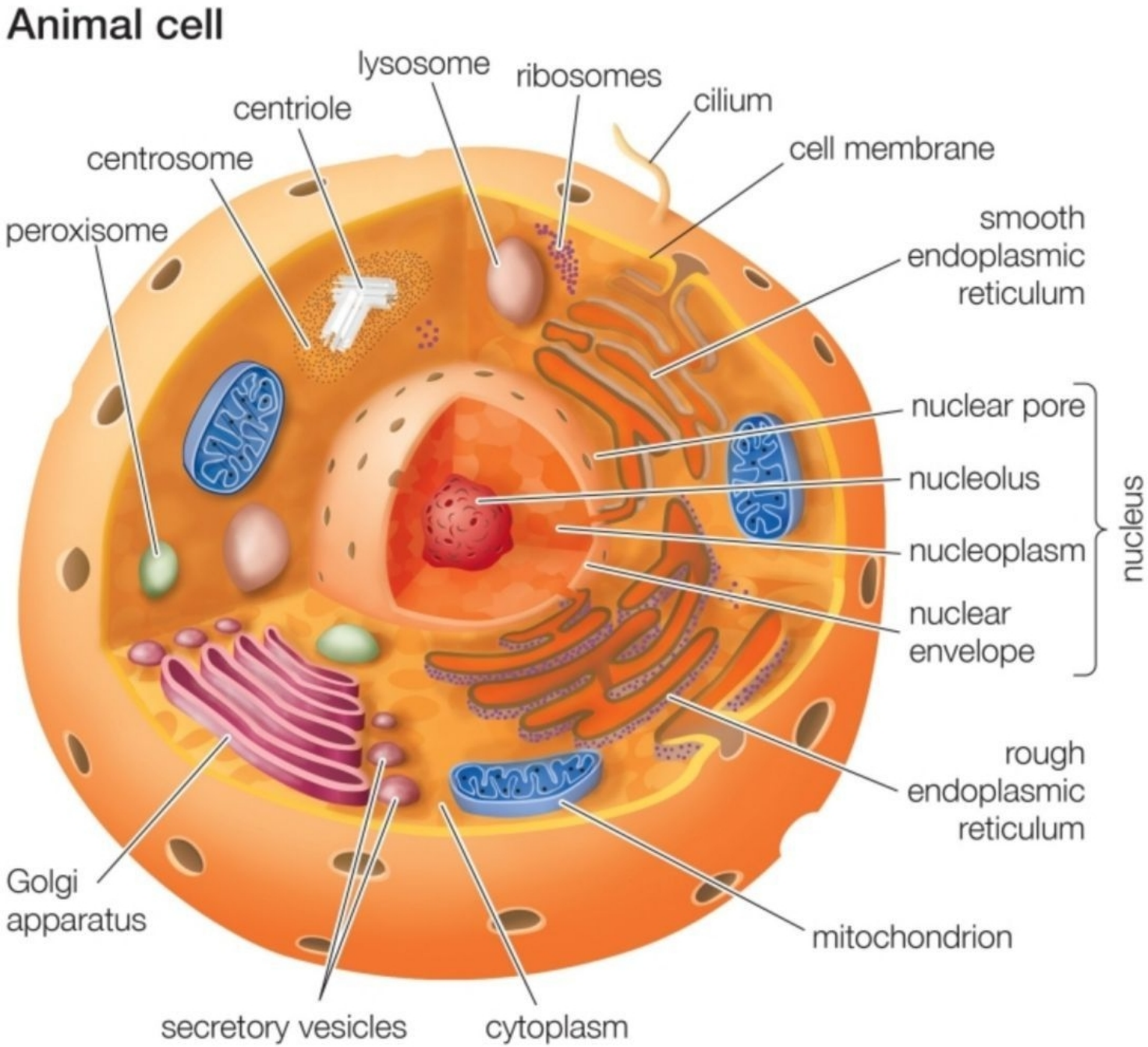
Membrane Computing [1]

Membrane computing is branch of computer science that studies computing models known as *P systems*.

***P** systems*, named after Gheorghe **P**ăun, are computing models inspired by natural processes or more specifically biological processes.

The term '*membrane*' in membrane computing stands for the <u>membrane structure</u> in used by the P systems which is inspired by biological membranes.

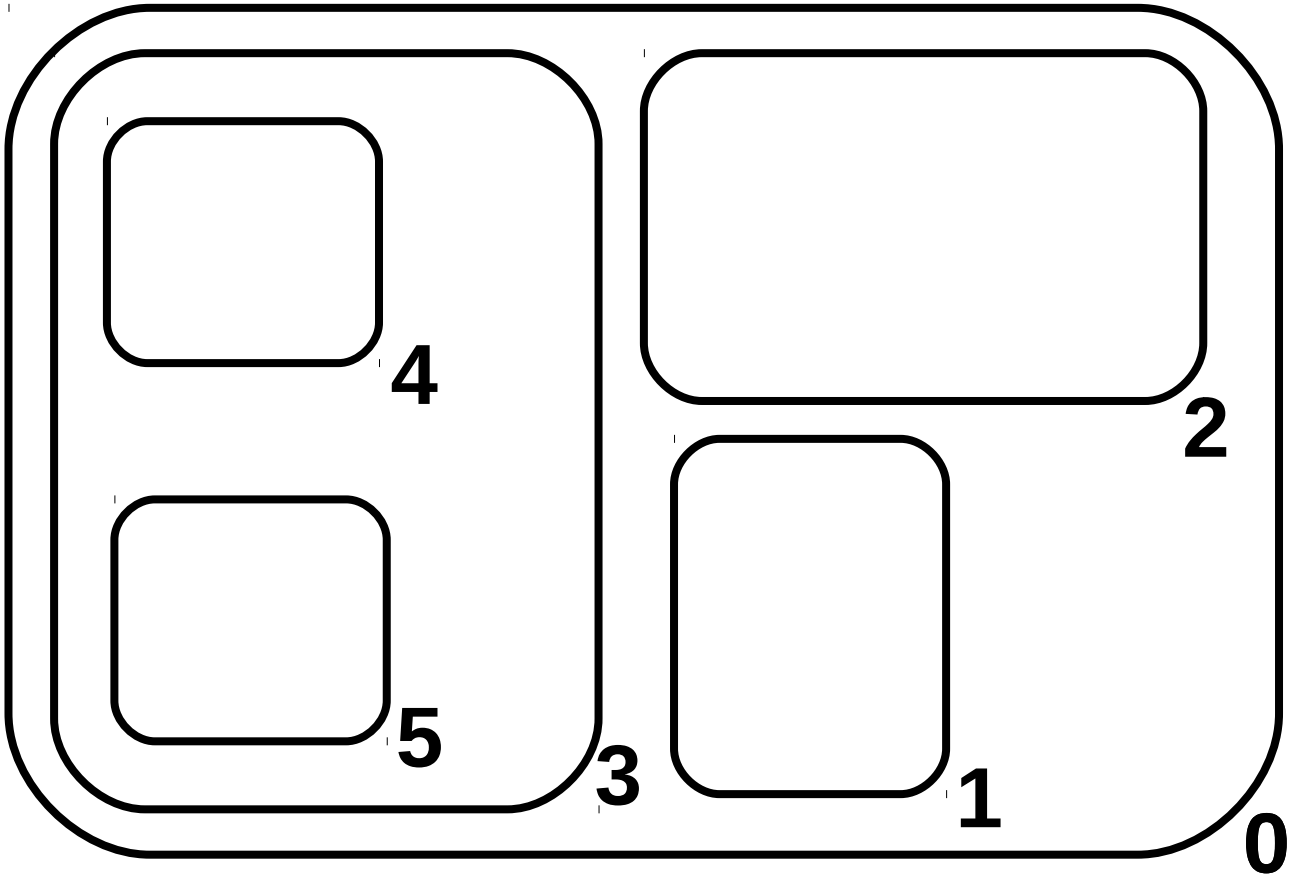
Membrane Computing [1]

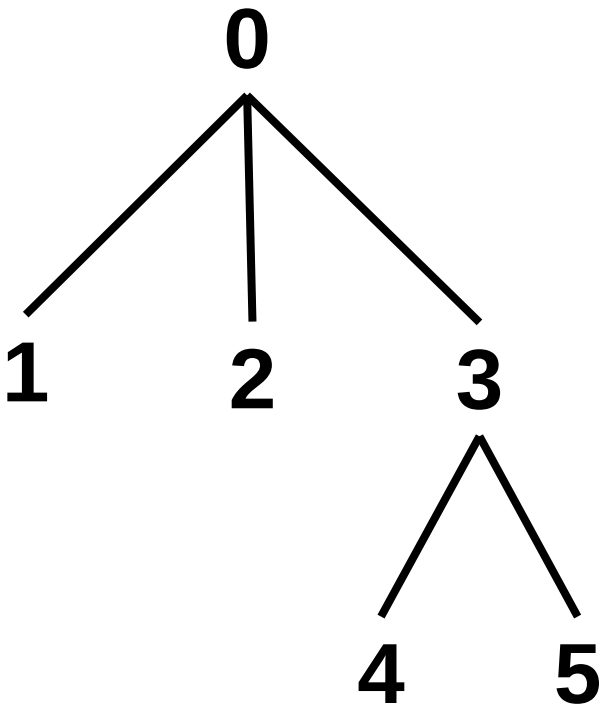


[3]

Membrane Computing [1]

Membrane Structure of a cell-like P System:





Membrane Computing [1]

Chemical reactions that occur inside the cell also serve as inspiration for the mechanisms in the P systems

$2(H_2O_2) \rightarrow 2(H_2O) + O_2$ (chemical reaction)

P – **H_20_2** – Hydrogen Peroxide

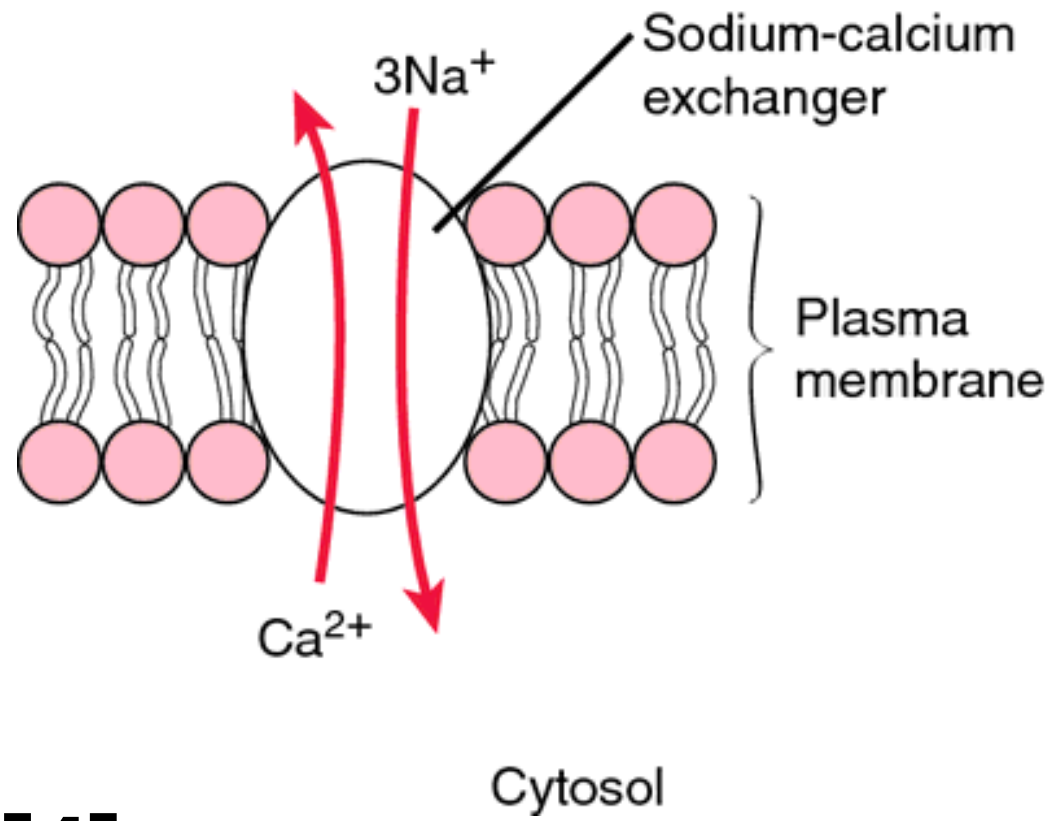
W – **H_2O** – Water

X – **O_2** – Oxygen Gas

$P^2 \rightarrow W^2X$ (**multiset rewriting rule**)

Membrane Computing [1]

Ion transportation between membrane regions is also an inspiration for a type of P systems rules.



N – Na^+

(Sodium Ion)

C – Ca^{2+}

(Calcium Ion)

Communication Rule:

(3N, in; C, out)

[4]

Membrane Computing [1]

[1] Gh. Păun: "*Computing with Membranes*". Journal of Computer and System Sciences, 61, 1 (**2000**), 108–143 (and Turku Center for Computer Science-TUCS Report 208, November **1998**, www.tucs.fi).

[2] https://alchetron.com/Gheorghe-P%C4%83un-2643317-W

[3] Getty Images, Encyclopaedia Britannica/UIG,http://www.gettyimages.com/detail/illustration/typical-animal-cell-and-plant-cell-stock-graphic/141482965

[4] http://medical-dictionary.thefreedictionary.com/antiport

Spiking Neural P Systems

(SNP Systems)

Spiking Neural P System [1]

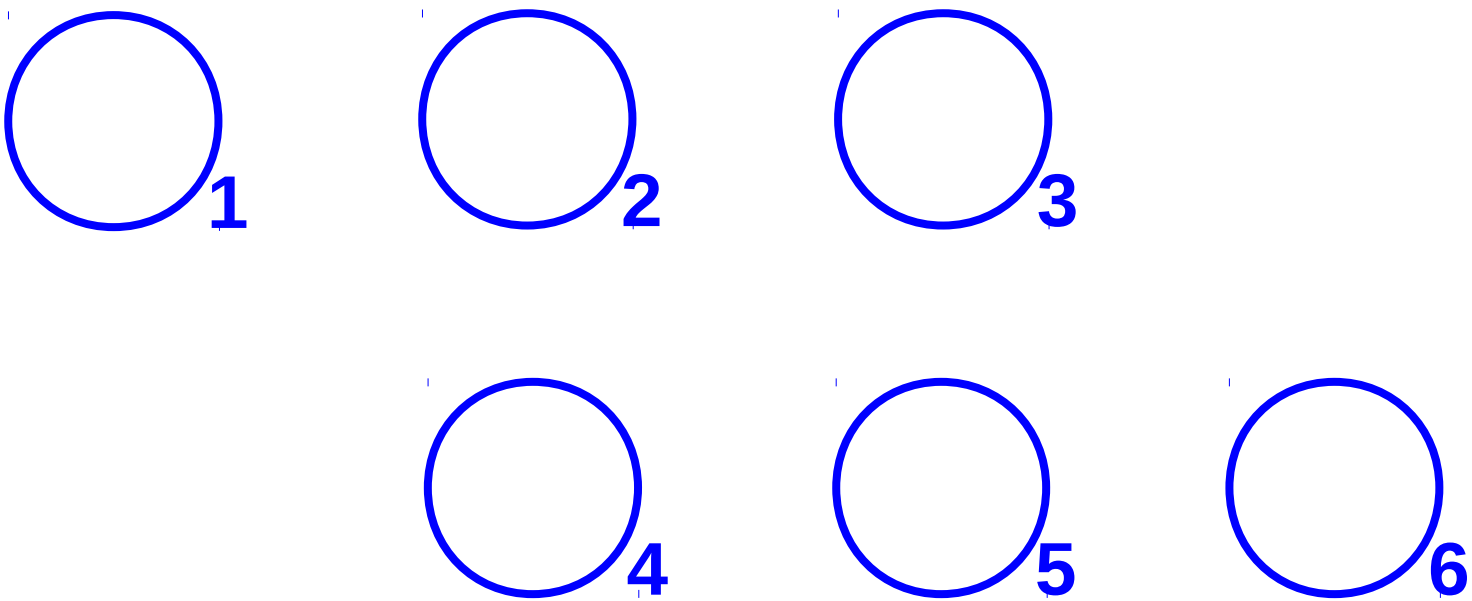
A spiking neural P (SNP) system **Π** of degree **m** is the construct:

$$\mathbf{\Pi = (O, \sigma_1, \ldots, \sigma_m, \mathit{syn}, \mathit{out})}$$

Spiking Neural P System [1]

SNP System Π = (O, σ_1, ... ,σ_m, *syn*, *out*)

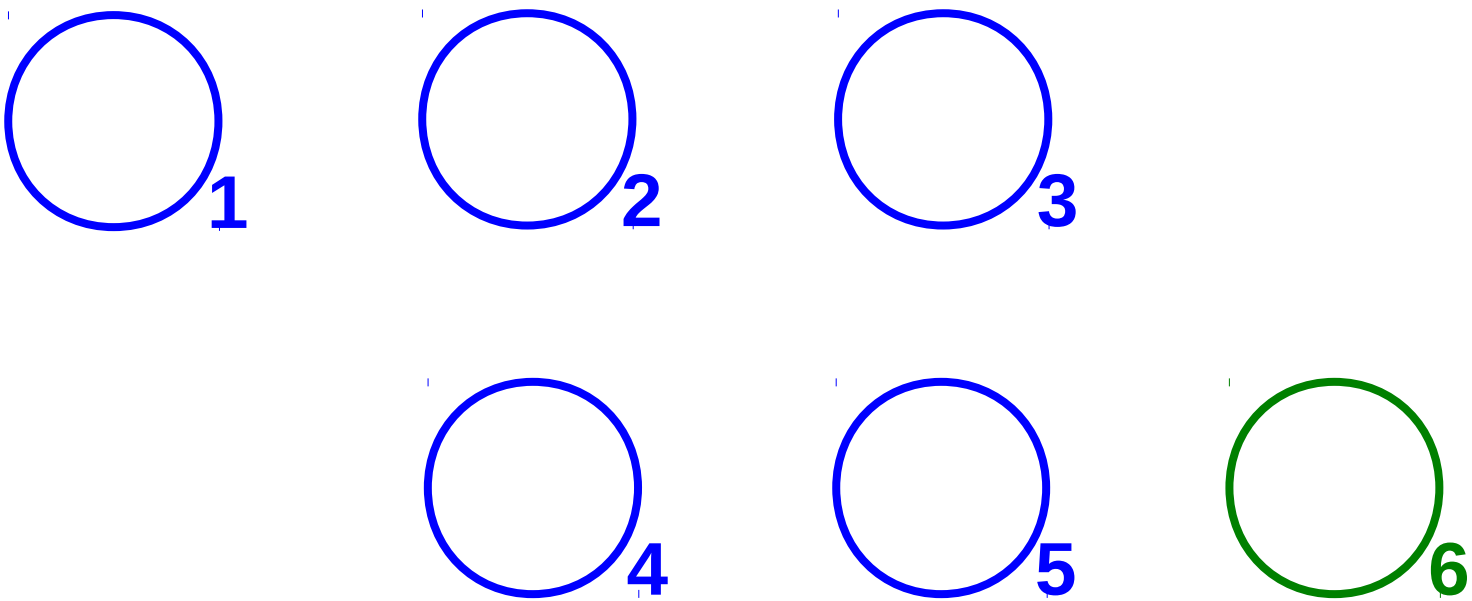
σ_1, ... ,σ_m are the **m** neurons of the system.



Spiking Neural P System [1]

SNP System Π = (O, σ_1, ... ,σ_m, *syn*, *out*)

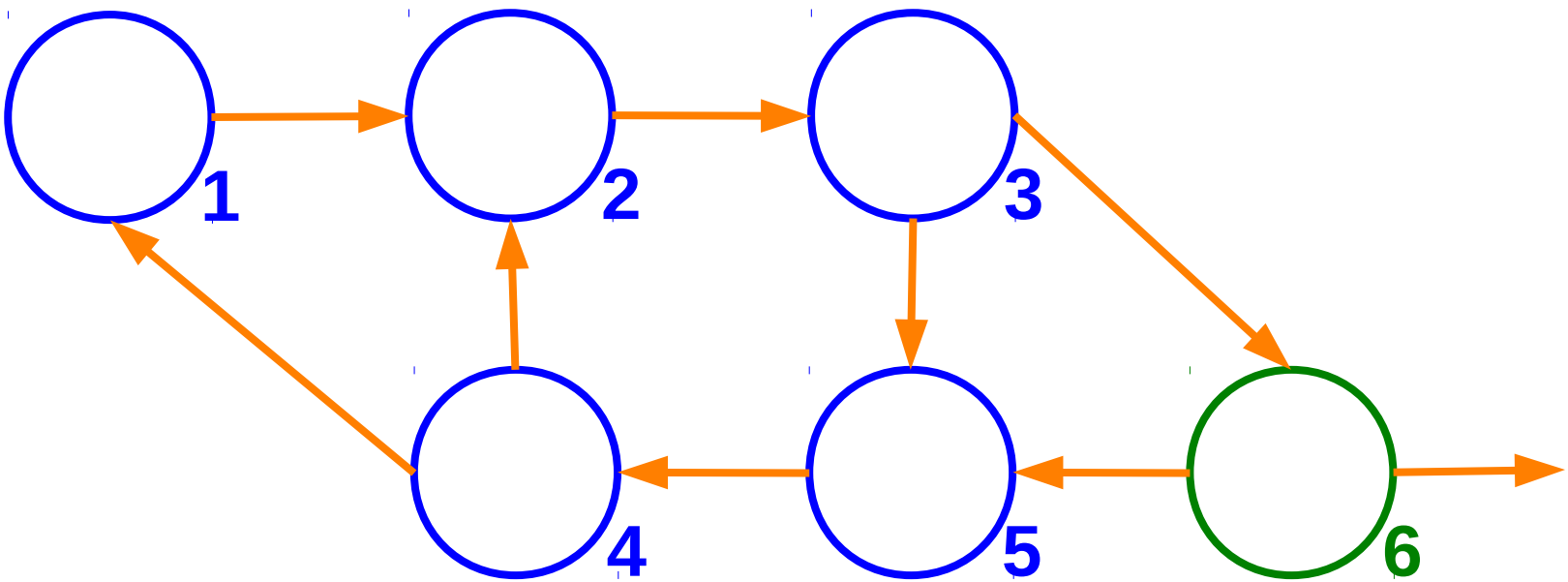
***out* ∈ {1,...,m}** is the label of the designated output neuron.



Spiking Neural P System [1]

SNP System $\Pi = (O, \sigma_1, \ldots, \sigma_m, syn, out)$

$syn \subseteq \{1,\ldots,m\} \times \{1,\ldots,m\}$ is the set of synapses between the neurons.



Spiking Neural P System [1]

SNP System $\Pi = (O, \sigma_1, \ldots, \sigma_m, syn, out)$

$O = \{a\}$ is the object alphabet of the system. The system only uses a single type (**a**) of object known as *spike*.

Spiking Neural P System [1]

SNP System $\Pi = (O, \sigma_1, \ldots, \sigma_m, syn, out)$

Each neuron (σ_i) has the following form:

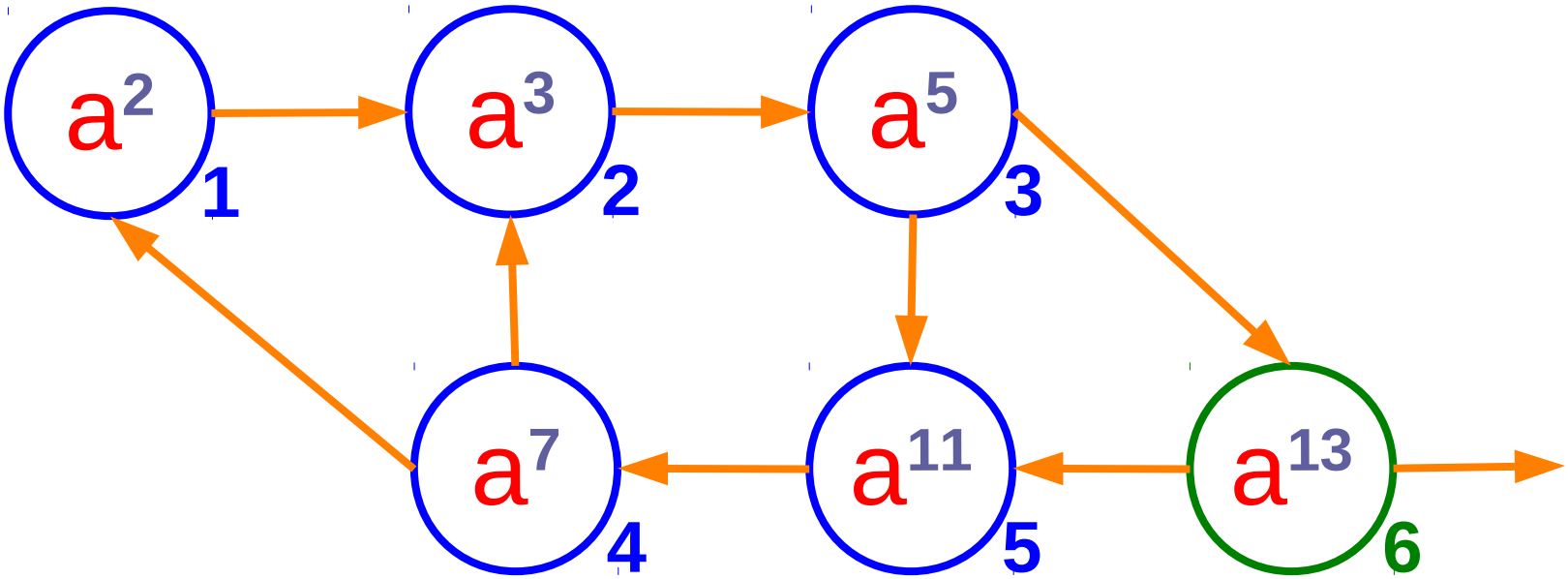
$\sigma_i = (n_i, R_i)$

$n_i \geq 0$ is the initial number of spikes in σ_i.

Spiking Neural P System [1]

SNP System $\Pi = (O, \sigma_1, \ldots, \sigma_m, syn, out)$

Using the alphabet **O**, the number of spikes n_i in σ_i is represented by the string a^{n_i}.



Spiking Neural P System [1]

SNP System $\Pi = (O, \sigma_1, \ldots, \sigma_m, syn, out)$

Each neuron (σ_i) has the following form:

$\sigma_i = (n_i, R_i)$

R_i is the finite set of rules of σ_i.

Spiking Neural P System [1]

A rule $\mathbf{r_j} \in \mathbf{R_i}$ is either one of two possible forms.

(1) <u>Spiking Rule</u>: $\mathbf{E / a^c \rightarrow a : d}$

(2) <u>Forgetting Rule</u>: $\mathbf{a^s \rightarrow \lambda}$

Spiking Neural P System [1]

<u>Spiking Rule</u>: $\mathbf{E} / \mathbf{a}^{c} \rightarrow \mathbf{a} : \mathbf{d}$

E is a regular expression over **O**. The rule can only activate when the number of spikes ($\mathbf{a}^{n}$) in the neuron is covered by the regular expression **E**. $\mathbf{a}^{n} \in \mathbf{L}(\mathbf{E})$.

$\mathbf{a}^{n} = \mathbf{a}^{7}$, $\mathbf{E} = \mathbf{a}^{2}(\mathbf{a}^{5})^{*}$ - $\mathbf{a}^{n} \in \mathbf{L}(\mathbf{E})$ - OK

$\mathbf{a}^{n} = \mathbf{a}^{9}$, $\mathbf{E} = (\mathbf{a}^{2})^{*}$ - $\mathbf{a}^{n} \notin \mathbf{L}(\mathbf{E})$ - Not OK

Spiking Neural P System [1]

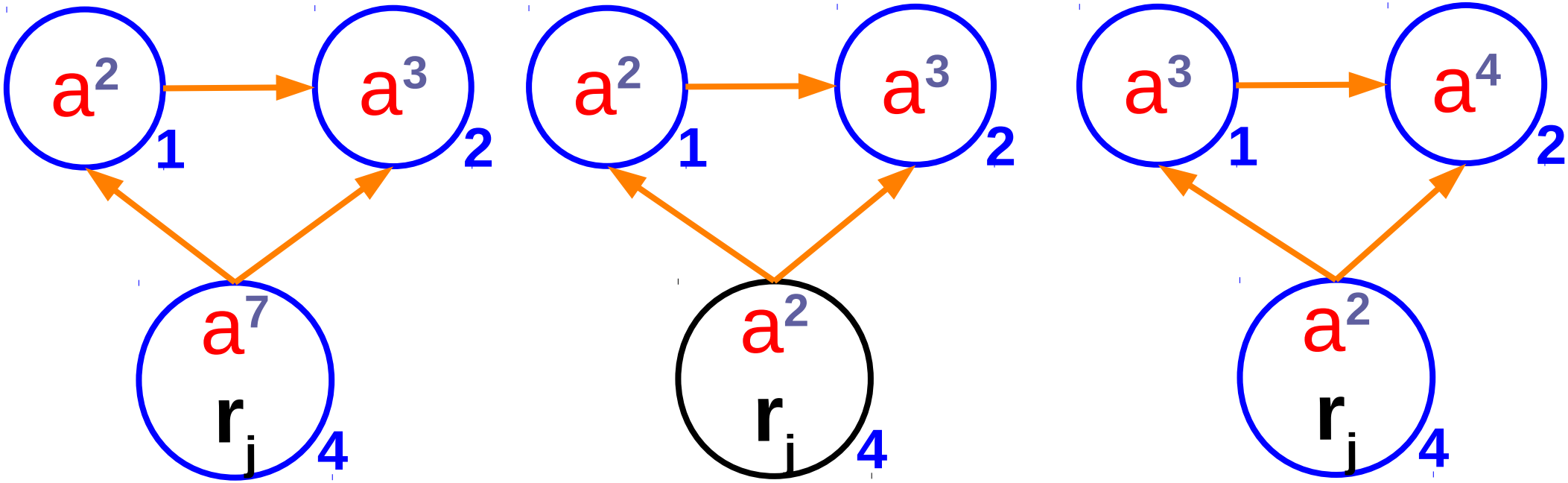
<u>Spiking Rule</u>: $\mathbf{E} / \mathbf{a}^{c} \rightarrow \mathbf{a} : \mathbf{d}$

$\mathbf{a}^{c}$ represents the number ($\mathbf{c} \geq \mathbf{1}$) of spike that will be consumed when the rule is activated. The rule can only be activated when the number ($\mathbf{n}$) of spikes in the neuron is at least $\mathbf{c}$ ($\mathbf{n} \geq \mathbf{c}$).

$\mathbf{d} \geq \mathbf{0}$ is delay between rule activation and sending of spikes.

Spiking Neural P System [1]

<u>Spiking Rule</u>: $\mathbf{r_j}$: $\mathbf{a^2(a^5)^* / a^5 \rightarrow a : 2}$



Time: t
$\mathbf{r_j}$ activates

Time: t
$\boldsymbol{\sigma_4}$ closes & consumes 5 spikes

Time: t+2
$\boldsymbol{\sigma_4}$ opens & sends spikes

$\boldsymbol{\sigma_4}$ is said to be 'closed' during time **t** and **t+1** and will not receive any spikes.

Spiking Neural P System [1]

<u>Forgetting Rule</u>: $\mathbf{a^s} \rightarrow \boldsymbol{\lambda}$

$\mathbf{a^s}$ represents the number ($\mathbf{s \geq 1}$) of spike that will be consumed when the rule is activated. The rule can only be activated when the number ($\mathbf{n}$) of spikes in the neuron is exactly $\mathbf{s}$ ($\mathbf{n = s}$).

When a forgetting rule is activated, all ($\mathbf{n}$) spikes are immediately consumed and no spikes are sent out.

Spiking Neural P System [1]

It's possible for a neuron to have multiple spiking rules,

$\mathbf{r_1 : E_1 / a^{c1} \rightarrow a: d_1}$

$\mathbf{r_2 : E_2 / a^{c2} \rightarrow a: d_2}$,

that can be activated at the same time. It can occur when $L(\mathbf{E_1}) \cap L(\mathbf{E_2}) \neq \emptyset$.

In case multiple spiking rules can be activated, the neuron *non-deterministically* select a rule to activate.

Spiking Neural P System [1]

Activation conditions of spiking rules and forgetting rules do not intersect. i.e.

$\mathbf{r_i : E_i / a^{ci} \rightarrow a: d_i}$

$\mathbf{r_j : a^s \rightarrow \lambda,}$

For any spiking rule $\mathbf{r_i}$ and any forgetting rule $\mathbf{r_j}$, $\mathbf{a^s} \notin L(\mathbf{E_i})$.

Spiking Neural P System [1]

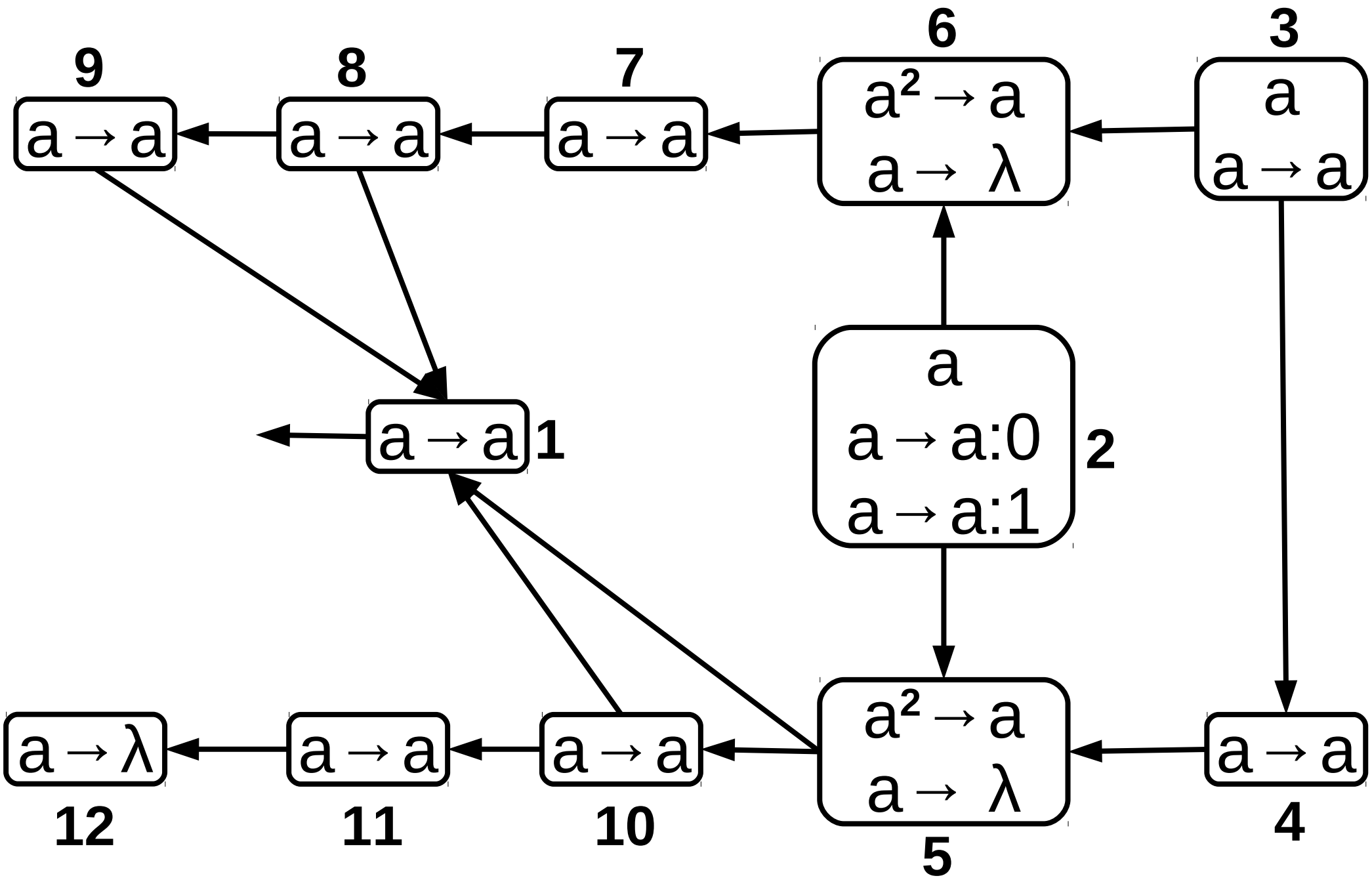
The following conventions are used for writing spiking rules:

If **E = a^c**, rule **r : "E / a^c → a: d"** can be written as **r : "a^c → a: d"**.

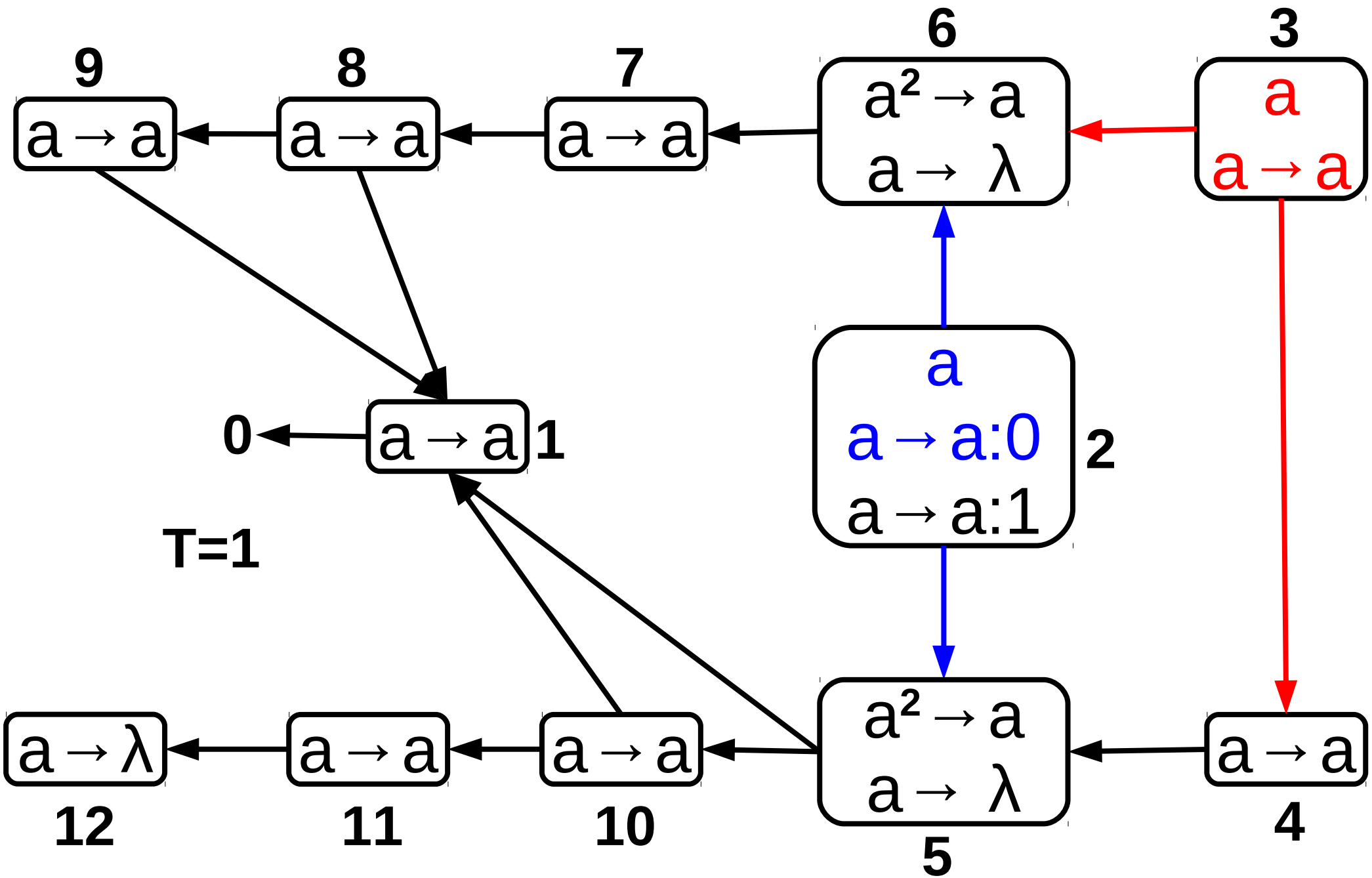
If **d = 0**, rule **r : "E / a^c → a: d"** can be written as **r : "E / a^c → a"**.

If **E = a^c** & **d = 0**, rule **r : "E / a^c → a: d"** can be written as **r : "a^c → a"**.

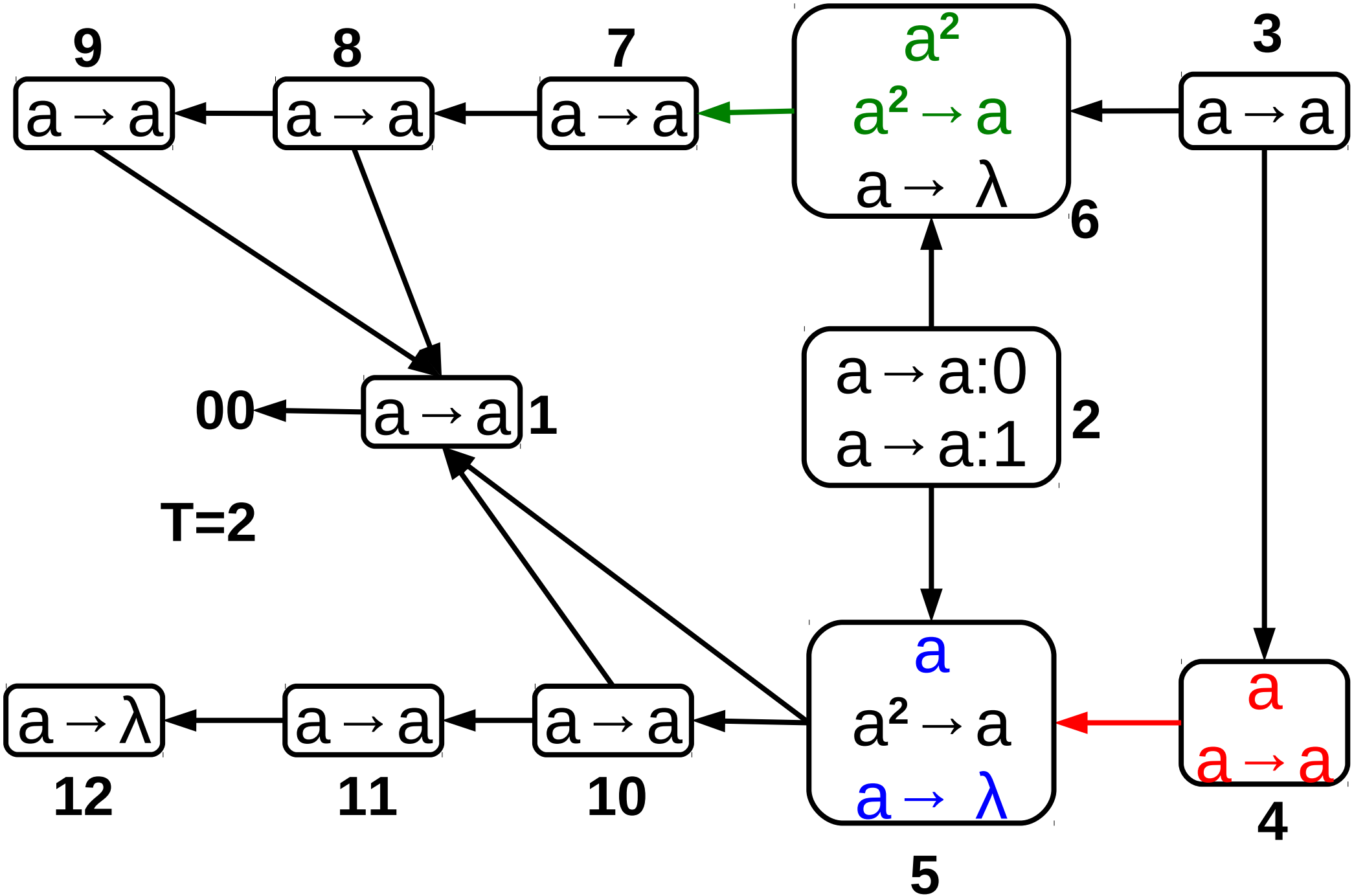
Spiking Neural P System [1]



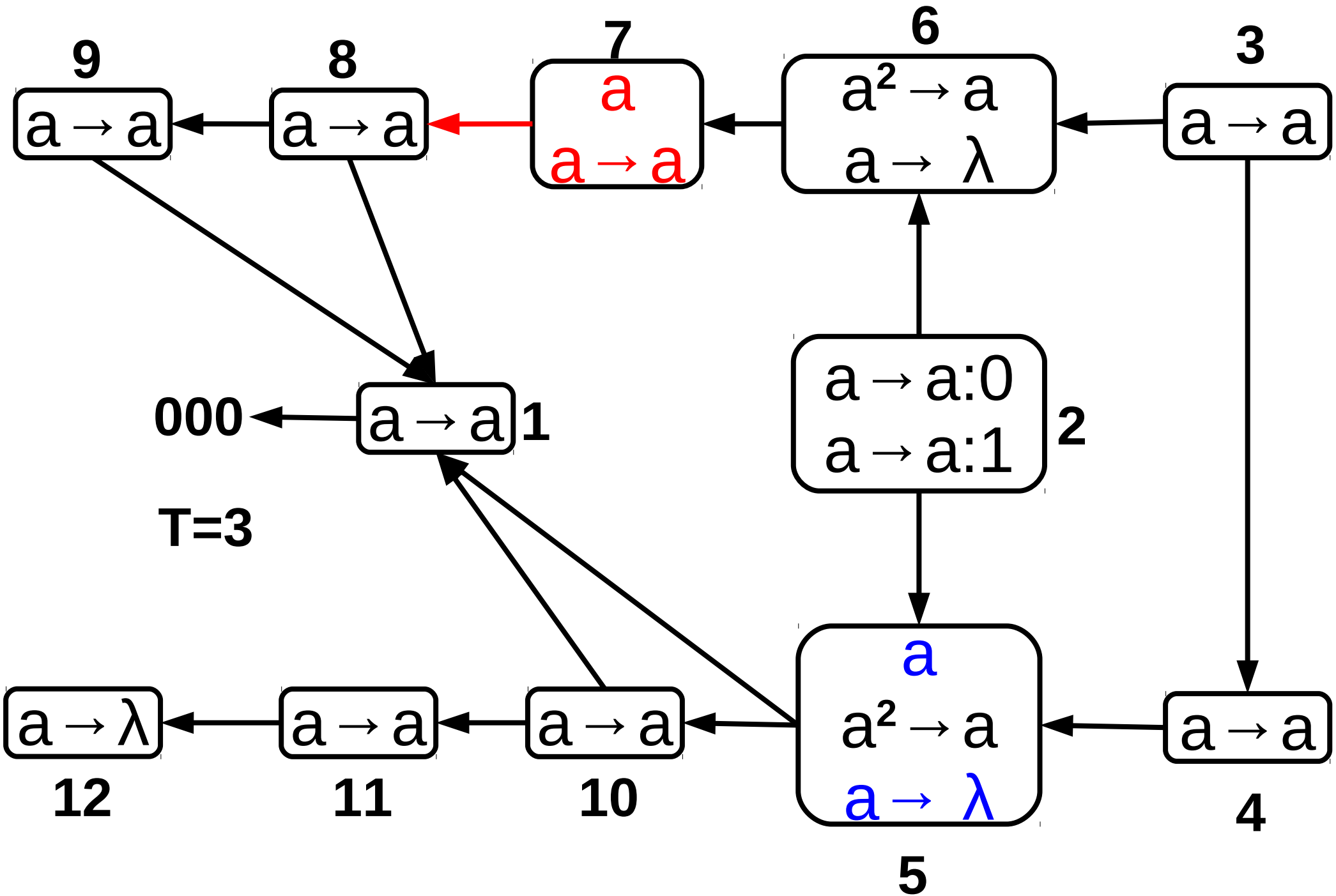
Spiking Neural P System [1]



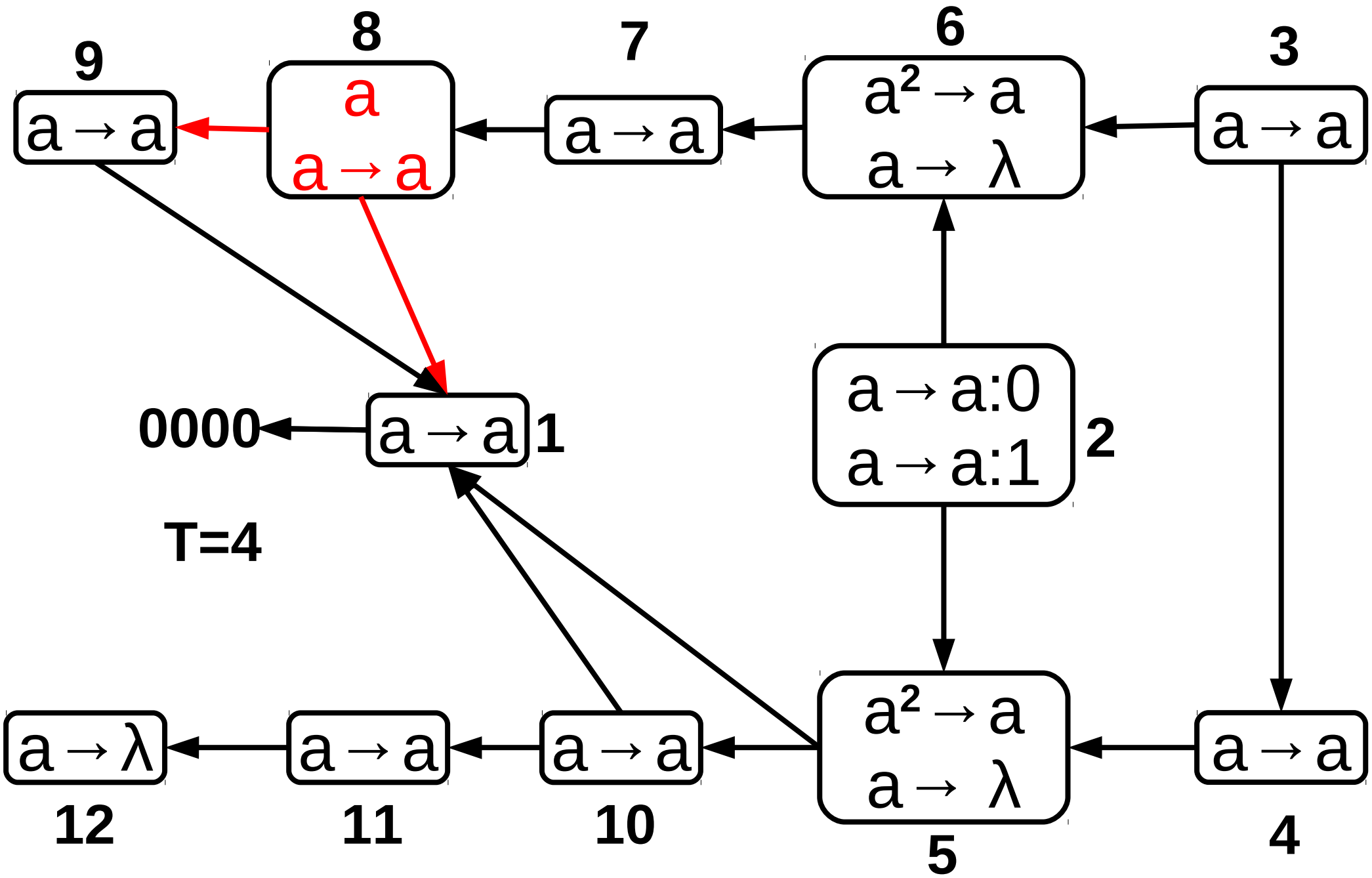
Spiking Neural P System [1]



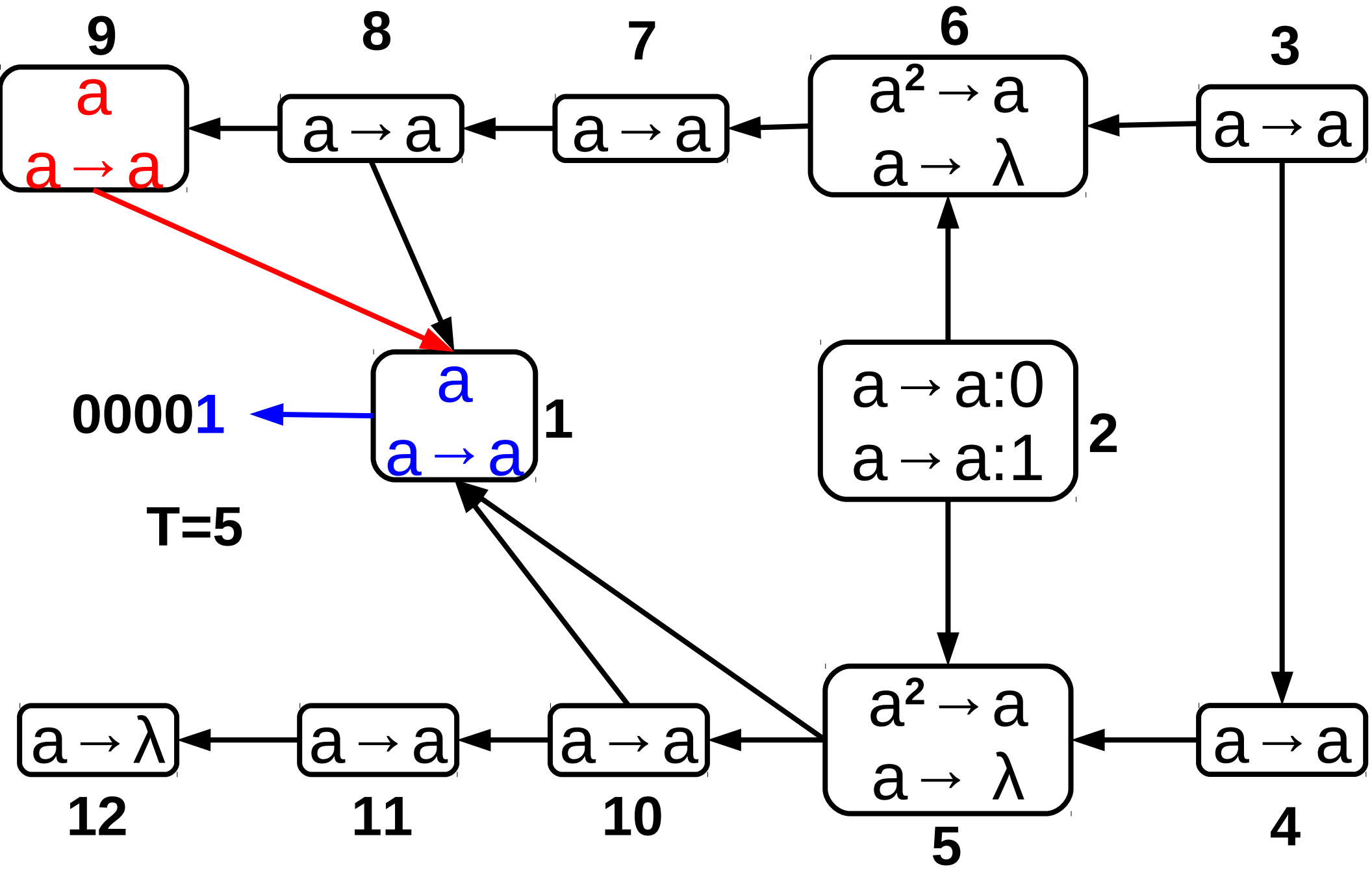
Spiking Neural P System [1]



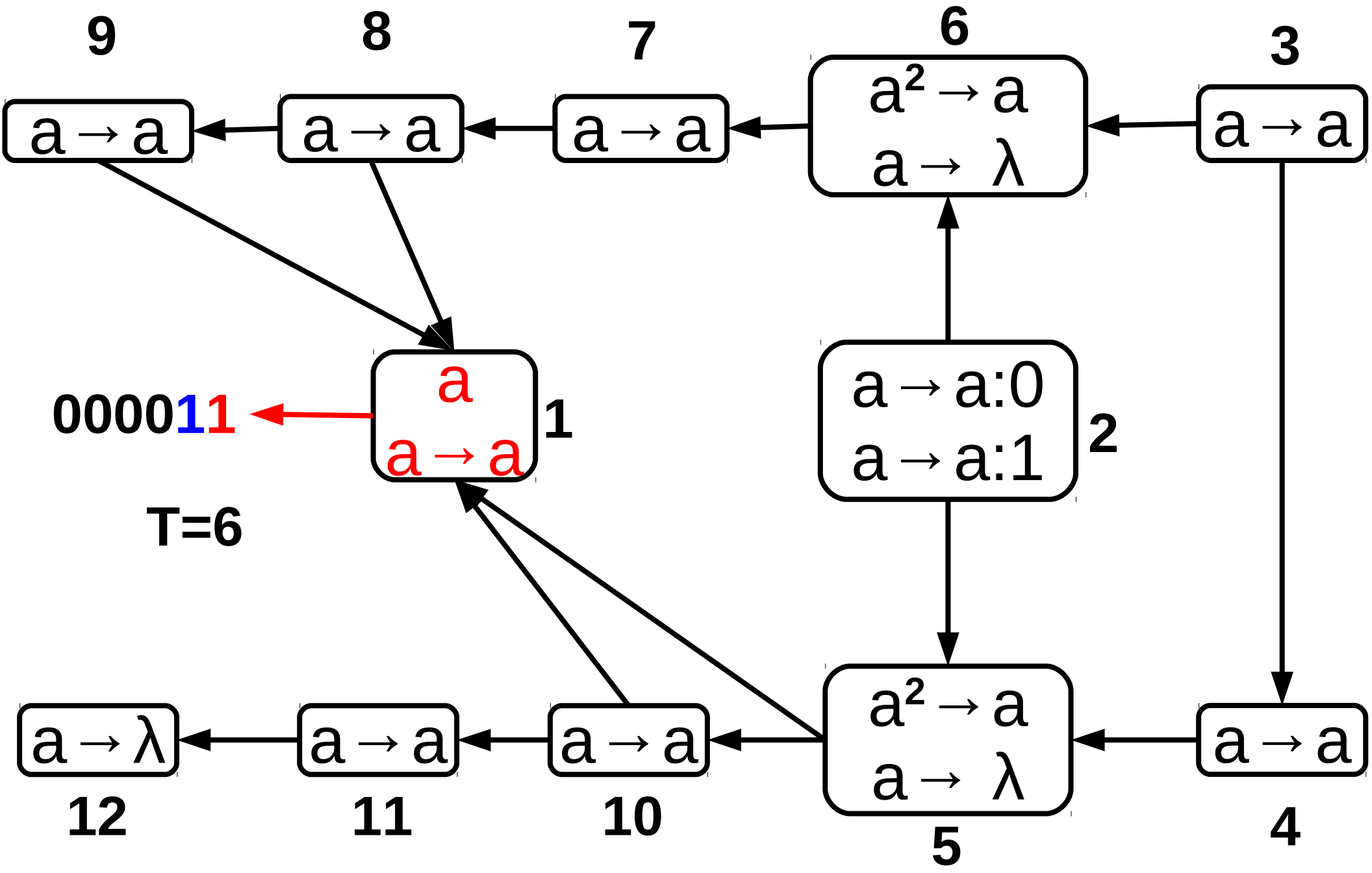
Spiking Neural P System [1]



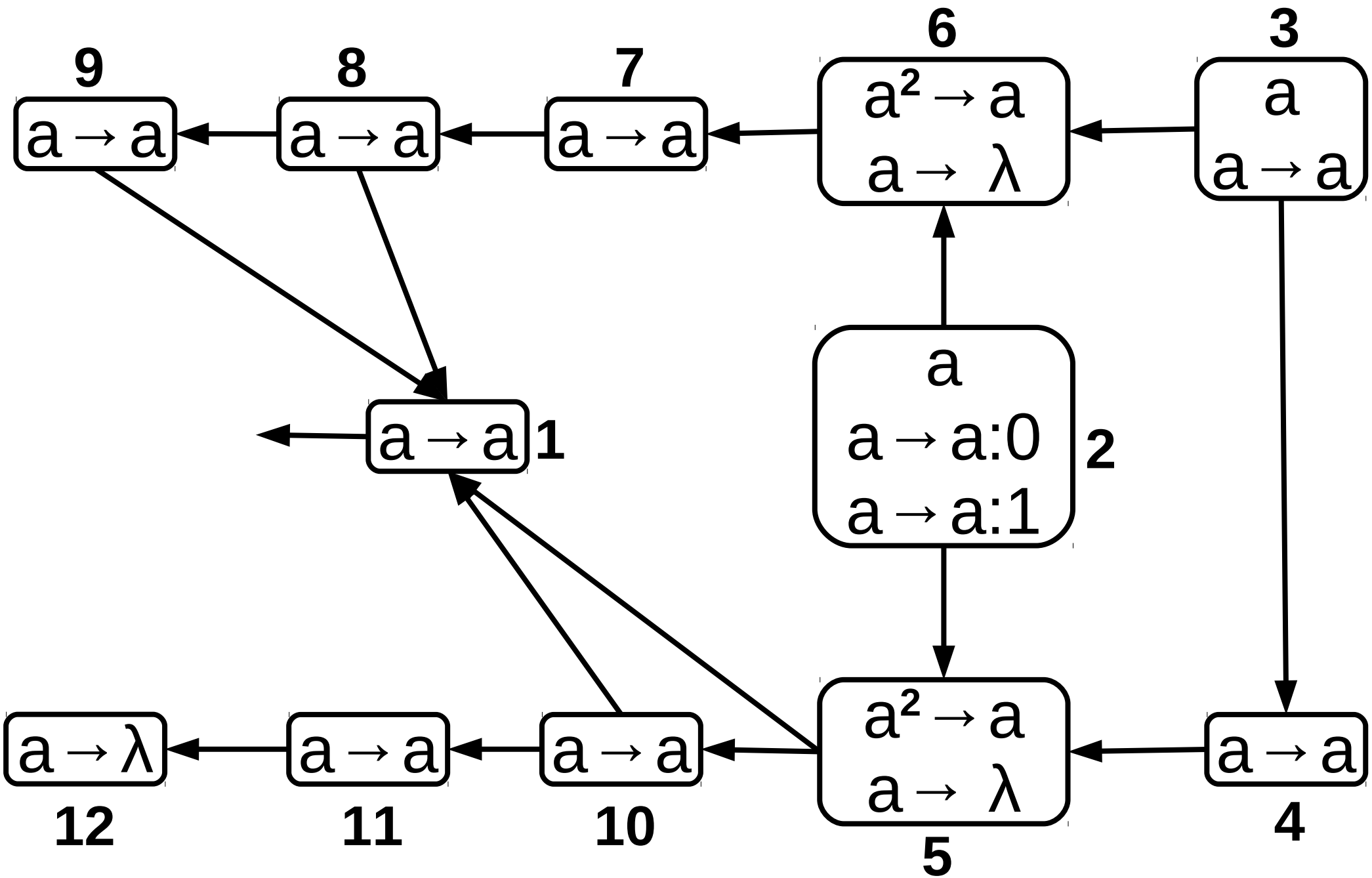
Spiking Neural P System [1]



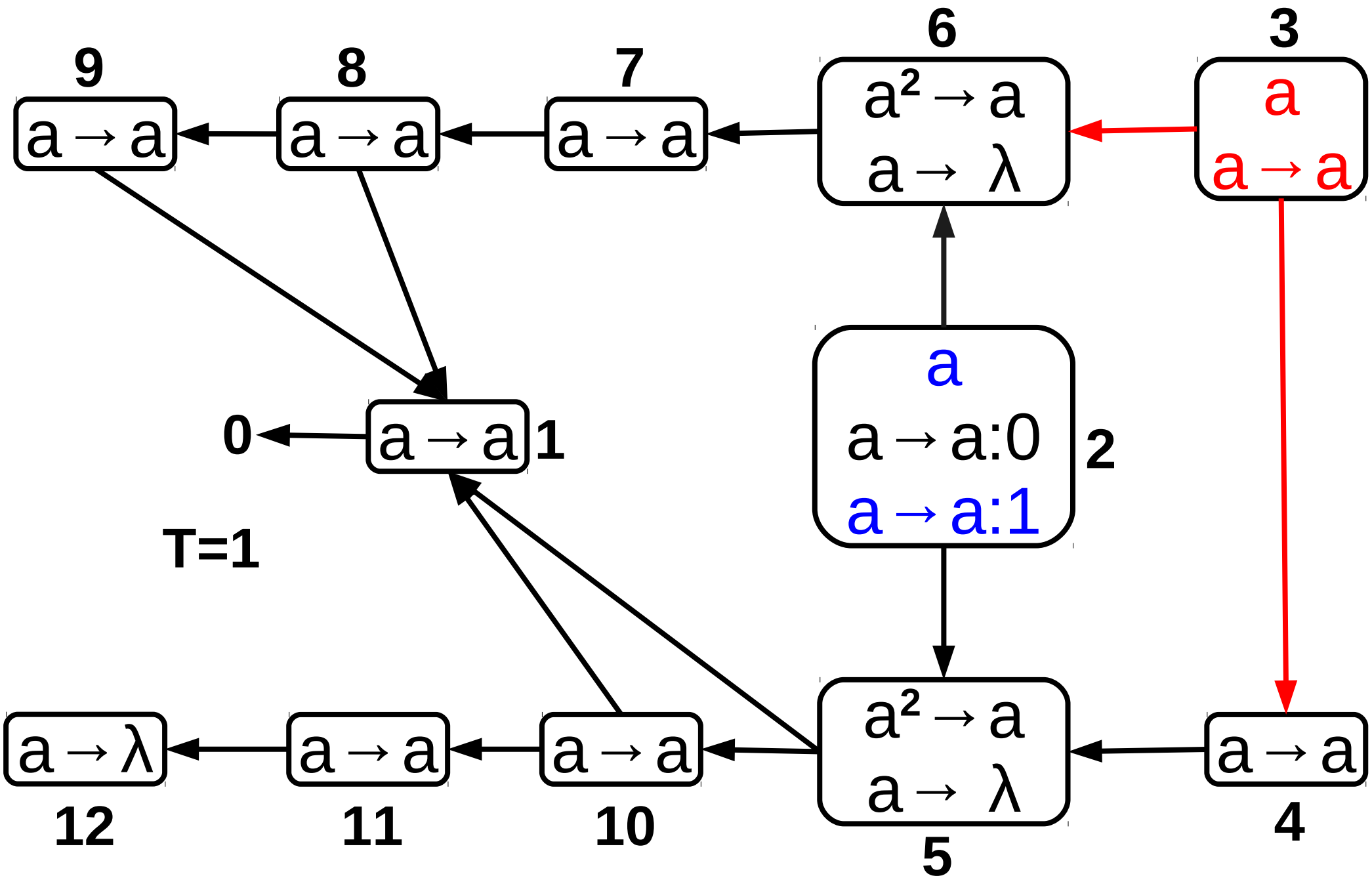
Spiking Neural P System [1]



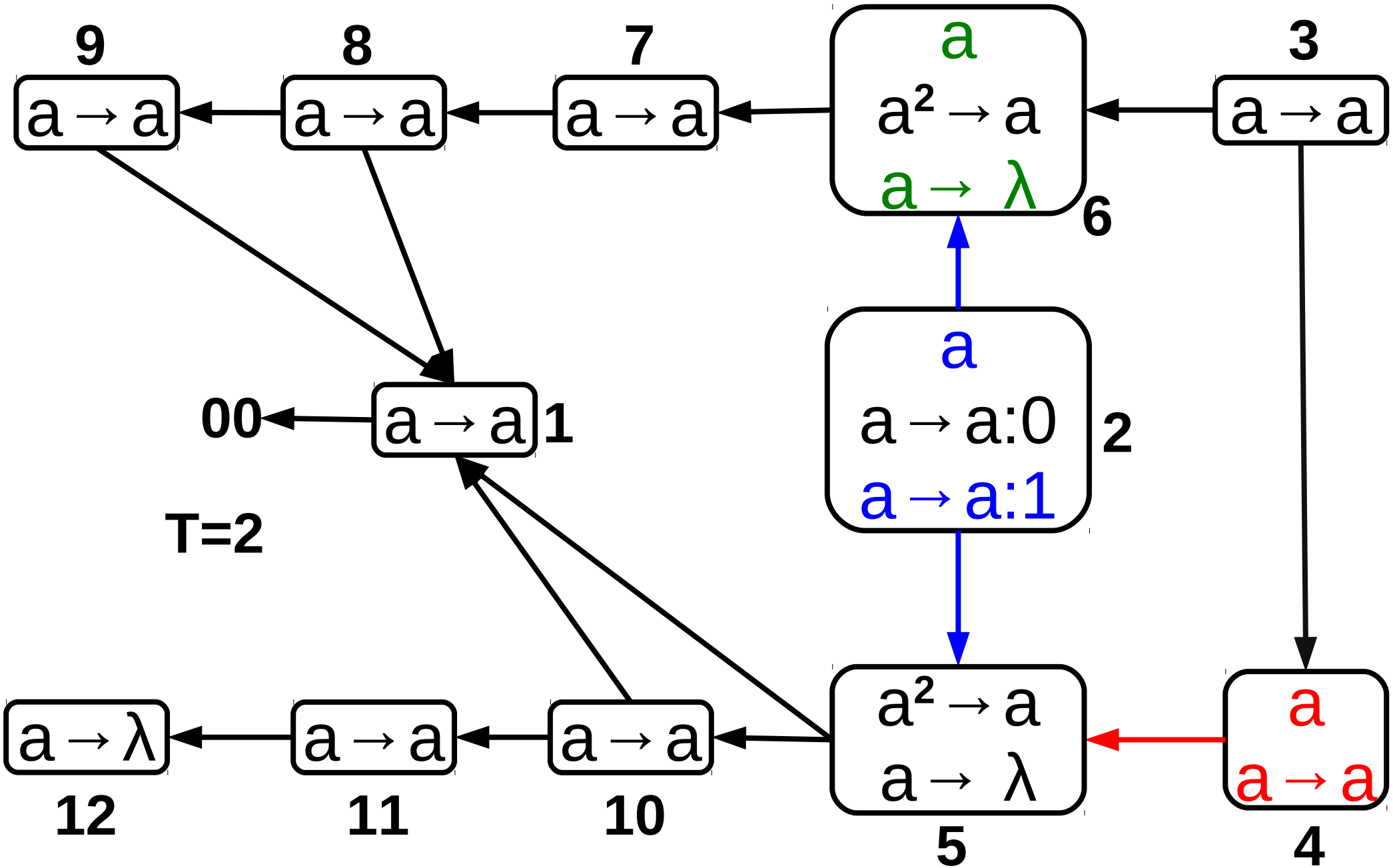
Spiking Neural P System [1]



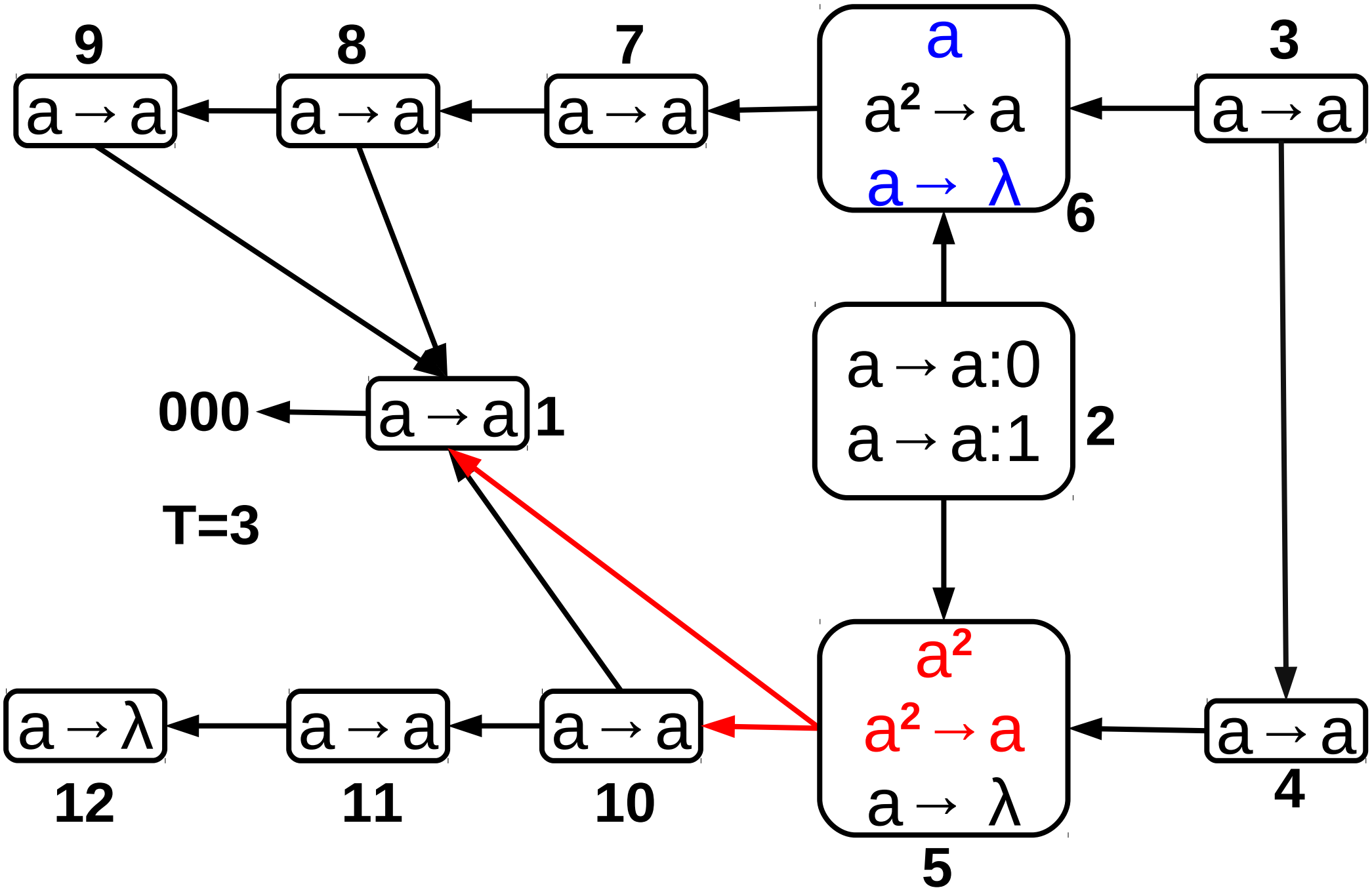
Spiking Neural P System [1]



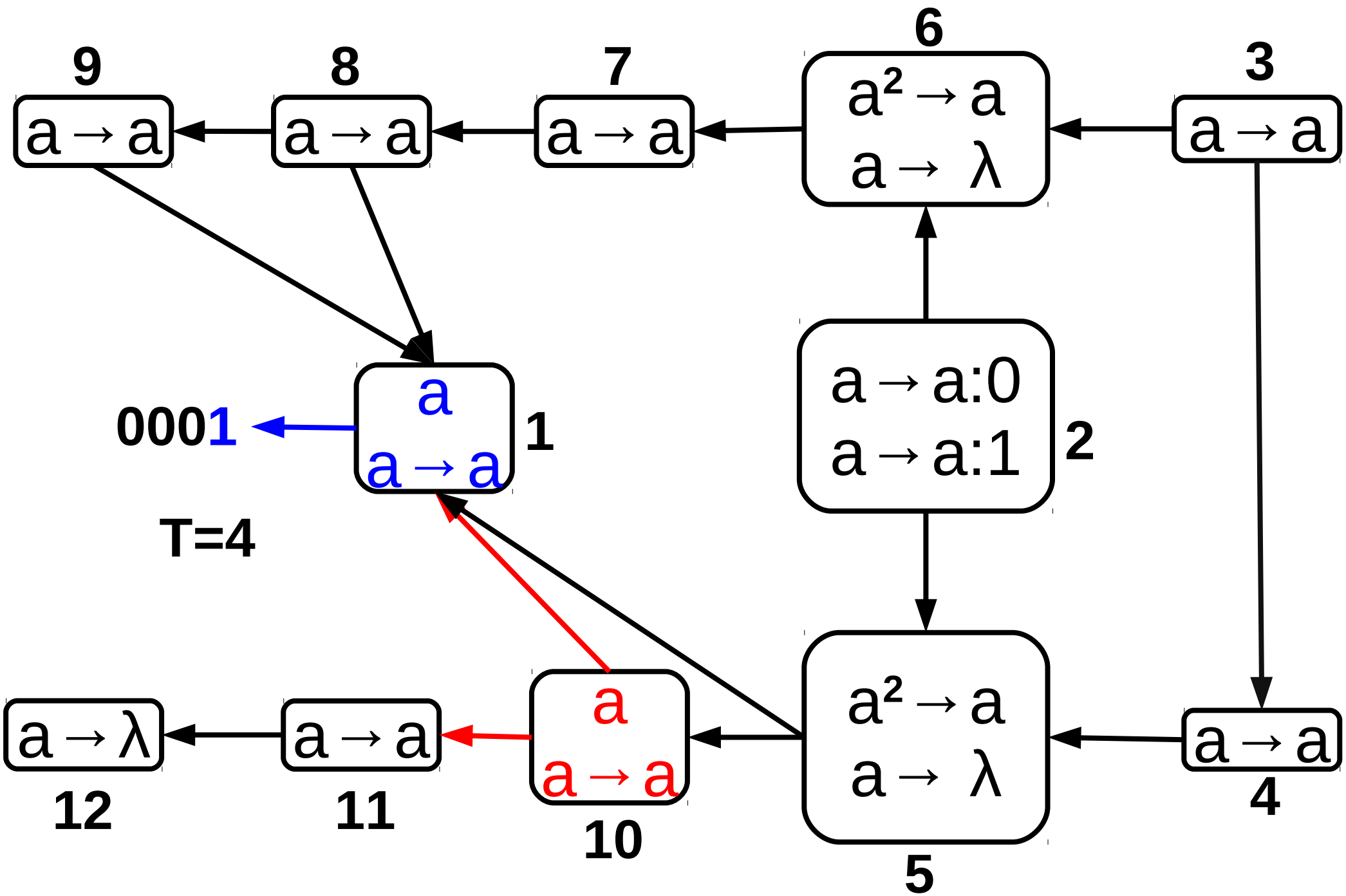
Spiking Neural P System [1]



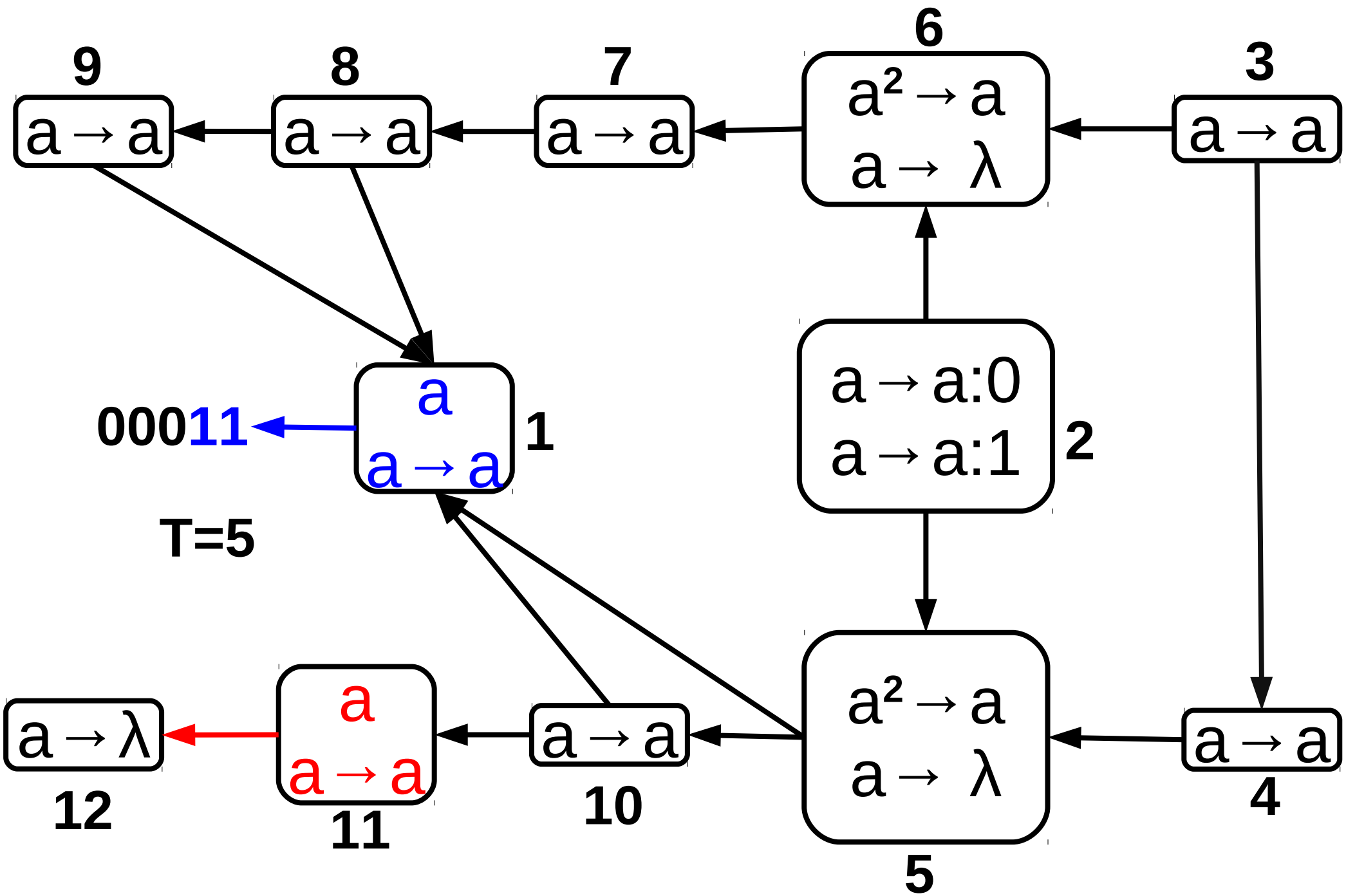
Spiking Neural P System [1]



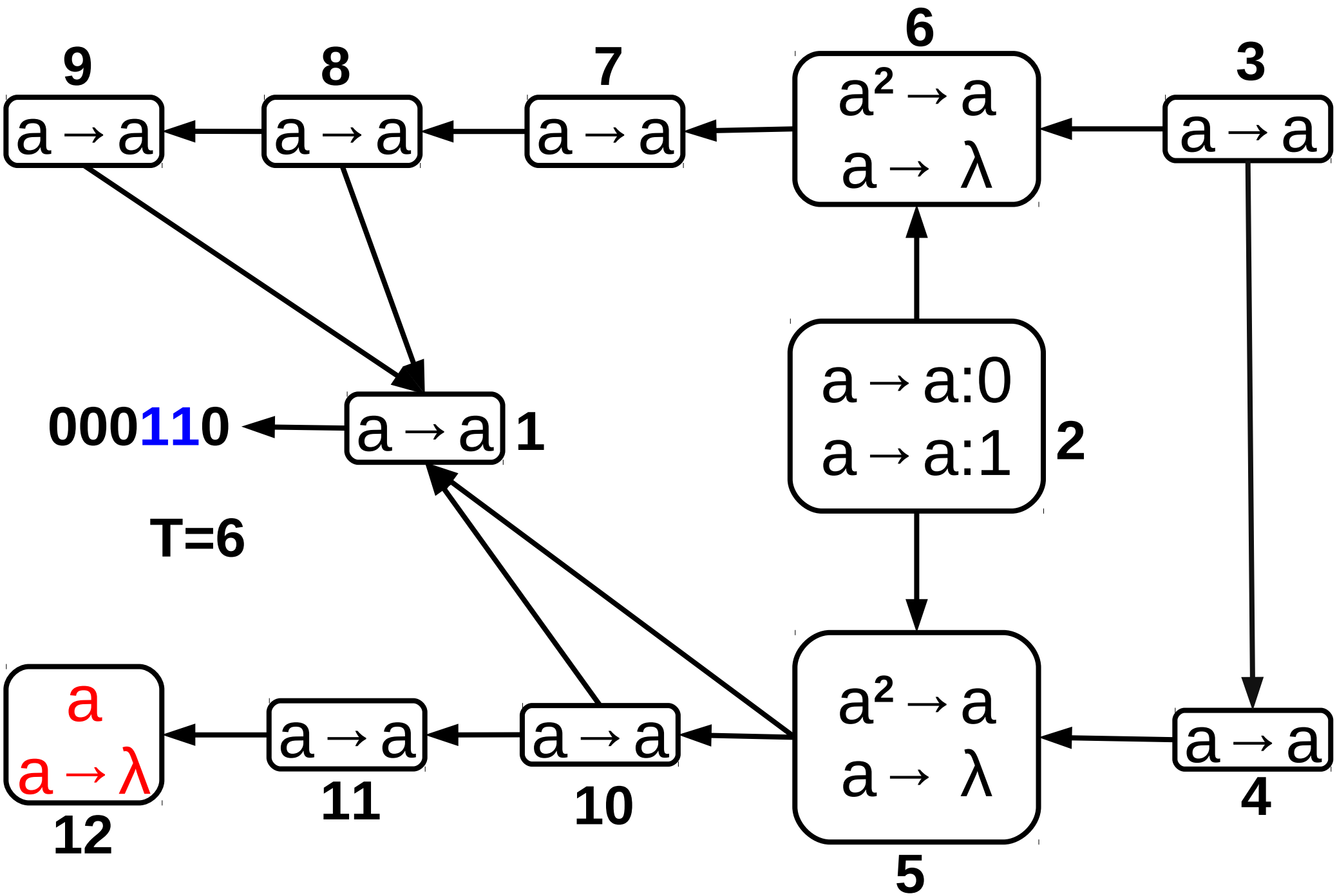
Spiking Neural P System [1]



Spiking Neural P System [1]



Spiking Neural P System [1]



Spiking Neural P System

[1] Ionescu, M., Păun, G., and Yokomori, T. (2006). Spiking neural p systems. Fundam. Inf., 71(2,3):279–308.

Spiking Neural P Systems with Structural Plasticity

(SNPSP Systems)

SNPSP System [1]

An SNPSP system **Π** of degree **m** is the construct:

$\mathbf{\Pi = (O, \sigma_1, \ldots, \sigma_m, \mathit{syn}, \mathit{out})}$

$\mathbf{O = \{a\}}$ is the object alphabet.

$\mathbf{\sigma_1, \ldots, \sigma_m}$ are the neurons of the system

$\mathbf{\mathit{syn} \subseteq \{1,\ldots,m\} \times \{1,\ldots,m\}}$ is the initial set of synapses.

$\mathbf{\mathit{out} \in \{1,\ldots,m\}}$ is the label of the designated output neuron.

SNPSP System [1]

Each neuron ($\boldsymbol{\sigma}_i$) has the following form:

$\boldsymbol{\sigma}_i = (\mathbf{n}_i, \mathbf{R}_i)$

$\mathbf{n}_i \geq \mathbf{0}$ is the initial number of spikes in $\boldsymbol{\sigma}_i$.

$\mathbf{R}_i$ is the finite set of rules of $\boldsymbol{\sigma}_i$.

A rule $\mathbf{r}_j \in \mathbf{R}_i$ is either one of two possible forms.

(1) <u>Spiking Rule</u>: $\mathbf{E} \,/\, \mathbf{a}^c \rightarrow \mathbf{a}$ (no delay)

(no forgetting rule)

SNPSP System [1]

(2) Plasticity Rule: $\mathbf{E} / \mathbf{a}^c \rightarrow \alpha k(i, N)$

$\mathbf{E} / \mathbf{a}^c$ - (same to SNP systems)

i is the label of the neuron containing the rule.

$\alpha \in \{+,-,\pm, \mp\}$ is the action to be performed.

$N \subseteq \{1,\ldots,m\}$ is the set of neuron labels of i's "neighbor" neurons.

$1 \leq k \leq |N|$ is the number of synapses to be added or deleted.

SNPSP System [1]

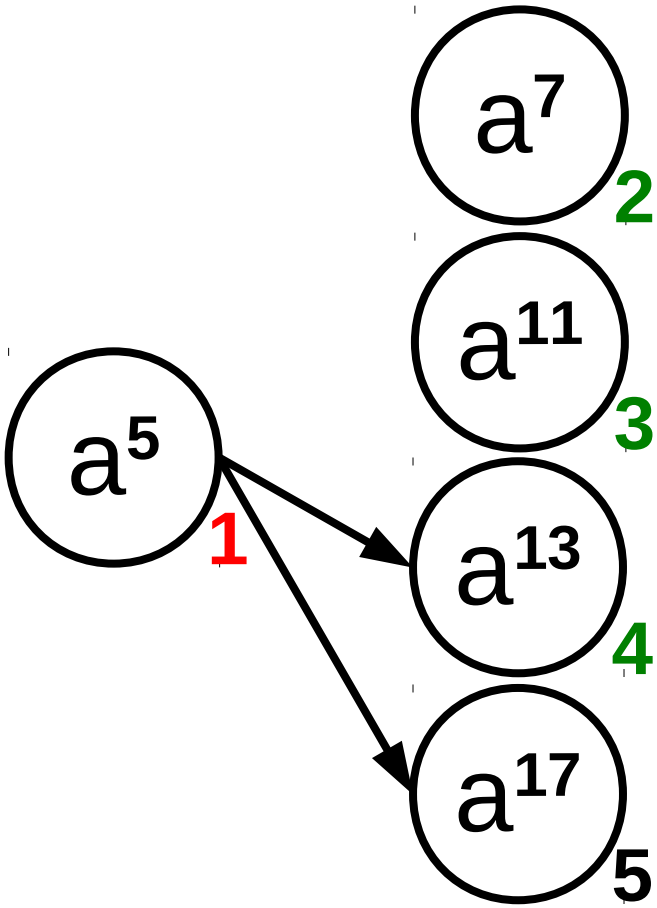
(2) Plasticity Rule: $\mathbf{E} / \mathbf{a}^c \rightarrow \alpha k(i, N)$

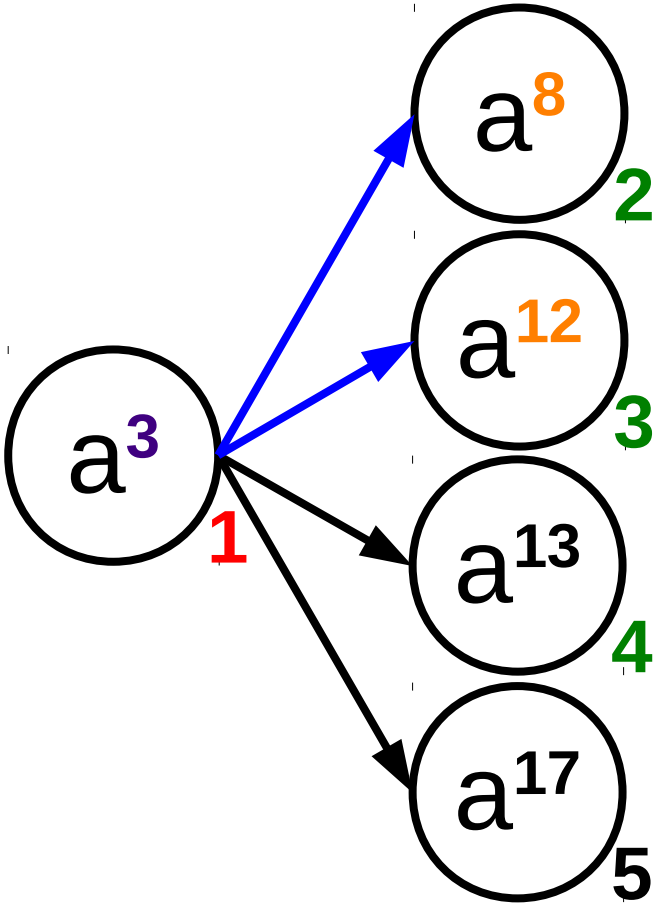
The action "**+k(i, N)**" means to *attempt* to add **k** synapses from neuron **i** to neurons **n** $\in$ **N**. When a synapse is created from neuron **i** to neuron **n**, a spike is immediately sent to neuron **n**.

The action "**-k(i, N)**" means to *attempt* to delete **k** synapses that connect neuron **i** to neurons **n** $\in$ **N**.

SNPSP System [1]

r: a(aa)* / a^2 → +2(1, {2,3,4})

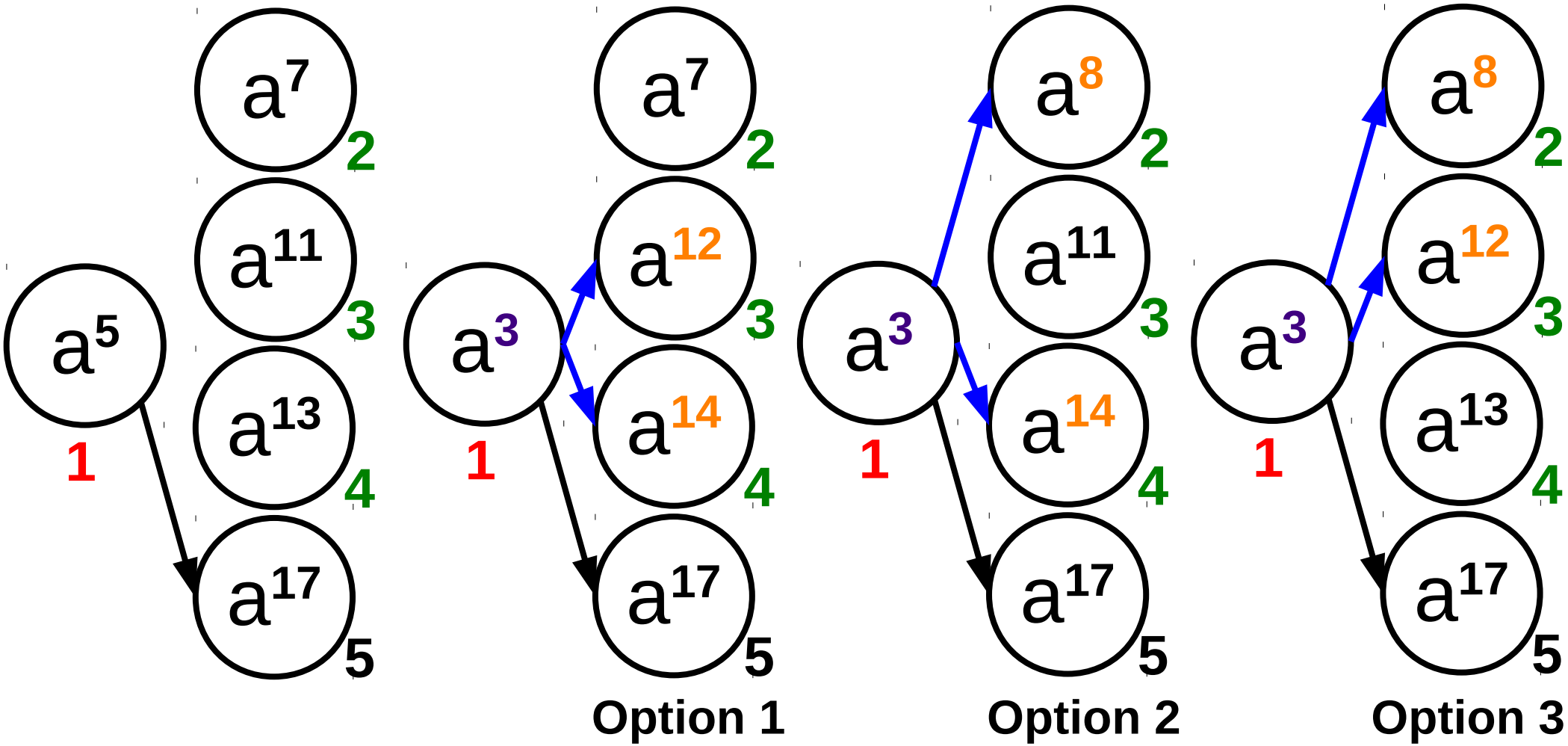




Case 1: Exactly k synapses were added. No non-deterministic selection was performed.

SNPSP System [1]

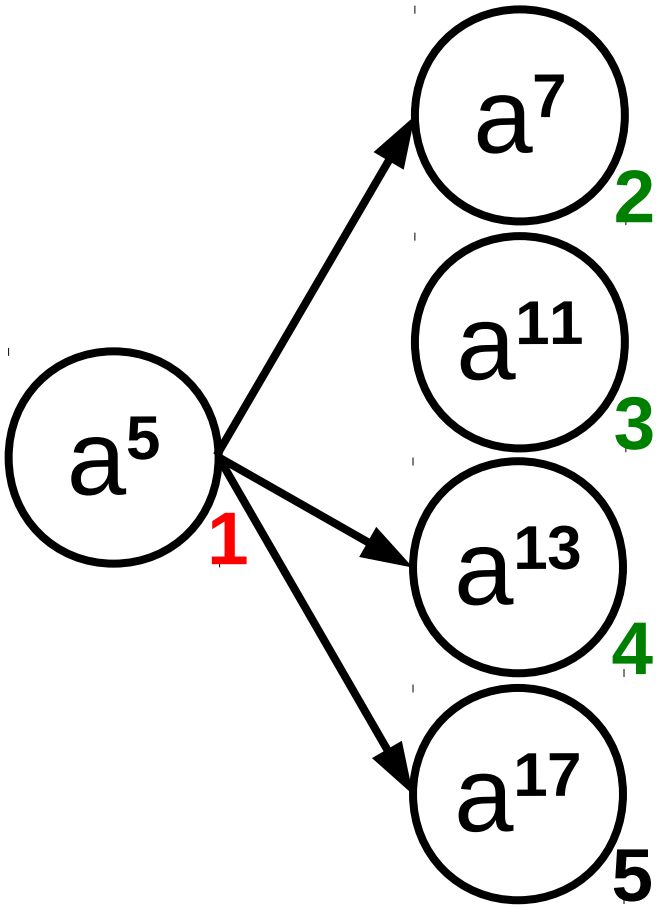
r: a(aa)* / a^2 → +2(1, {2,3,4})

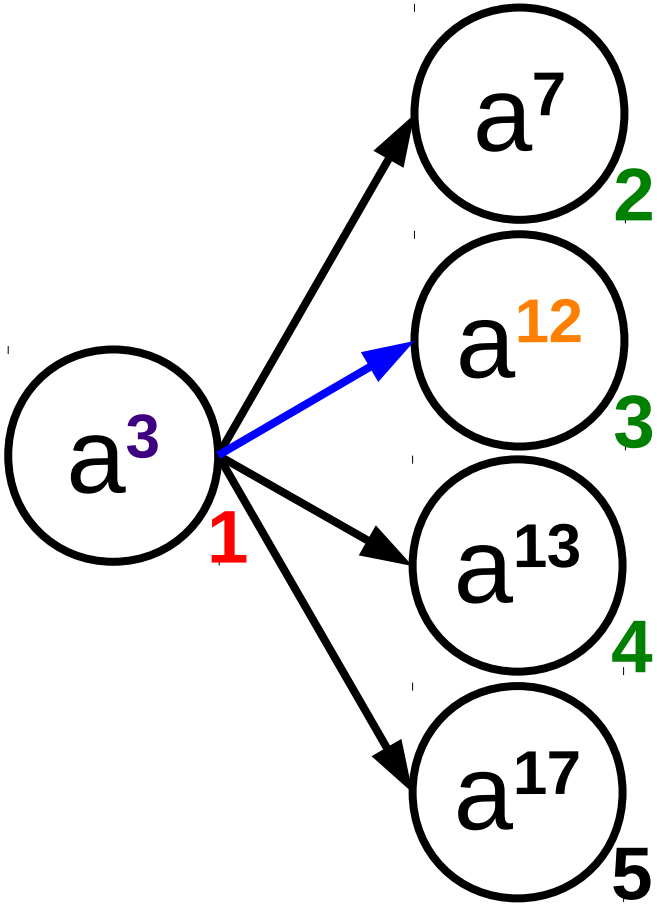


Case 2: Exactly k synapses were added. A <u>non-deterministic</u> selection was performed.

SNPSP System [1]

r: a(aa)* / a^2 → +2(1, {2,3,4})

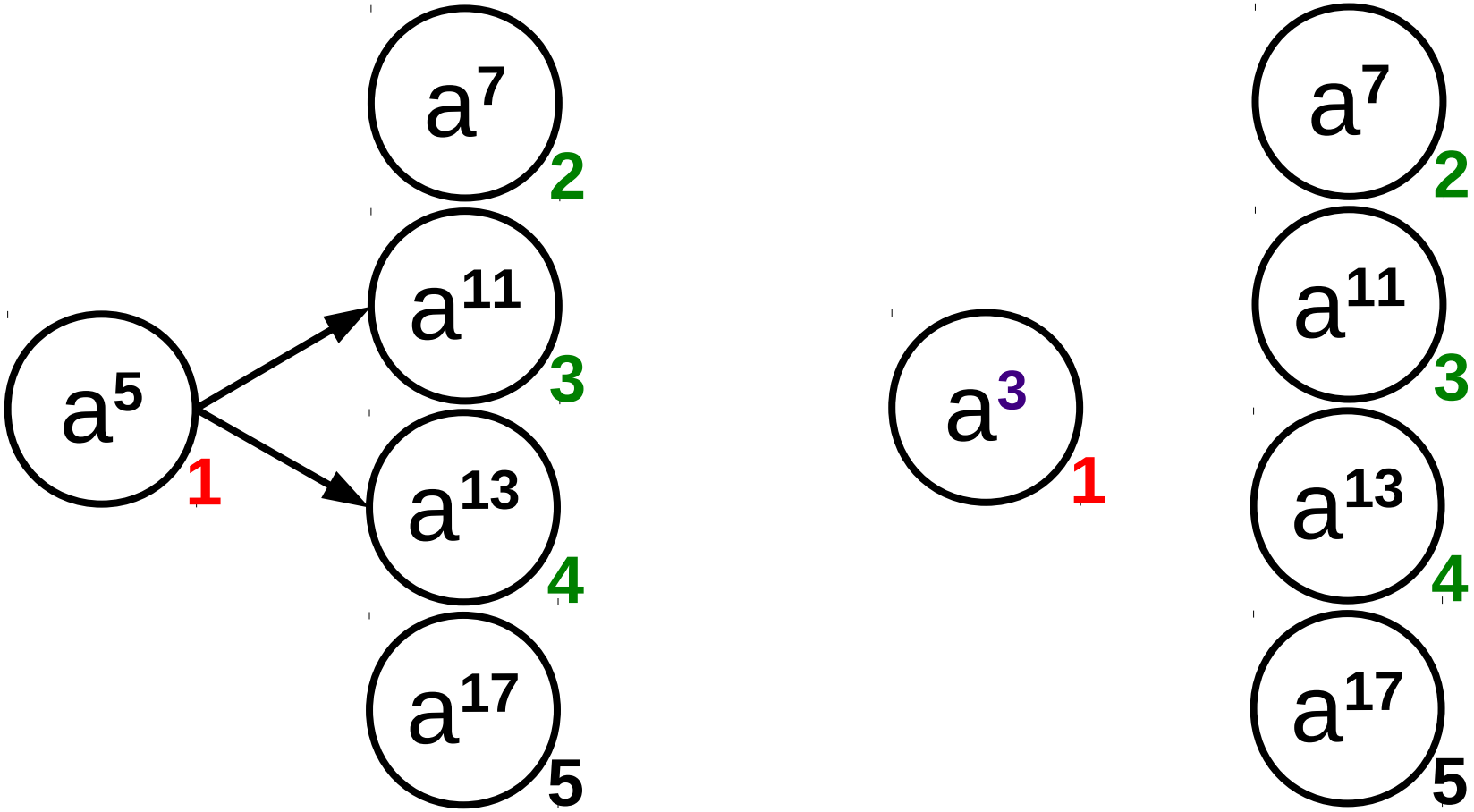




Case 3: Less than k synapses were added. No non-deterministic selection was performed.

SNPSP System [1]

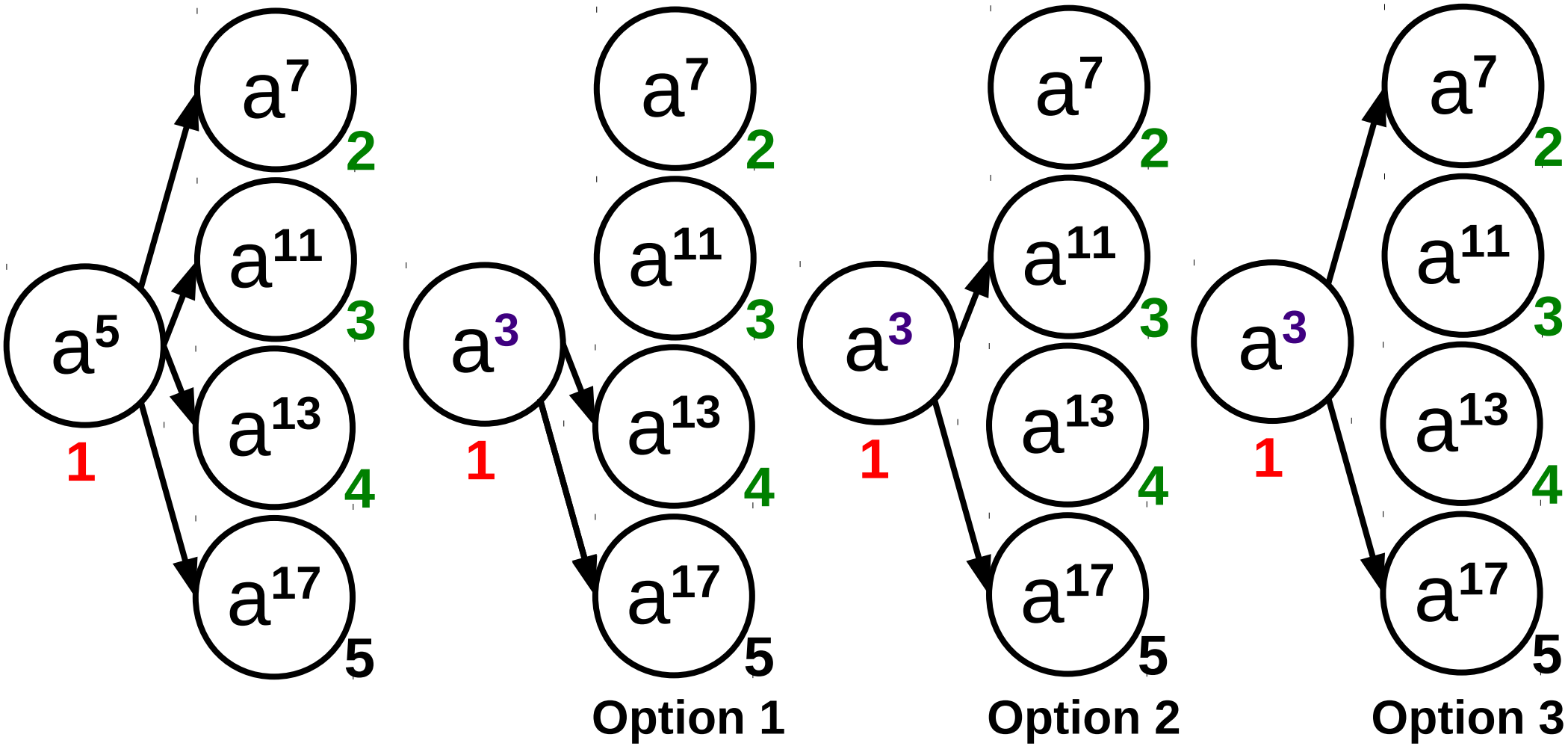
r: a(aa)* / a^2 → -2(1, {2,3,4})



Case 1: Exactly k synapses were deleted. No non-deterministic selection was performed.

SNPSP System [1]

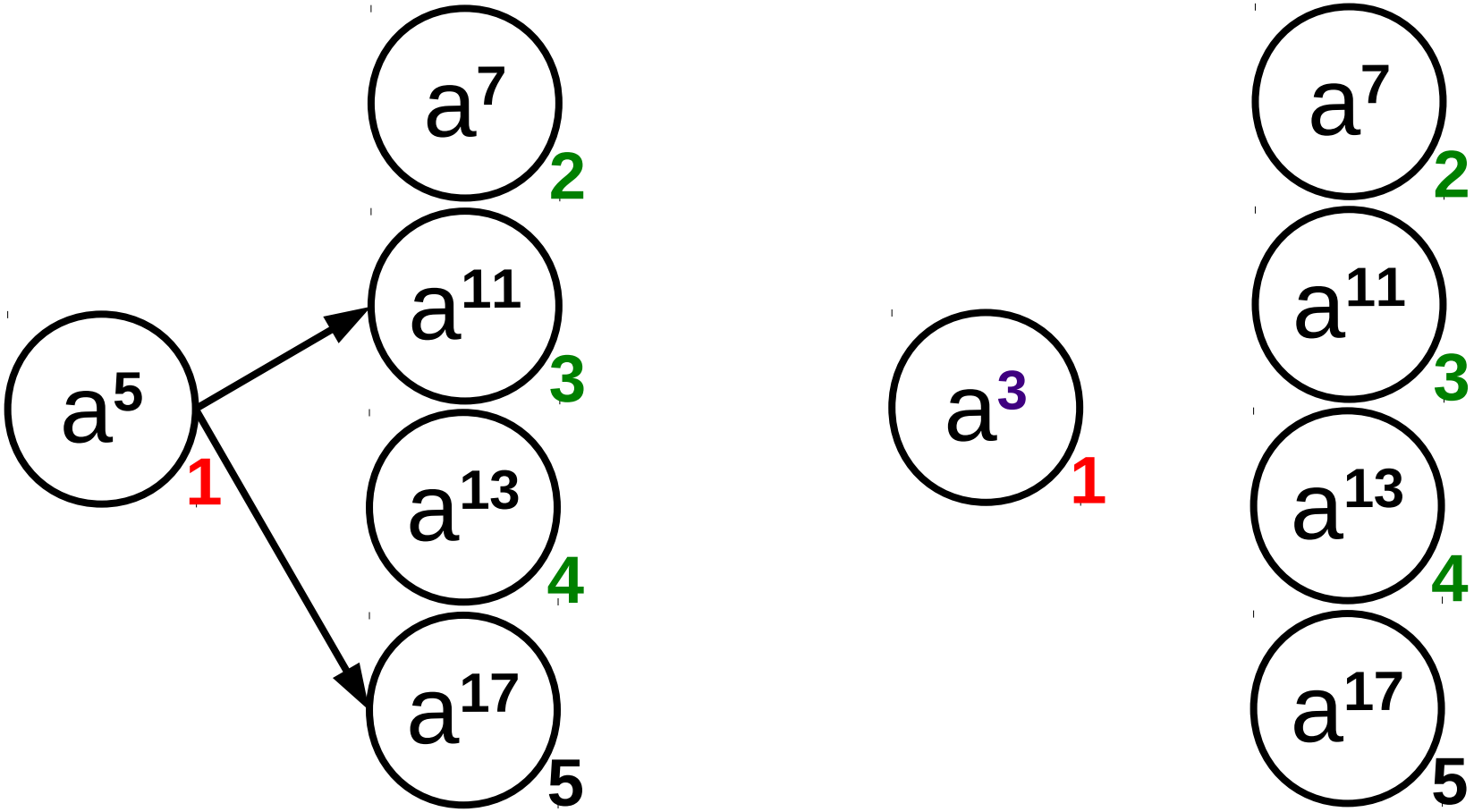
r: a(aa)* / a^2 → -2(1, {2,3,4})



Case 2: Exactly k synapses were deleted. A <u>non-deterministic</u> selection was performed.

SNPSP System [1]

r: a(aa)* / a^2 → -2(1, {2,3,4})



Case 3: Less than k synapses were deleted. No non-deterministic selection was performed.

SNPSP System [1]

(2) Plasticity Rule: $\mathbf{E} / \mathbf{a}^c \rightarrow \alpha k(i, N)$

The action "$\pm k(i, N)$" means to *attempt* to **add** k synapses from neuron i to neurons $n \in N$ at time t then *attempt* to **delete** k synapses that connect neuron i to neurons $n \in N$ at time $t+1$.

The action "$\mp k(i, N)$" means to *attempt* to **delete** k synapses that connect neuron i to neurons $n \in N$ at time t then *attempt* to **add** k synapses from neuron i to neurons $n \in N$ at time $t+1$.

SNPSP System [1]

It's possible for a neuron to have multiple plasticity rules,

$\mathbf{r_1 : E_1 / a^{c1} \rightarrow \alpha_1 k_1(i, N_1)}$

$\mathbf{r_2 : E_2 / a^{c2} \rightarrow \alpha_2 k_2(i, N_2)}$

that can be activated at the same time. It can occur when $L(\mathbf{E_1}) \cap L(\mathbf{E_2}) \neq \emptyset$.

In case multiple plasticity rules can be activated, the neuron <u>*non-deterministically*</u> select a rule to activate.

SNPSP System [1]

The following conventions are used for writing plasticity rules:

If **E = a^c**, rule **r : "E / a^c → αk(i, N)"** can be written as **r : "a^c → αk(i, N)"**.

SNPSP System

[1] Cabarle, F. G. C., Adorna, H. N., Pérez-Jiménez, M. J., and Song, T. (2015). Spiking neural p systems with structural plasticity. Neural Computing and Applications, 26(8):1905–1917.

SNP Systems as Language Generator

SNP Systems as Language Generator

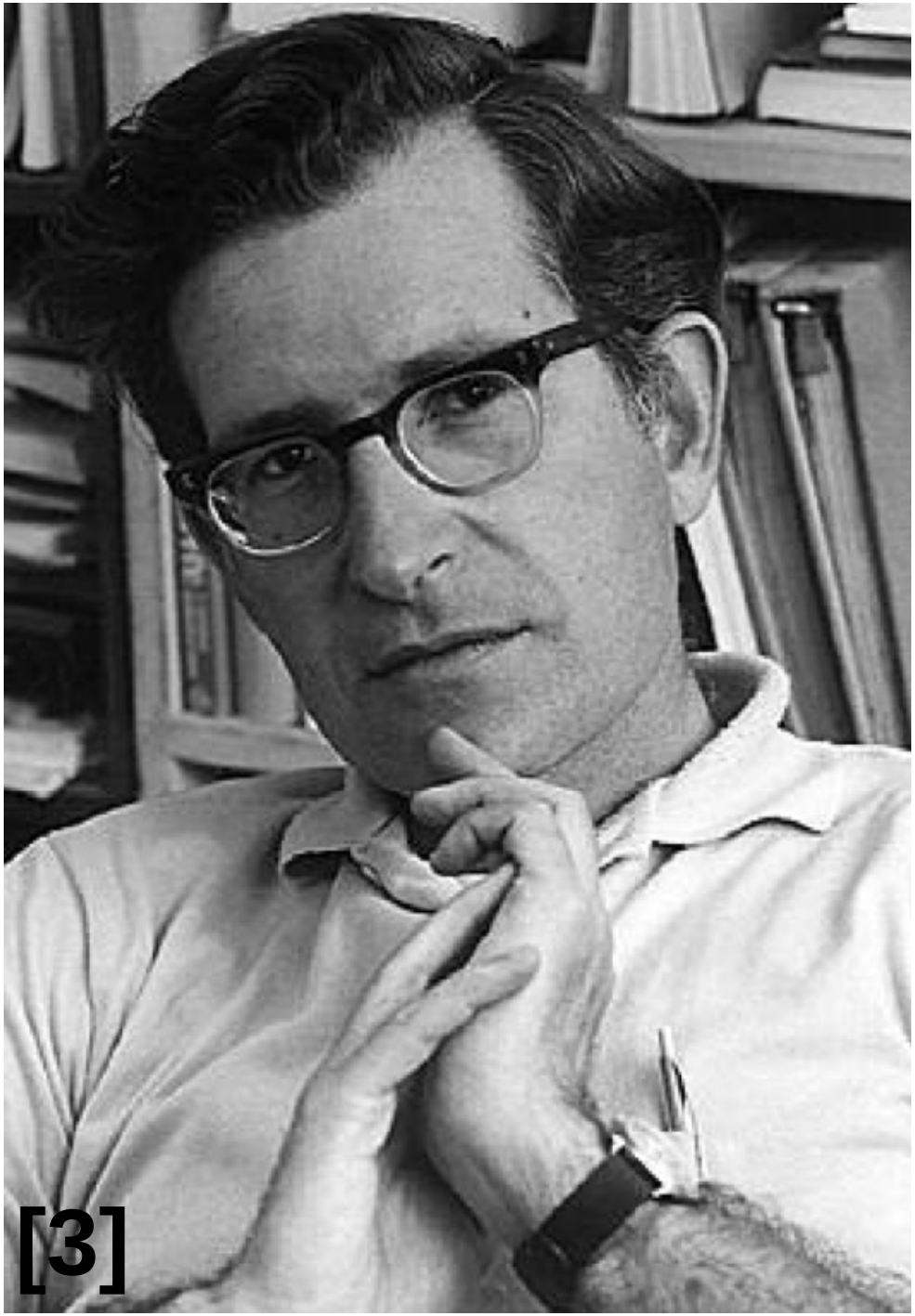
Chomsky Hierarchy	Grammar	Language	Machine
Type-0	Unrestricted	Recursively Enumerable	Turing Machine
Type-1	Context-Sensitive	Context-Sensitive	Linear-bounded
Type-2	Context-Free	Context-Free	Nondeterministic Pushdown
Type-3	Regular	Regular	Finite State

SNP Systems as Language Generator

Before the 1970's, the link between families of languages, formal grammars, and abstract machines (computing models) have been established by mathematicians and linguists.

Four of the main contributors were Chomsky, Schutzenberger, Myhill and Kuroda.

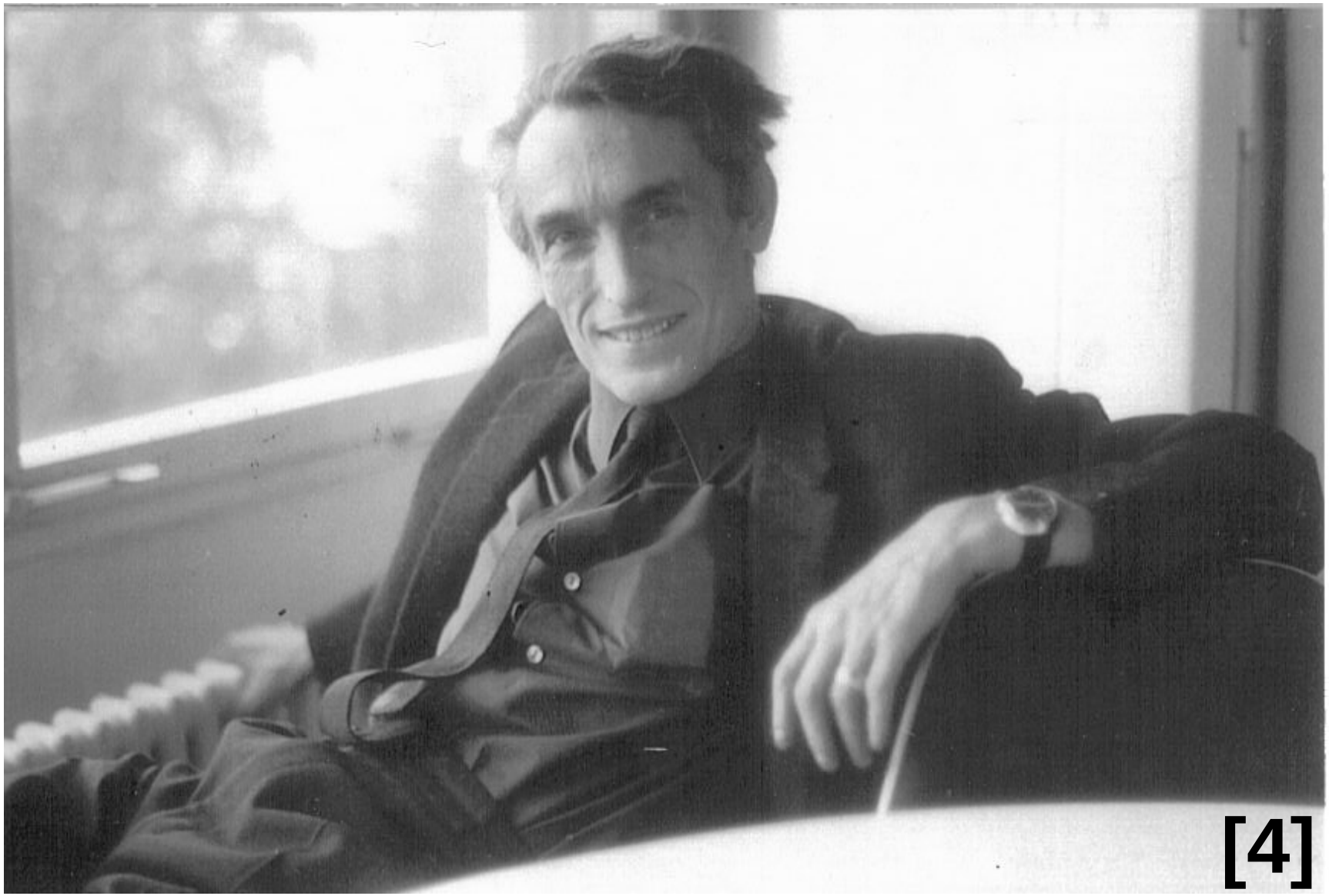
SNP Systems as Language Generator



Noam Chomsky

SNP Systems as Language Generator

Marcel-Paul Schützenberger



SNP Systems as Language Generator

John Myhill

SNP Systems as Language Generator

Sige-Yuki Kuroda

SNP Systems as Language Generator

Since there is an intimate link between languages and Turing machines (and its restricted versions), when a new computing model is introduced it is natural to ask how the new computing model can be used to generated languages.

SNP Systems as Language Generator

SNP systems were initially used as number generators. In **[5]**, **Chen et.al**. Used SNP sytems as language generators. They obtain the following results:

1. Procedure for constructing SNP systems for finite (*FIN*) languages.

If $\mathbf{L \in}$ ***FIN***, $\mathbf{L \subseteq \{0,1\}^+}$, $\mathbf{L = \{x_1,\ldots,x_n\}}$, then the procedure can construct a $\mathbf{\Pi}$ that generates $\mathbf{L' = \{0^{i+3}x_i \mid 1 \leq i \leq n\}}$.

SNP Systems as Language Generator

2. Procedure for constructing SNP systems for regular (*REG*) languages.

If $\mathbf{L} \in \boldsymbol{REG}$, $\mathbf{L} \subseteq \boldsymbol{V}^{+}$, then the procedure can construct a $\mathbf{\Pi}$ that generates $\mathbf{L' = L(\Pi)}$ and a morphism $\mathbf{h: V^* \rightarrow \{0,1\}^*}$ such that $\mathbf{L = h^{-1}(L')}$.

SNP Systems as Language Generator

3. Procedure for constructing SNP systems for recursively enumerable (*RE*) languages.

If $\mathbf{V = \{a_1, ..., a_k\}}$, $\mathbf{L \subseteq V^*}$, $\mathbf{L \in RE}$, then the procedure construct $\mathbf{\Pi}$ and morphisms:

$\mathbf{h_1 : (V \cup \{b, c\})^* \rightarrow \{0,1\}^*}$

$\mathbf{h_2 : (V \cup \{b, c\})^* \rightarrow V^*}$

such that the language generate by $\mathbf{\Pi}$, $\mathbf{L' = L(\Pi)}$ is related to $\mathbf{L}$ as follows:

$\mathbf{L = h_2(h_1^{-1}(L'))}$.

SNP Systems as Language Generator

[1] Kuroda Image: http://library.ucsd.edu/dc/object/bb3004443n/zoom/0

[2] Myhill Image:

https://www.maa.org/book/export/html/117813

[3] Chomsky Image:

https://libcom.org/library/noam-chomsky-reading-guide

- **[4]** Schützenberger Image:

http://mps2016.labri.fr/index_en.html

SNP Systems as Language Generator

[5] Chen, H., Freund, R., Ionescu, M., Păun, G., and Pérez-Jiménez, M. J. (2007). *On string languages generated by spiking neural P systems.* Fundam. Inform., 75(1-4):141–162.

Thesis

Problems

1. How can SNPSP systems be used to generate finite, regular, and recursively enumerable languages?

2. How do the resulting SNPSP systems for generating finite, regular, and recursively enumerable languages differ to their counter-part SNP systems or SNP systems with extended rules that generate finite, regular, and recursively enumerable languages?

Problems

3. How are plasticity rules used in these SNPSP systems?

4. How can SNPSP systems be used to generate context-free (*CF*) languages?

Objectives

1. For each of family (*FIN, REG, CF, RE*) of languages, create at least one procedure forconstructing SNPSP systems that generate the languages in that family.

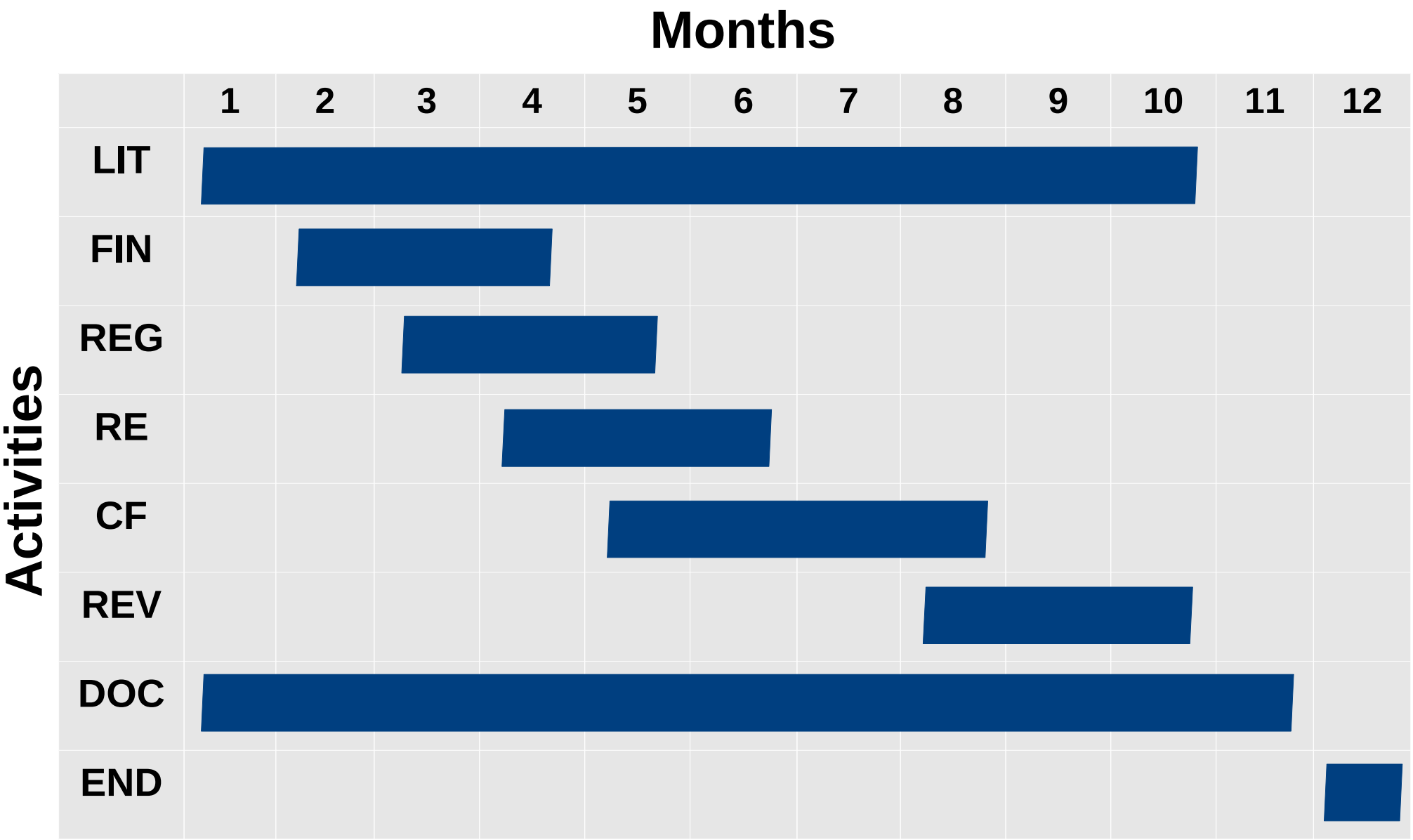
2. Compare the SNPSP systems for generating *FIN*, *REG* and *RE* languages to their counter-part SNP systems and SNP systems with extended rules.

Objectives

3. Discuss how plasticity rules are used in the resulting SNPSP systems.

4. Provide recommendations on how the technique for creating SNPSP systems that generate *CF* languages can be applied to classic SNP systems and other variants.

Schedule


Months
1
2
3
4
5
6
7
8
9
10
11
12
Activities
LIT
FIN
REG
RE
CF
REV
DOC
END

Schedule

LIT: General literature review.

FIN: Create procedure for ***FIN*** languages.

REG: Create procedure for ***REG*** languages

RE: Create procedure for ***RE*** languages

CF: Create procedure for ***CF*** languages

REV: Results revision.

DOC: Results documentation (publication).

END: Thesis defense + manuscript submission.