On String Languages Generated by Spiking Neural P Systems with Structural Plasticity

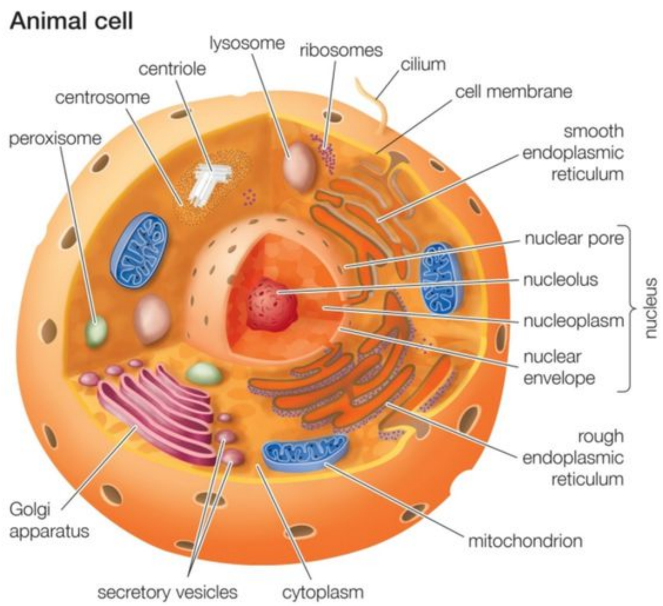
Master of Science Thesis by
Ren Tristan A. de la Cruz

Algorithms and Complexity Laboratory
Department of Computer Science
University of the Philippines
Diliman, Quezon City, Philippines

2018 May 11

Introduction

Introduction



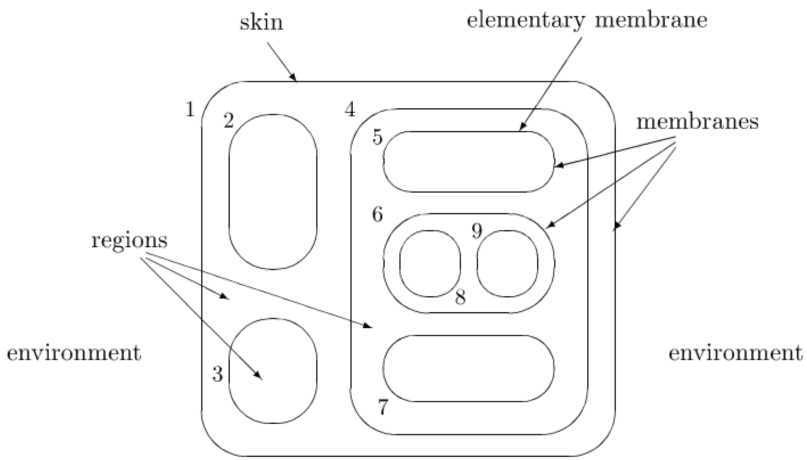
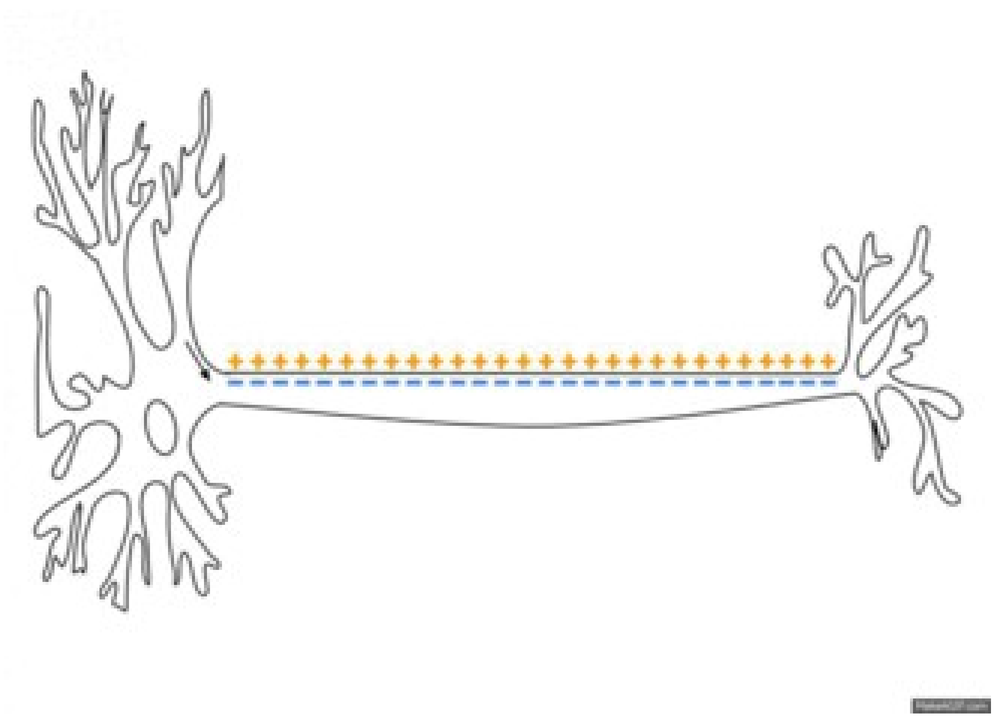
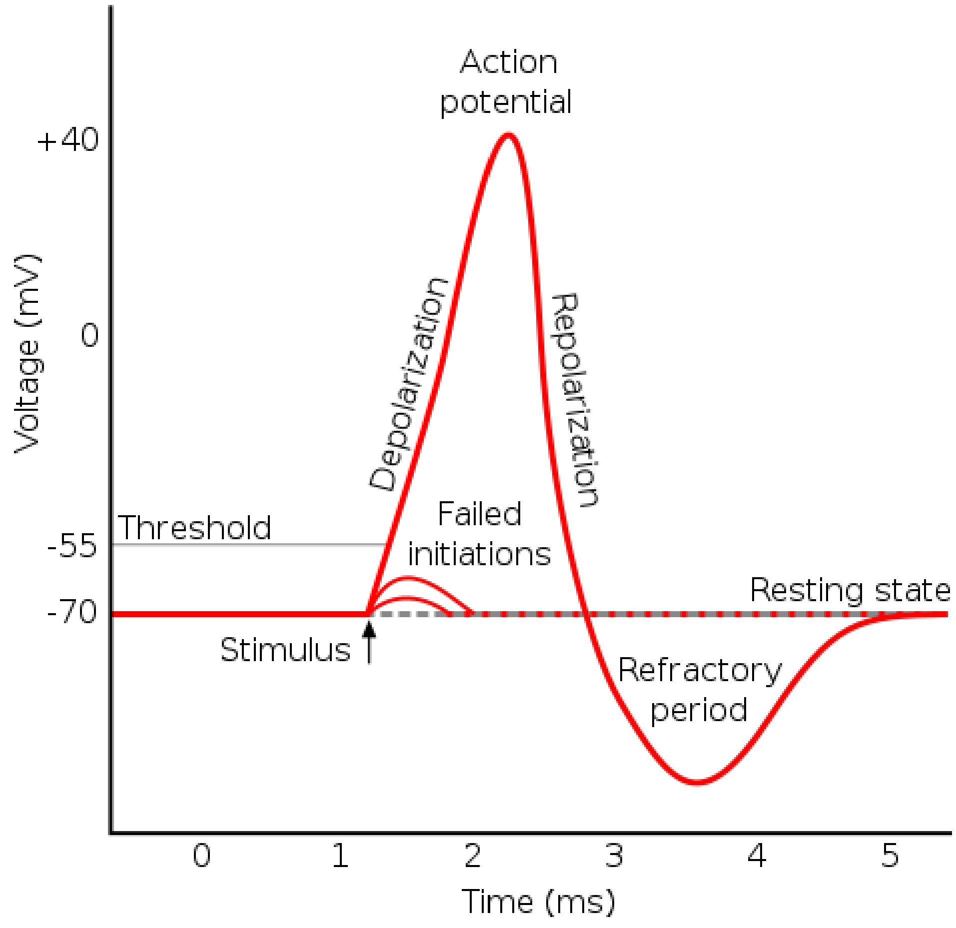


Figure 1: A membrane structure

1998: Computing with Membranes (P Systems)

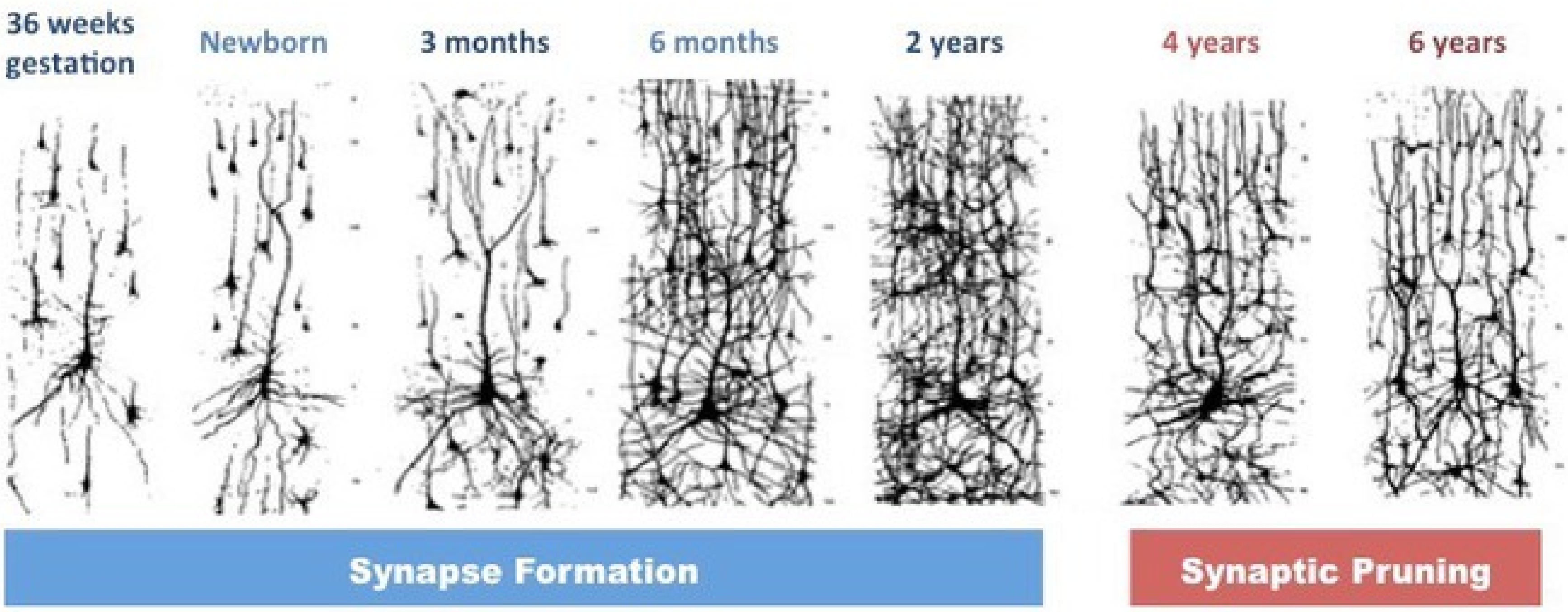
Introduction





2007: Spiking Neural P Systems

Introduction



2015: Spiking Neural P Systems with Structural Plasticity

Contributions

Contributions

Results:

1. Two procedures for constructing **SNPSP systems** that generate ***FIN*** languages

2. Two procedures for constructing SNPSP systems that generate ***REG*** languages

3. Two versions of a procedure for constructing SNPSP systems that generate ***RE*** languages

4. An implementation of an SNPSP module called **Arithmetic-Memory module** that performs arithmetic operations

Contributions

Results:

5. A procedure for constructing an **SNPSP** system that generates ***CF*** languages

6. A way for **simulating** SNP system's **forgetting rules** in SNPSP systems

7. Ways for **simulating** the '**delay**' aspect of an SNP system's spiking rule in SNPSP systems

8. A way for **simulating**, in SNP systems, some aspects of SNPSP system's **plasticity rules**

SNP & SNPSP Systems

SNP & SNPSP Systems (1/11)

SNP system Π of degree **m:**

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

Alphabet: **O = {a}, a** - spike

Synapses: $syn \subset \{1,\ldots,m\} \times \{1,\ldots,m\}$

Neurons: $\sigma_1, \ldots, \sigma_m$

$\sigma_i = (n_i, R_i)$

n_i - initial number of spikes

R_i - set of rules of the neuron

Output Neuron (label):

$out \in \{1,\ldots,m\}$

SNPSP system Π of degree **m:**

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

Alphabet: **O = {a}, a** - spike

Initial Synapses: $syn \subset \{1,\ldots,m\} \times \{1,\ldots,m\}$

Neurons: $\sigma_1, \ldots, \sigma_m$

$\sigma_i = (n_i, R_i)$

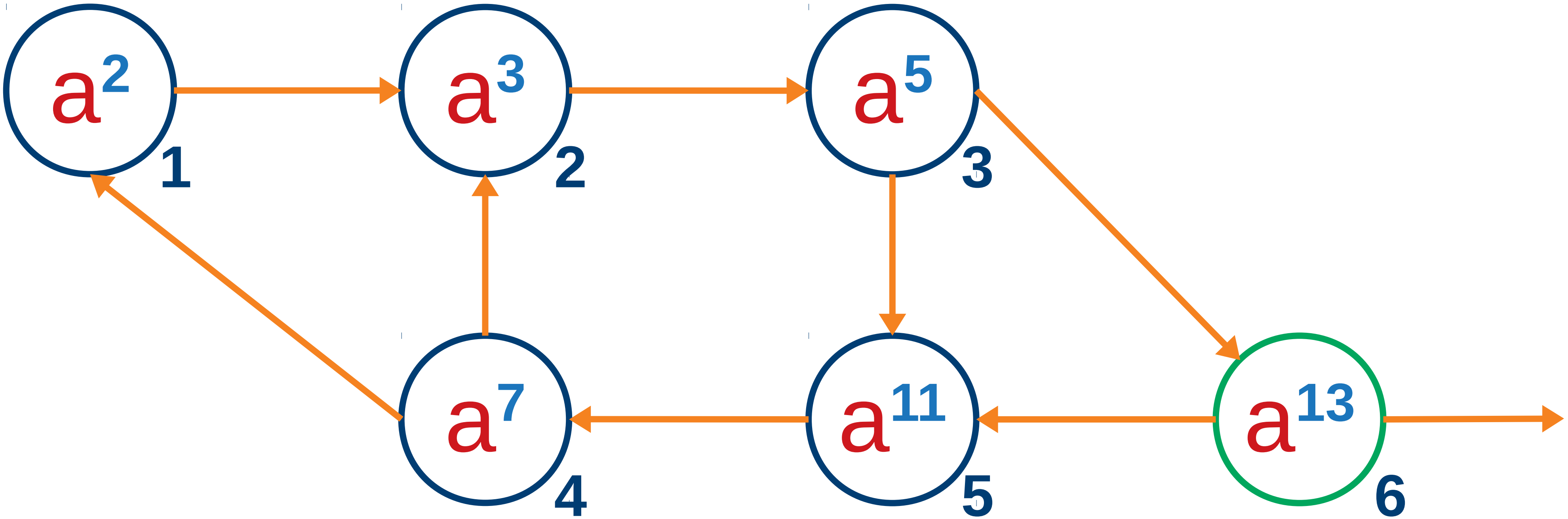
n_i - initial number of spikes

R_i - set of rules of the neuron

Output Neuron (label):

$out \in \{1,\ldots,m\}$

SNP & SNPSP Systems (2/11)



SNP/SNPSP system Π of degree **m:** $\Pi = (O, \sigma_1, \ldots, \sigma_6, syn, 6)$

Alphabet: **O = {a}**

Synapses: **syn =** {(1,2), (2,3), (3,5), (3,6), (4,1), (4,2), (5,4)}

Neurons: $\sigma_1=(2,R_1)$, $\sigma_2=(3,R_2)$, $\sigma_3=(5,R_3)$, $\sigma_4=(7,R_4)$,

$\sigma_5=(11,R_5)$, $\sigma_6=(13,R_6)$

Output Neuron (label): ***6***

SNP & SNPSP Systems (3/11)

SNP system Rules:

Neurons: $\sigma_i = (n_i, R_i)$

R_i - set of rules of the neuron

Rule: $r_j \in R_i$

r_j is a rule in R_i. It can have either of the following forms:

Spiking Rule (Form):

$E / a^c \rightarrow a{:}d$

Forgetting Rule (Form):

$a^c \rightarrow \lambda$

SNPSP system Rules:

Neurons: $\sigma_i = (n_i, R_i)$

R_i - set of rules of the neuron

Rule: $r_j \in R_i$

r_j is a rule in R_i. It can have either of the following forms:

Spiking Rule (Form):

$E / a^c \rightarrow a$

Plasticity Rule (Form):

$E / a^c \rightarrow \alpha k(i, N)$

SNP & SNPSP Systems (4/11)

Rule's Activation Criteria: **E / a^c**

1) **a^c – c** is positive integer. **a^c** means that the neuron that contains the rule should have at least **c** spikes.

2) **E** – Regular Expression over **$O = \{a\}$**. The rule can only activate when the number of spikes (represented by **a^n**) in the neuron is 'covered' by the regular expression **E**.

$a^n \in L(E)$.

Examples:

$a^n=a^7$, $E_1=a^2(a^5)^*$ – $a^7 \in L(E_1)$ – OK – $L(E_1) =\{a^{5x+2} \mid x \geq 0\}$

$a^n=a^9$, $E_2=(a^2)^*$ – $a^9 \notin L(E_2)$ – Not OK – $L(E_2) =\{a^{2x} \mid x \geq 0\}$

$a^n=a^{10}$, $E_3=(a^2+a^3)^*$ – $a^{10} \in L(E_3)$ – OK –

$L(E_3)=\{a^{2x+3y} \mid x \geq 0, y \geq 0\}$

SNP & SNPSP Systems (5/11)

Notes on: **E / a^c**

1) When the regular expression $E = a^c$, the rule criteria "a^c / a^c" can simply be written as "a^c".

2) For **forgetting rule**, the activation criteria are written as "a^c". The regular expression E (of forgetting rules) is restricted and is always "a^c". For any forgetting rule with criteria a^c, the string $a^c \notin L(E')$ where E' a regular expression of any of the spiking rule in the same neuron.

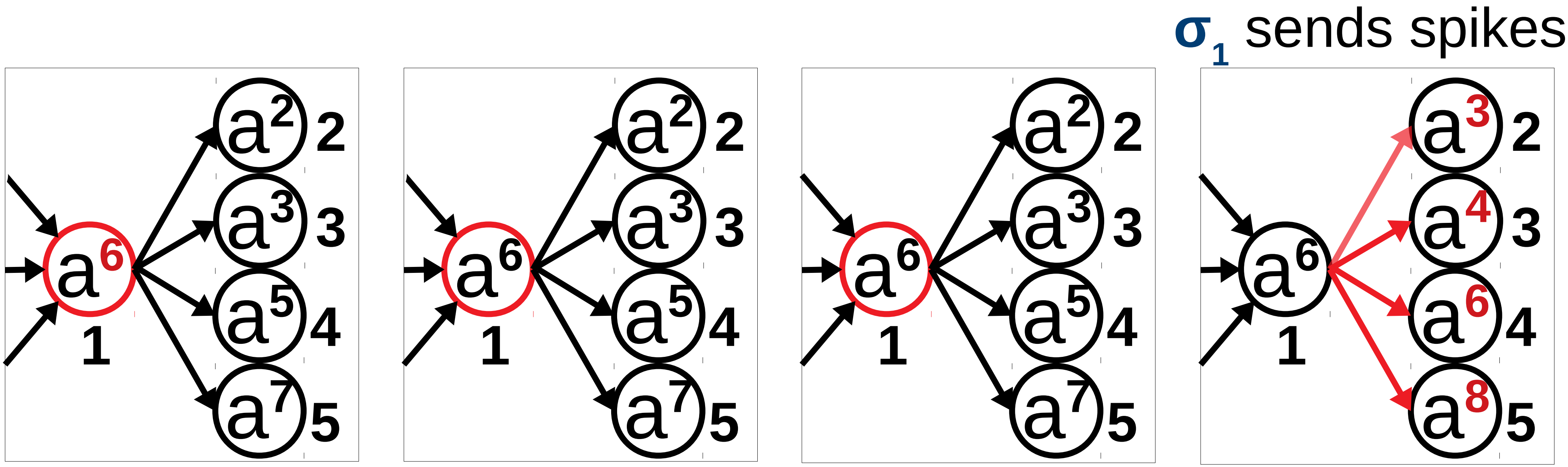
3) If rule r_1 has regular expression E_1 and rule r_2 has regular expression E_2, it is possible that $L(E_1) \cap L(E_2) \neq \varnothing$**.** The languages defined by the regular expressions can intersect. It is possible that <u>multiple rules are applicable</u>. If this is the case, then one rule in non-deterministically selected and appliced/activated.

SNP & SNPSP Systems (6/11)

Spiking Rule: $E / a^c \rightarrow a{:}d$

$d \geq 0$ is known as the delay. When a spiking rule is activated / applied at time t, c spikes are consumed and the neuron containing the rule will be 'closed' for d steps, then will open at time $t+d$ and send a spike to connected neurons.

Example: $r_j: (a^2)^* / a^2 \rightarrow a{:}3,\ r_j \in R_1,\ n_1 = 8$



$t+0$ — σ_1 closes

$t+1$ — σ_1 is closed

$t+2$ — σ_1 is closed

$t+3$ — σ_1 opens

SNP & SNPSP Systems (7/11)

Notes on Spiking Rules: $E / a^c \rightarrow a{:}d$

- When a neuron is ‘closed’, it can not receive spikes from other neurons and can not activate/applied other rules.

- After applying a spiking rule at time **t**, only at time **t+d+1** can another rule be applied.

Forgetting Rule: $a^c \rightarrow \lambda$

When a forgetting rule is applied/activated it will simply consume **c** spikes (all the spikes) from the neuron that contains the rule.

SNP & SNPSP Systems (8/11)

Plasticity Rules: $E / a^c \rightarrow \alpha k(i, N)$

1) i is the label of the neuron containing the rule.

2) $\alpha \in \{+,-,\pm,\mp\}$ **-** action to be performed.

+ add synapses

- delete synapses

± add then (in the next step) delete synapses

∓ delete then (in the next step) add synapses.

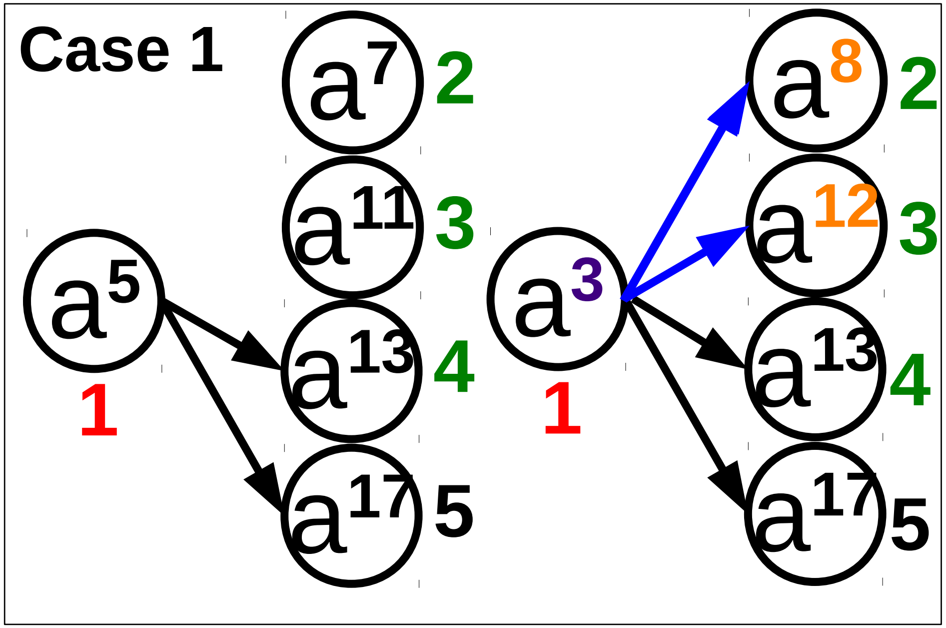
3) $N \subseteq \{1,\ldots,m\}-\{i\}$ - the set of target neurons

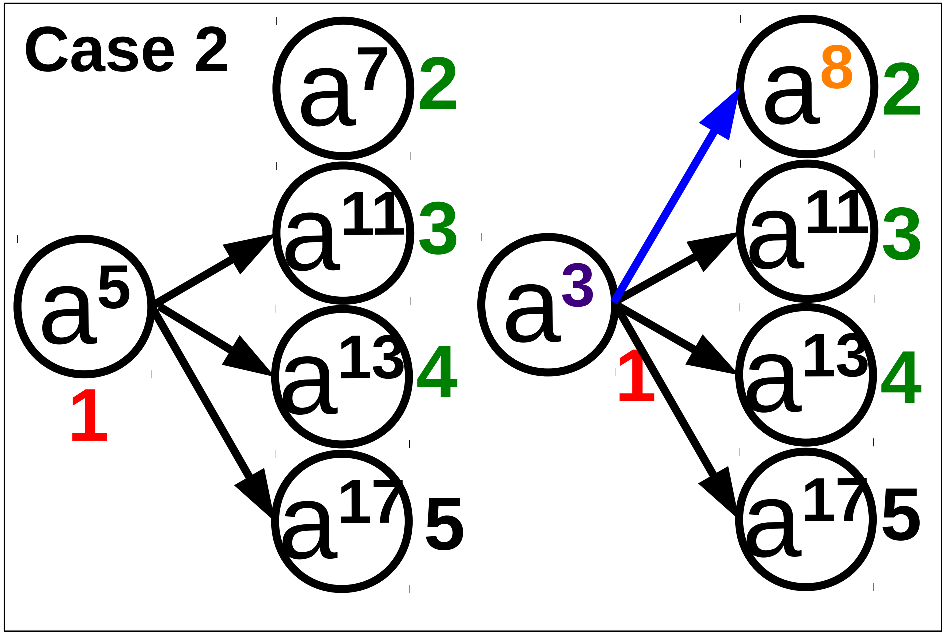
4) $1 \leq k \leq |N|$ **-** number of synapses to be added or deleted.

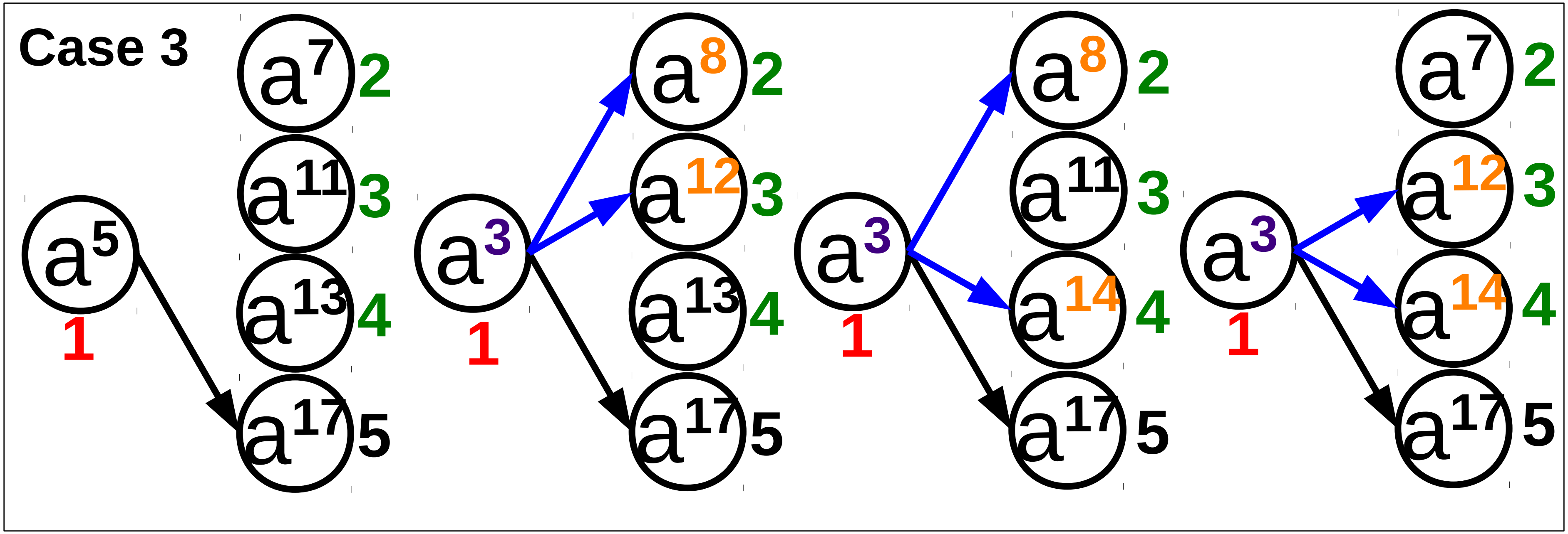
5) When a synapse **(i,j)** is created, it will send one spike to neuron **j**.

SNP & SNPSP Systems (9/11)

Example: r_j: $a(aa)^* / a^2 \rightarrow +2(1, \{2,3,4\})$, $r_j \in R_1$, $n_1 = 5$

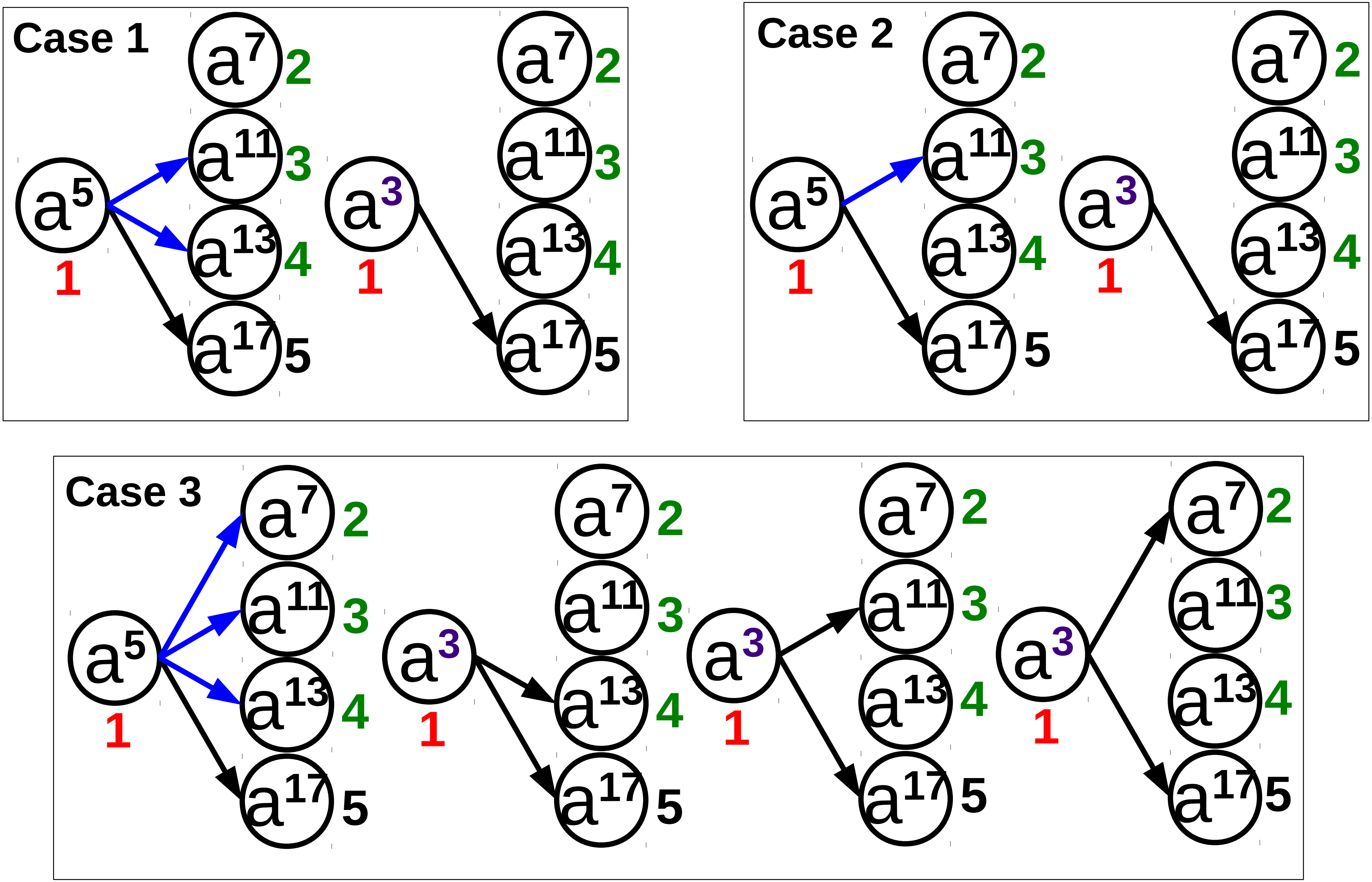






SNP & SNPSP Systems

Example: $r_j: a(aa)^* / a^2 \rightarrow -2(1, \{2,3,4\}),\ r_j \in R_1,\ n_1 = 5$



SNP & SNPSP Systems (11/11)

Semantics:

1) There is a global clock. For every step, every (open) neuron will check if there are any rules that are applicable.

2) If there are applicable rules, the neuron will non-deterministically select and apply a rule.

3) The system will halt if there are no active rules and there are no rules in any neuron that can be activated.

Output of the System (Interpretations):

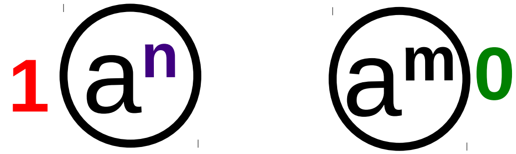
1) Take note of the times when the first two spikes were sent by the output neuron to the environment. If it halts, the time difference is the output (number generated) of the system.

2) If the output neuron sends a spike to the environment then symbol **‘1’** is generated, otherwise **‘0’** is generated. If the system halts, then the string of **0** and **1** symbols is the output of the system.

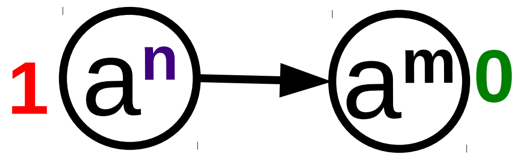
SNP & SNPSP System Features

SNP & SNPSP Features: Forgetting Rule

Example 1: r_j: E / $a^c \rightarrow -1(1, \{0\}), r_j \in R_1$



Example 2: r_j: E / $a^c \rightarrow +1(1, \{0\}), r_j \in R_1$



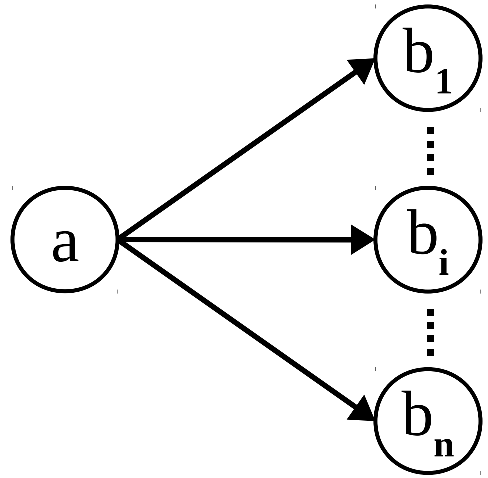
Notes:

1. SNP's forgetting rule $a^c \rightarrow \lambda$ has a restriced regular expression $E=a^c$.

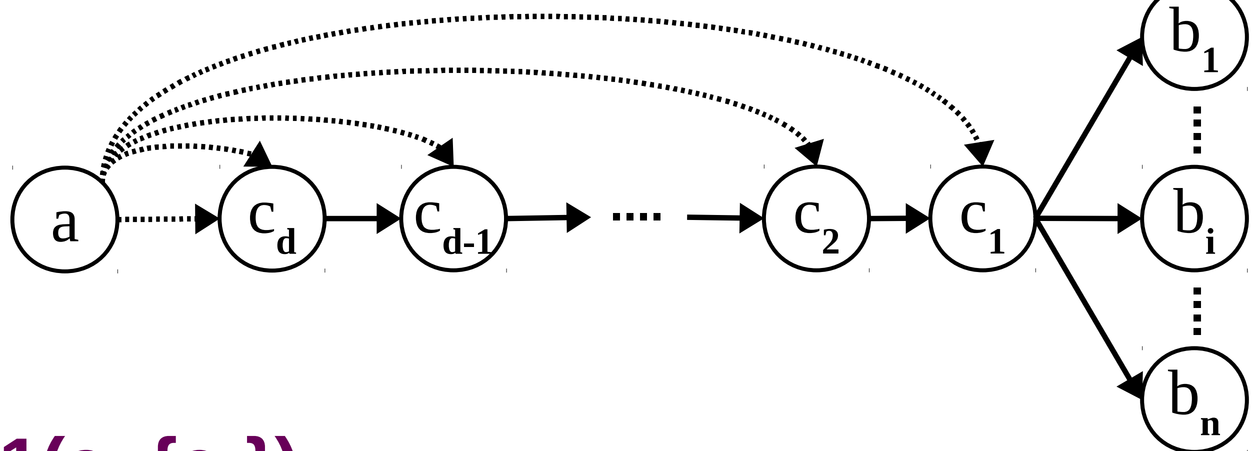
2. SNPSP's "forgetting" rule $E/a^c \rightarrow \lambda$ cis more general that SNP's.

SNP & SNPSP Features: Rules with Delay

SNP spiking rule at neuron **a**: $E / a^c \rightarrow a{:}d$



SNPSP plasticity rule at neuron **a**: $E / a^c \rightarrow \pm 1(a, \{c_1, c_2, \ldots, c_d\})$

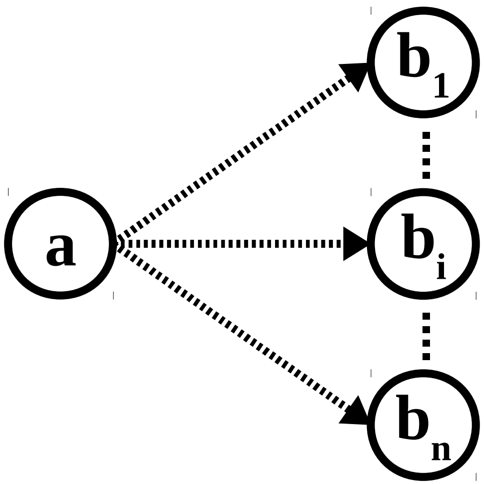


$E / a^c \rightarrow \pm 1(a, \{c_d\})$

SNP & SNPSP Features: Selecting Neurons

Plasticity Rule:

$E / a^{c} \rightarrow +1(a,\{b_1,\ldots,b_n\})$



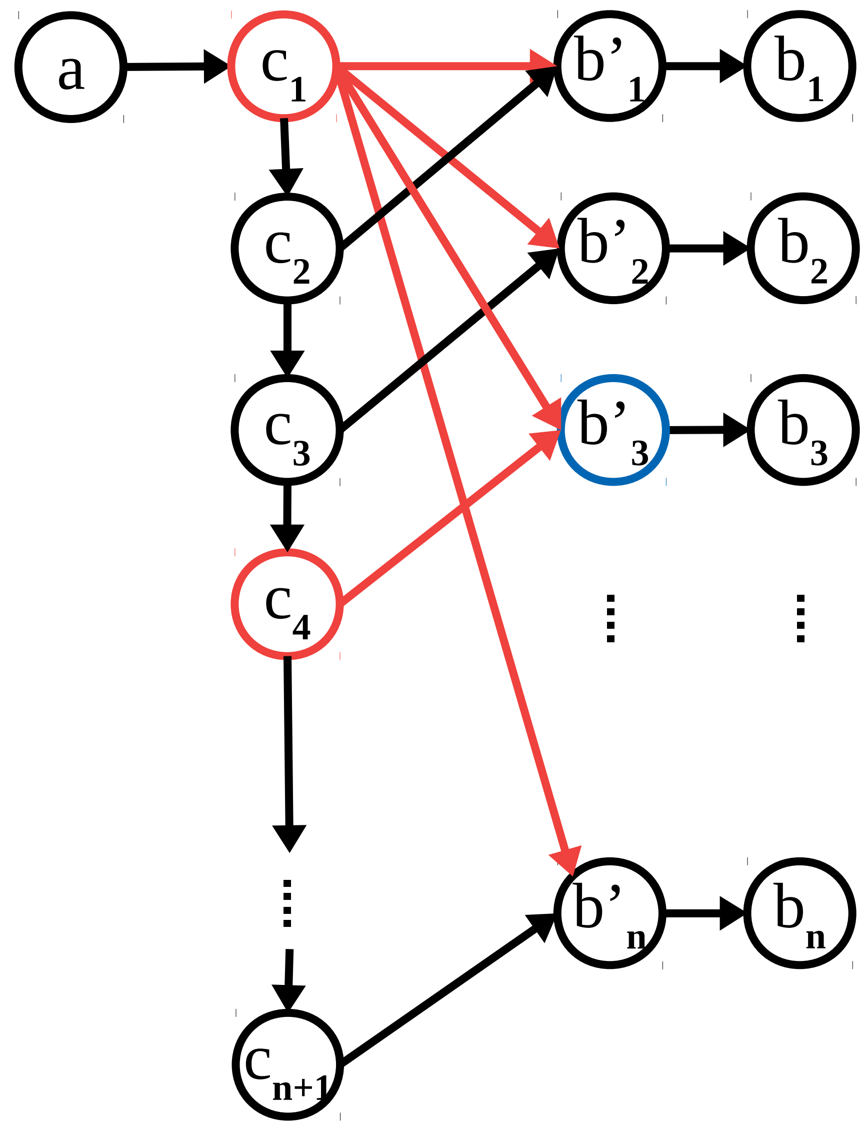
SNP: r_1:$E_1/a^{c1} \rightarrow a{:}0$, r_2:$E_2/a^{c2} \rightarrow a{:}2$

t+0: r_1 ('1')

t+1: r_2 ('0')

t+2: r_2 ('0')

t+3: r_2 ('1')



SNPSP Systems & FIN Languages 1

SNPSP Systems & FIN Languages 1

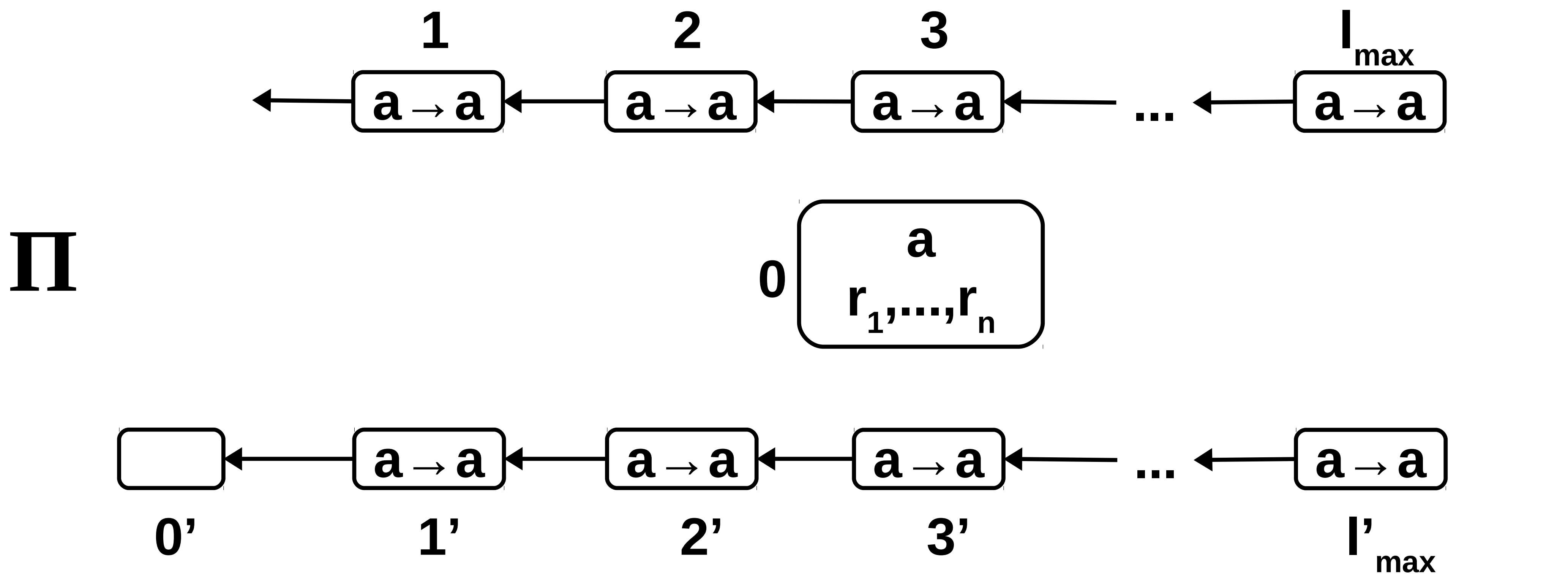
Theorem: If $L \in$ **FIN** and $L \subseteq \{0,1\}^+$, then the SNPSP system Π generates words in L' where $L' = \{0\}L$.

$L = \{b_1, b_2, b_3, \ldots, b_n\}$

$L' = \{0\}L = \{0b_1, 0b_2, 0b_3, \ldots, 0b_n\}$

$l_{max} = \max\{|b_i|\}$

l_{max} is the length of the longest word in L.



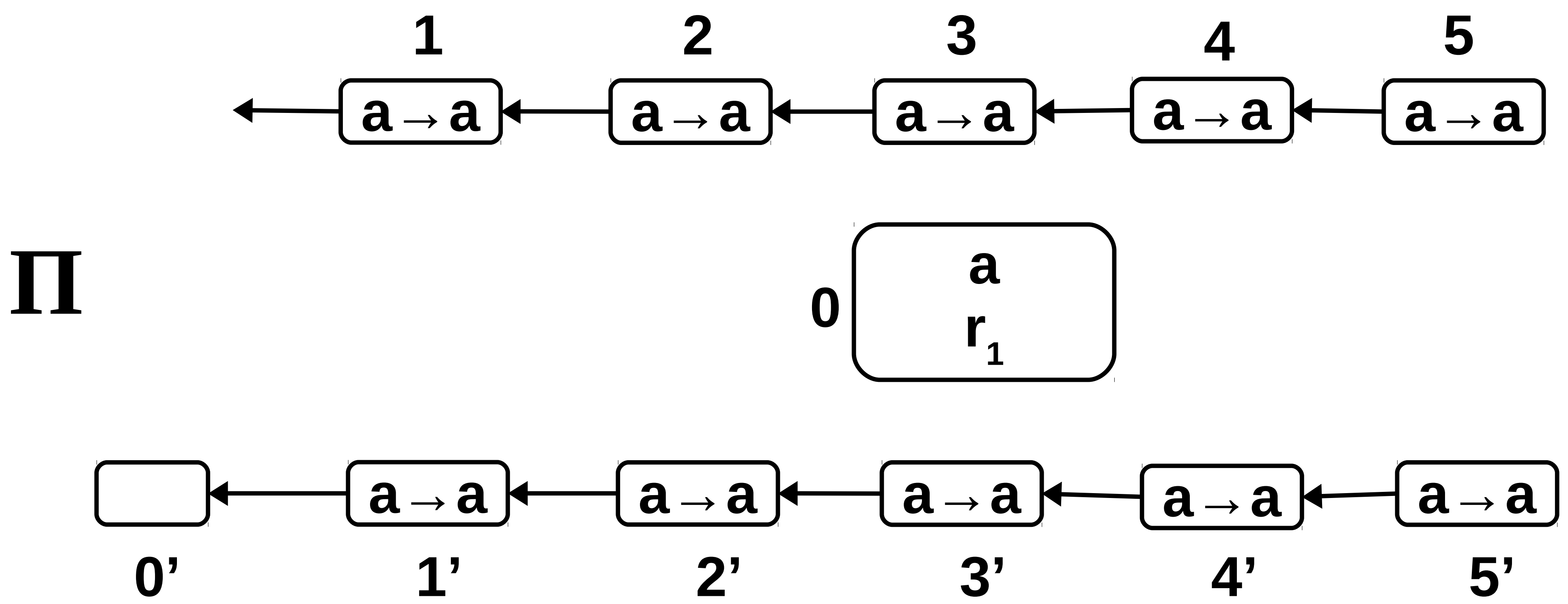
SNPSP Systems & FIN Languages 1

For each word $\mathbf{b_i} \in \mathbf{L}$, there will be plasticity rule $\mathbf{r_i}$ in neuron **0** with form $\mathbf{r_i: a \rightarrow +k_i(0, N_i \cup \{|b_i|'\})}$ where

$\mathbf{N_i} = \{\mathbf{p} \mid \mathbf{p}^{th}$ symbol of $\mathbf{b_i}$ is '**1**'$\}$ and $\mathbf{k_i = |b_i|_1 + 1}$.

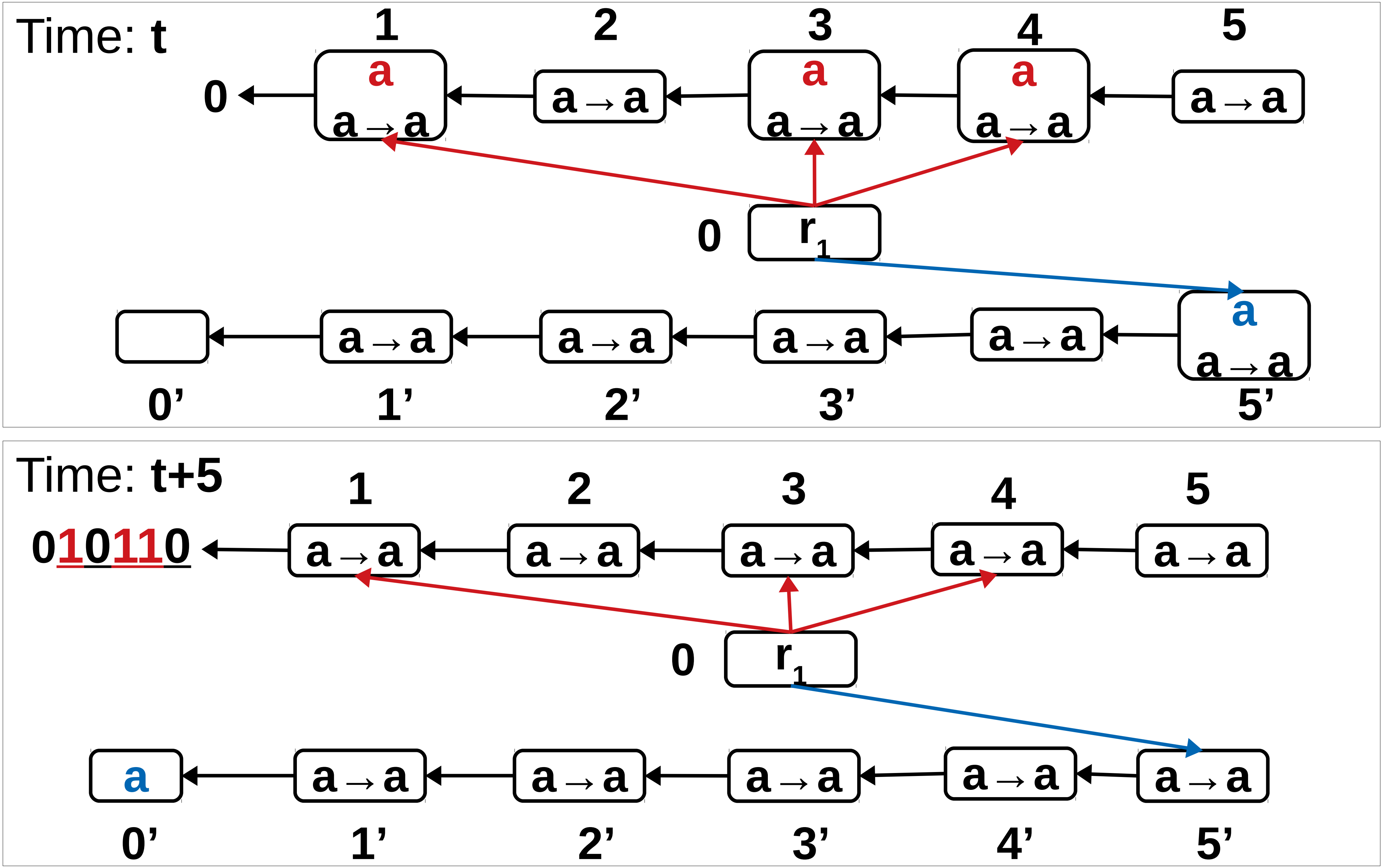
Example: $\mathbf{L = \{b_1 = 10110\}}$, $\mathbf{N_1 = \{1,3,4\}}$, $\mathbf{k_1 = 3+1 = 4}$,

$\mathbf{r_1: a \rightarrow +4(0, \{1,3,4\} \cup \{5'\})}$

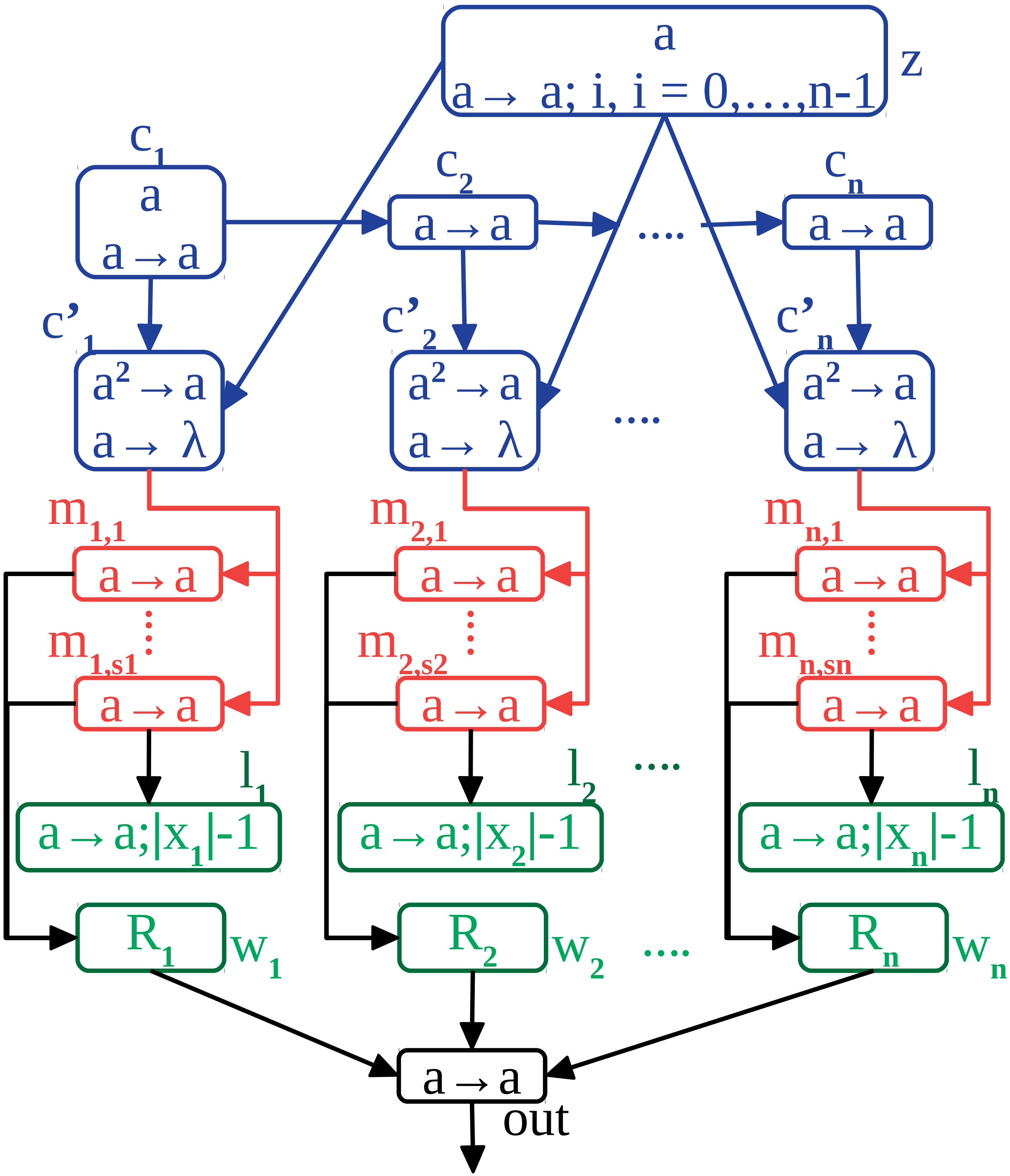


SNPSP Systems & FIN Languages 1

Example: $L=\{b_1=10110\}$, r_1: $a \rightarrow +4(0, \{1,3,4\} \cup \{5'\})$



SNP System for FIN Languages



$x = 0^{d1}1 \mid 0^{d2}1 \mid \ldots \mid 0^{di}1 \mid 0^{di+1}$

Neuron W:

r_1: $E_1/a \rightarrow a{:}d_1$ - $0^{d1}1$

r_2: $E_2/a \rightarrow a{:}d_2$ - $0^{d2}1$

...

r_i: $E_i/a \rightarrow a{:}d_i$ - $0^{di}1$

Neuron I_n:

r_i: $a \rightarrow a{:}|x|-1$ - 0^{di+1}

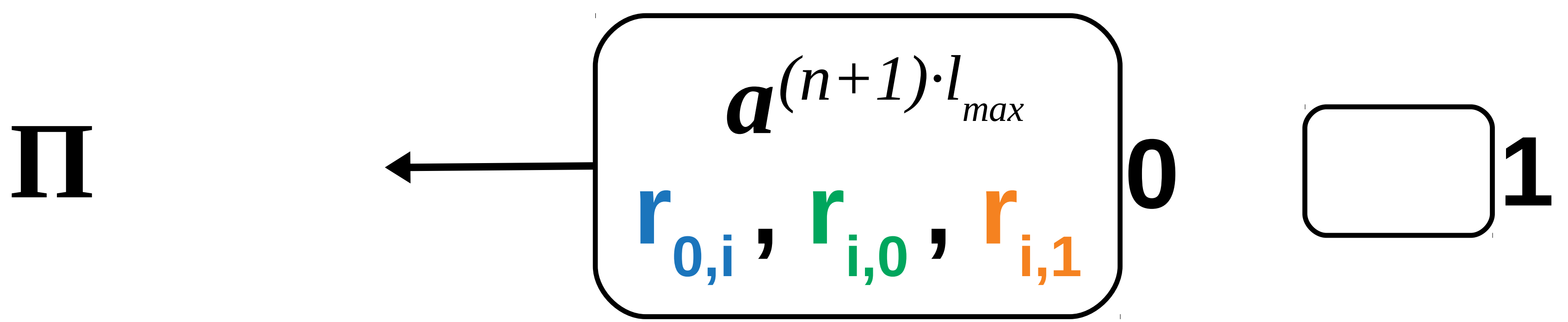
SNPSP Systems & FIN Languages 2

SNPSP Systems & FIN Languages 2

Theorem: If $L \in \mathbf{FIN}$ and $L \subseteq \{0,1\}^+$, then the SNPSP system Π that generates words in L.

Let $L = \{b_1, b_2, b_3, \ldots, b_n\}$.

Let $l_{max} = \max\{|b_i|\}$ is the length of the longest word in L.



For each b_i, the 3 rules $r_{0,i}$, $r_{i,0}$, $r_{i,1}$ are added in σ_0.

$r_{0,i}$ - generates the <u>**initial**</u> symbol of b_i

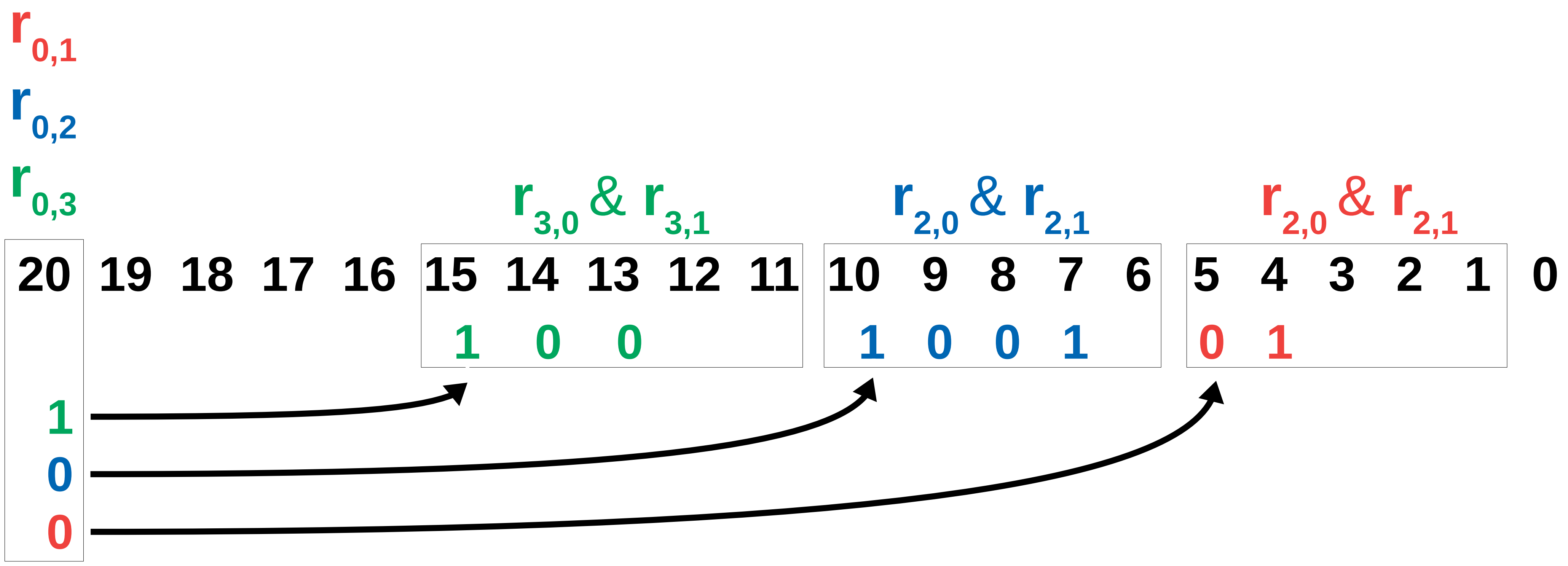
$r_{i,0}$ - generates the **‘0’** symbols of b_i

$r_{i,1}$ - generates the **‘1’** symbol of b_i

SNPSP Systems & FIN Languages 2

Example: $L = \{b_1 = 101,\ b_2 = 01001,\ b_3 = 1100\}$. $l_{max} = 5$.

$(n+1) \cdot l_{max} = (3+1)5 = 20$.



SNPSP Systems & FIN Languages 2

Example: $L = \{b_1 = 101,\ b_2 = 01001,\ b_3 = 1100\}$. $l_{max} = 5$.

$(n+1) \cdot l_{max} = (3+1)5 = 20.$

$r_{0,1}: a^{20}/a^{15} \to \lambda$

$r_{0,2}: a^{20}/a^{10} \to \lambda$

$r_{0,3}: a^{20}/a^{10} \to a$

$r_{3,0}: a^{13}+a^{14} \to \lambda$

$r_{3,1}: a^{15} \to a$

$r_{2,0}: a^{8}+a^{9} \to \lambda$

$r_{2,1}: a^{10}+a^{7} \to a$

$r_{1,0}: a^{5} \to \lambda$

$r_{1,1}: a^{4} \to a$

20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
					1	0	0			1	0	0	1		0	1				

1
0
0

SNPSP Systems & FIN Languages 2

For each b_i, the 3 rules $r_{0,i}$, $r_{i,0}$, $r_{i,1}$ are added in σ_0.

Rule Form: $r_{0,i}$: $a^{(n+1)\cdot l_{max}} / a^{(n+1-i)\cdot l_{max}} \rightarrow x$

(Spiking) $x = a$ if the first symbol of b_i is '1'.

(Forgetting) $x = \lambda$ if the first symbol of b_i is '0'.

Rule Form: $r_{i,0}$: $E_i / a \rightarrow \lambda$

$$E_i = \sum_{q_j \in Q'_i} a^{i\cdot l_{max} - (q_j - 2)}$$

$Q_i = \{q \mid q^{th}$ symbol of b_i is '0'$\}$

$Q'_i = Q_i - \{1\}$

Rule Form: $r_{i,1}$: $E_i / a \rightarrow a$

$$E_i = \sum_{p_j \in P'_i} a^{i\cdot l_{max} - (p_j - 2)}$$

$P_i = \{p \mid p^{th}$ symbol of b_i is '1'$\}$

$P'_i = P_i - \{1\}$

SNPSP Systems & REG Languages 1

SNPSP Systems & REG Languages 1

Right-Linear Grammar: $G=(N,B,S,P)$

$N=\{N_1,N_2,\ldots,N_m\}$ (non-terminal symbols)

$B=\{0,1\}$ (terminal symbols)

$S=N_m$ (start symbol)

P is the set of k production rules of the forms:

R_i: $N_{pi} \rightarrow bN_{qi}$, $b \in B$, $1 \leq p_i,q_i \leq m$, $1 \leq i \leq k$

R_i: $N_{pi} \rightarrow b$, $b \in B$, $1 \leq p_i \leq m$, $1 \leq i \leq k$

SNPSP Systems & REG Languages 1

Right-Linear Grammar: $G=(N,B,S,P)$

$N=\{N_1,N_2\}$ (non-terminal symbols)

$B=\{0,1\}$ (terminal symbols)

$S=N_2$ (start symbol)

P is the set of **3** production rules:

R_1: $N_2 \rightarrow 1N_1$

R_2: $N_1 \rightarrow 0N_1$

R_3: $N_1 \rightarrow 1$

String Derivation Example:

$$N_2 \xrightarrow[R_1]{} 1N_1 \xrightarrow[R_2]{} 10N_1 \xrightarrow[R_2]{} 100N_1 \xrightarrow[R_3]{} 1001$$

SNPSP Systems & REG Languages 1

Theorem: If $L \subseteq B^+ = \{0,1\}^+$ and $L \in$ **REG** , then there is an SNPSP system Π and a morphism $h: B^* \to B^*$ such that

$L = h^{-1}(L(\Pi))$.

If $x \in L$: $x = s_1 s_2 s_3 \ldots s_n$

then $x' \in L(\Pi)$: $x' = 0s_1s_1 0s_2s_2 0s_3s_3 \ldots 0s_ns_n$

$s_i \in B$ $(1 \leq i \leq n)$

Morphism: $h(0)=000$, $h(1)=011$

Example: If $x=10100$, then $x'=h(10100)$

$x'=h(1)h(0)h(1)h(0)h(0)$

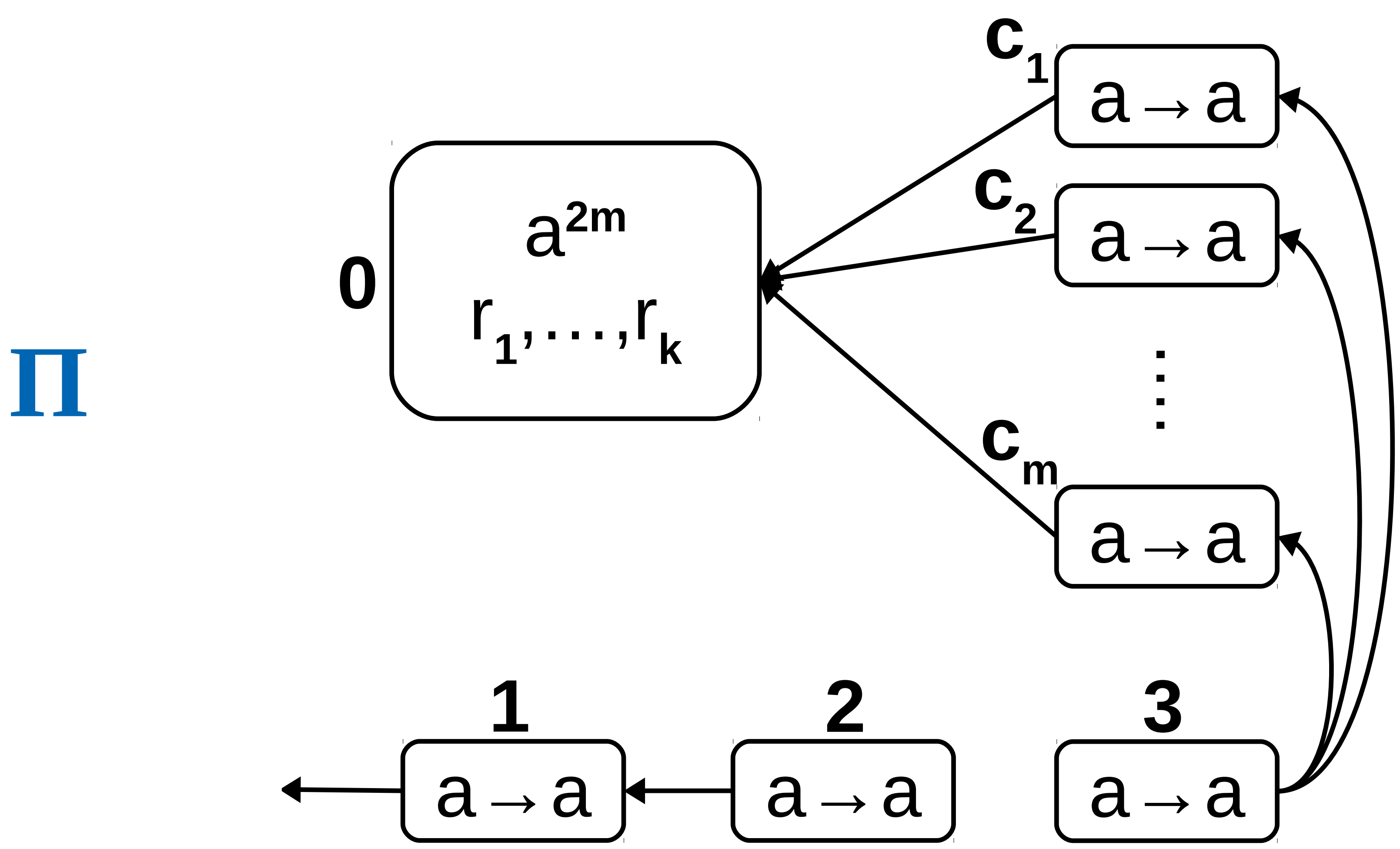
$x'=011\ 000\ 011\ 000\ 000$

$x'=011000011000000$

SNPSP Systems & REG Languages 1

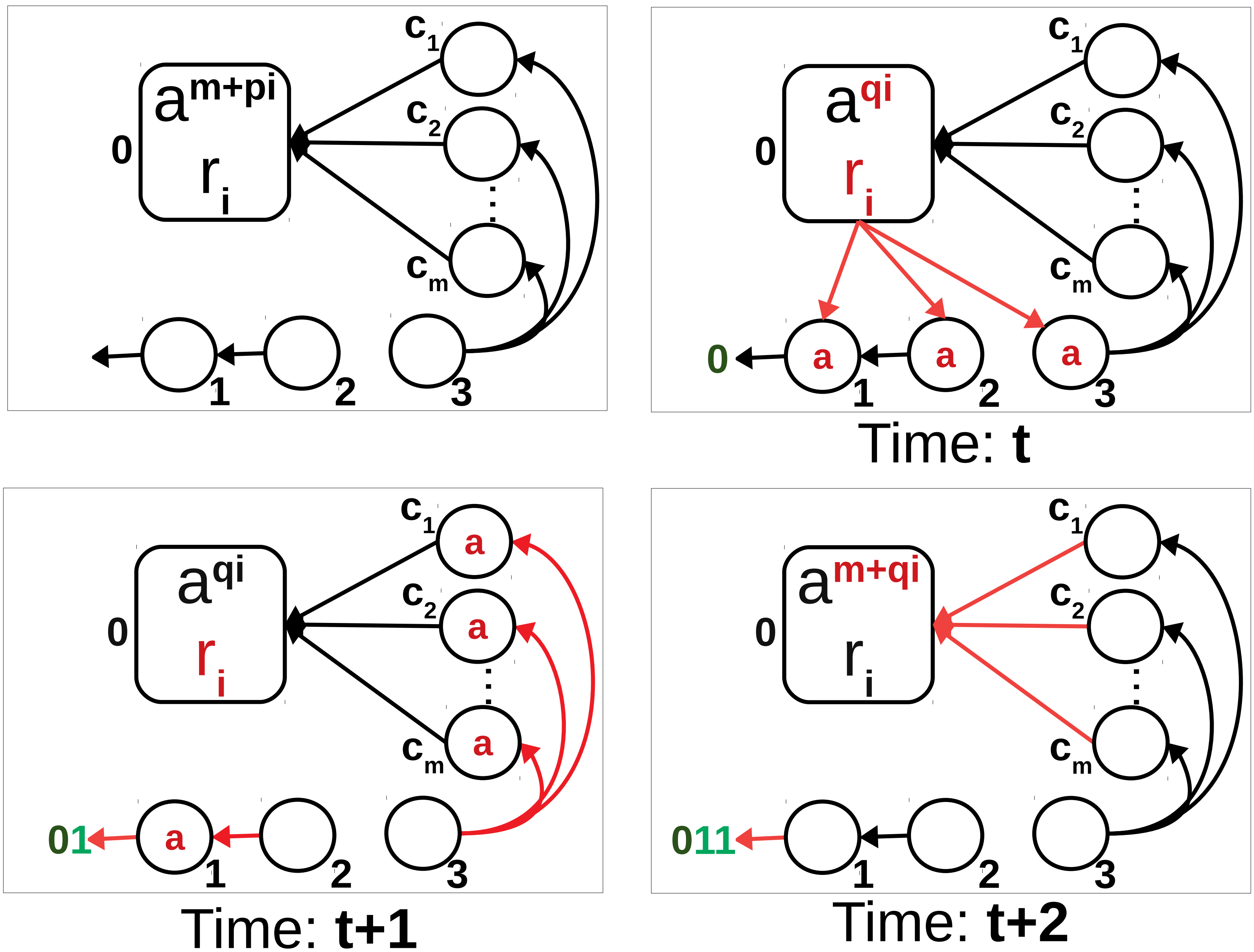
Theorem: If $\mathbf{L} \subseteq \mathbf{B}^{+}=\{0,1\}^{+}$ and $\mathbf{L} \in$ **REG** , then there is an SNPSP system Π and a morphism $\mathbf{h}: \mathbf{B}^{*} \to \mathbf{B}^{*}$ such that

$\mathbf{L} = \mathbf{h}^{-1}(L(\Pi))$.



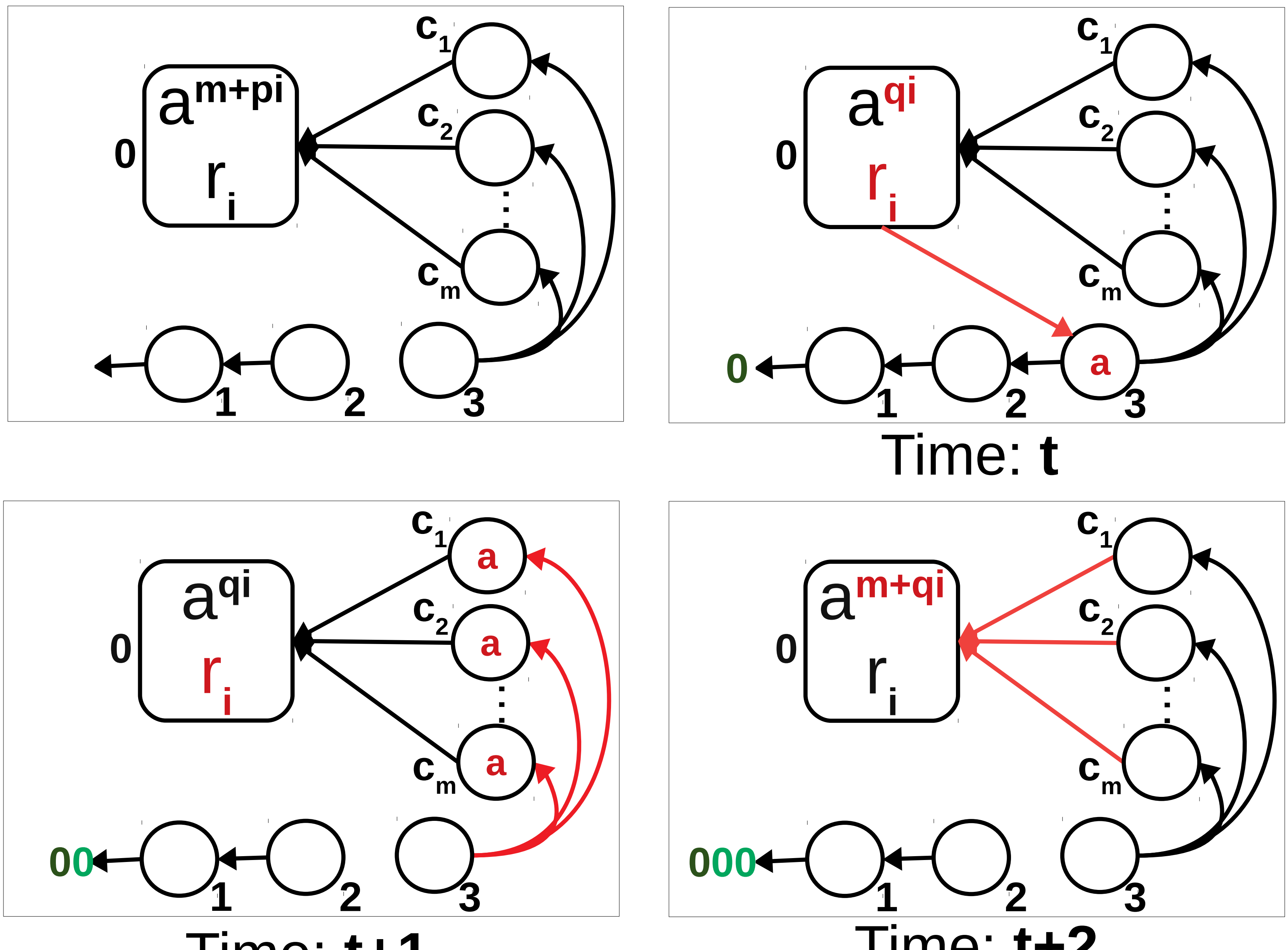
SNPSP Systems & REG Languages 1

Production Rule: R_i: $N_{pi} \rightarrow 1N_{qi}$



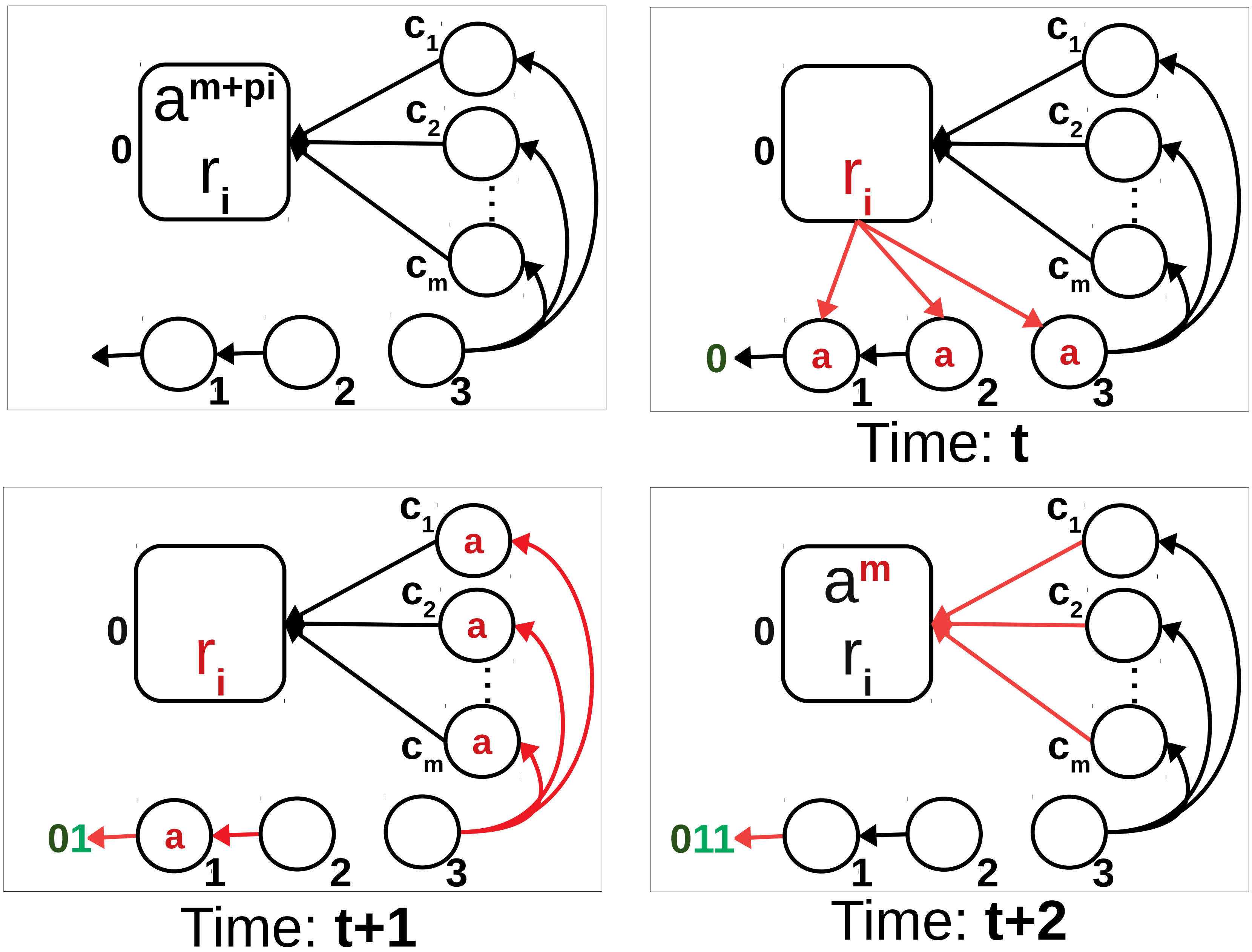
SNPSP Systems & REG Languages 1

Production Rule: $R_i: N_{pi} \rightarrow 0N_{qi}$



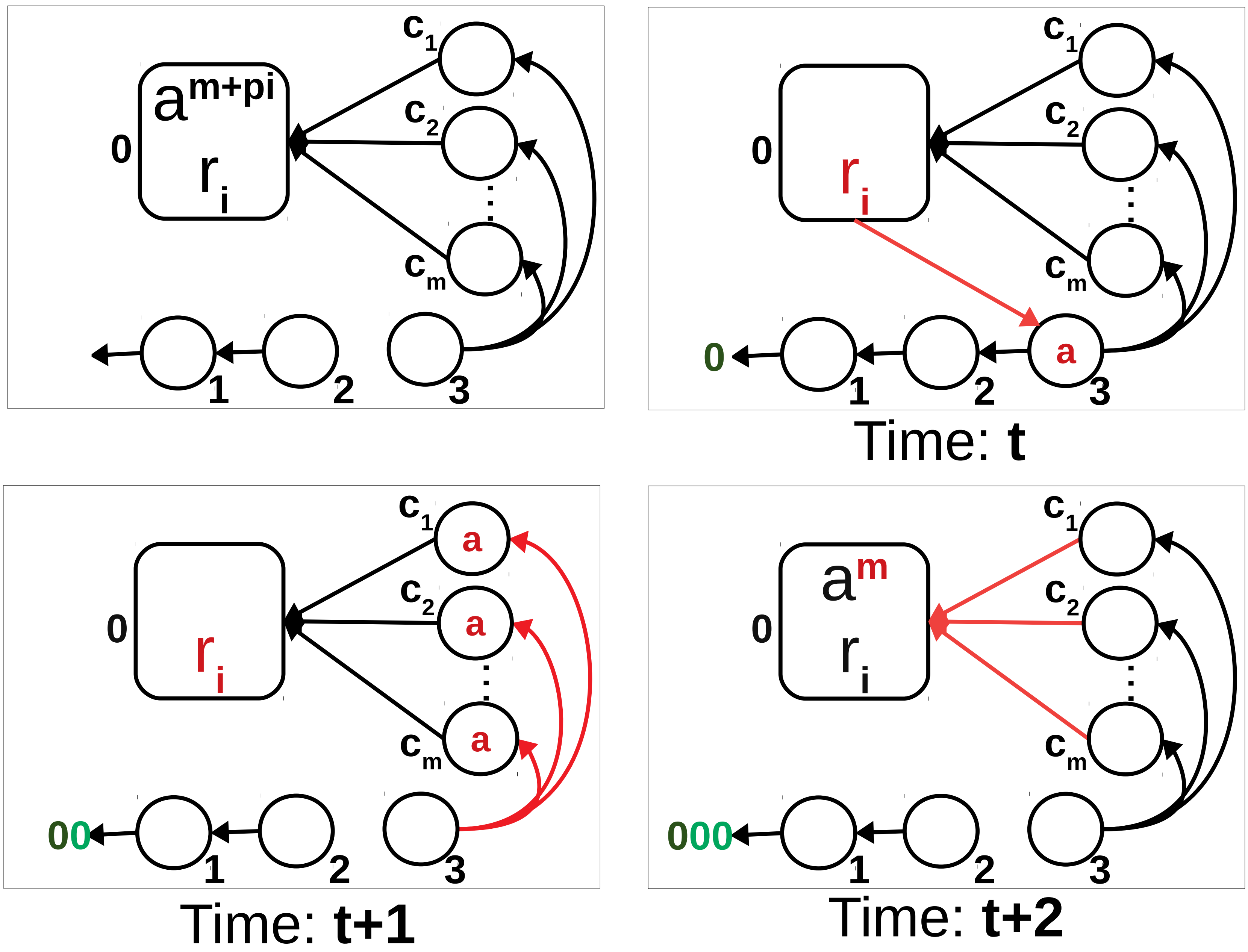
SNPSP Systems & REG Languages 1

Production Rule: R_i: $N_{pi} \rightarrow 1$



SNPSP Systems & REG Languages 1

Production Rule: R_i: $N_{pi} \rightarrow 0$



SNPSP Systems & REG Languages 1

Non-terminal Production rule: $\mathbf{R_i: N_{pi} \rightarrow bN_{qi}}$

Corresponding <u>plasticity rule</u> $\mathbf{r_i}$ in $\boldsymbol{\sigma_i}$:

If **b=1**, $\mathbf{r_i: a^{(m+pi)} / a^{(m+pi-qi)} \rightarrow \pm 3(0,\{1,2,3\})}$

If **b=0**, $\mathbf{r_i: a^{(m+pi)} / a^{(m+pi-qi)} \rightarrow \pm 1(0,\{3\})}$

Requires: $\mathbf{(m+p_i)}$ spikes. Consumes: $\mathbf{(m+p_i-q_i)}$ spikes.

Terminal Production rule: $\mathbf{R_i: N_{pi} \rightarrow b}$

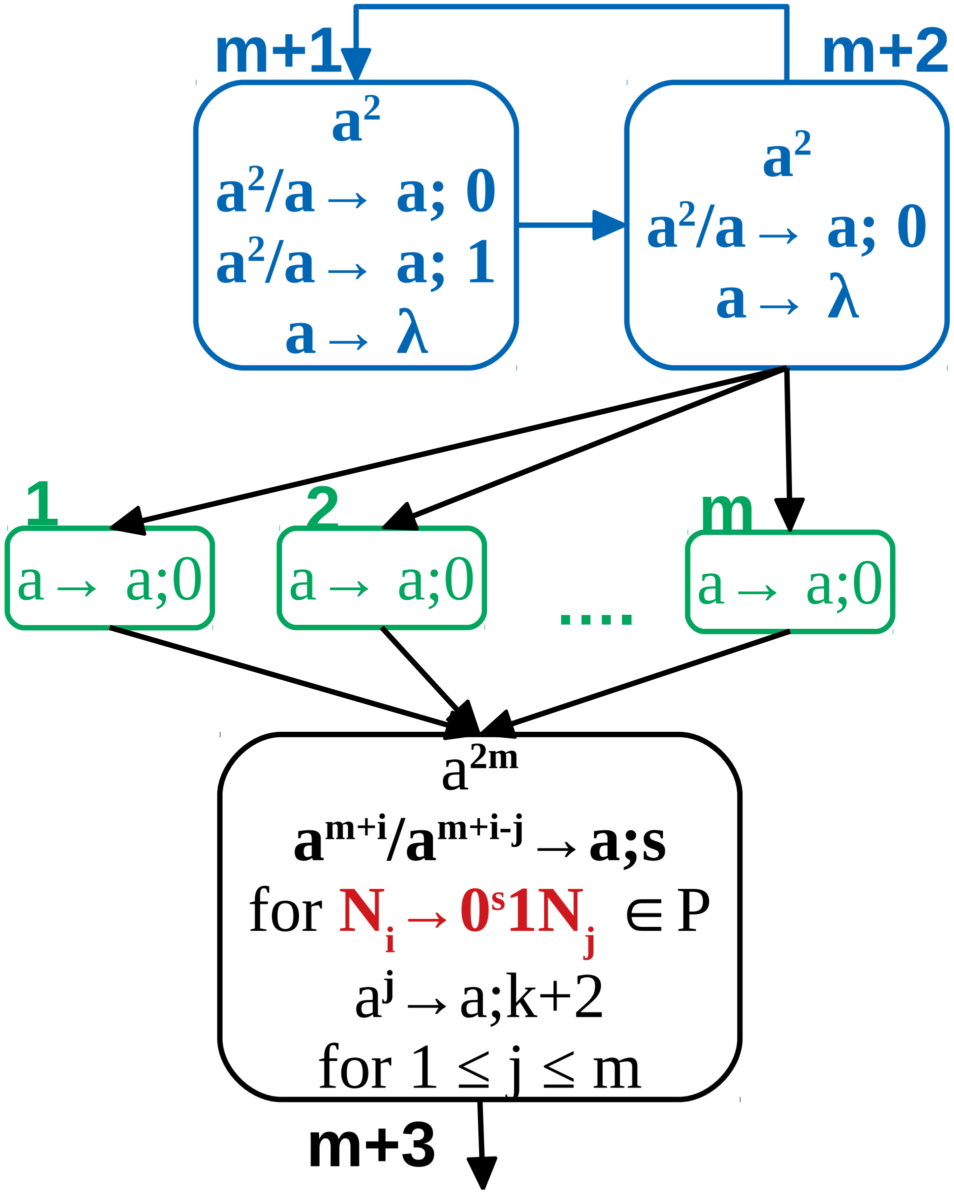
Corresponding <u>plasticity rule</u> $\mathbf{r_i}$ in $\boldsymbol{\sigma_i}$:

If **b=1**, $\mathbf{r_i: a^{(m+pi)} / a^{(m+pi)} \rightarrow \pm 3(0,\{1,2,3\})}$

If **b=0**, $\mathbf{r_i: a^{(m+pi)} / a^{(m+pi)} \rightarrow \pm 1(0,\{3\})}$

Requires: $\mathbf{(m+p_i)}$ spikes. Consumes: $\mathbf{(m+p_i)}$ spikes.

SNP System for REG Languages



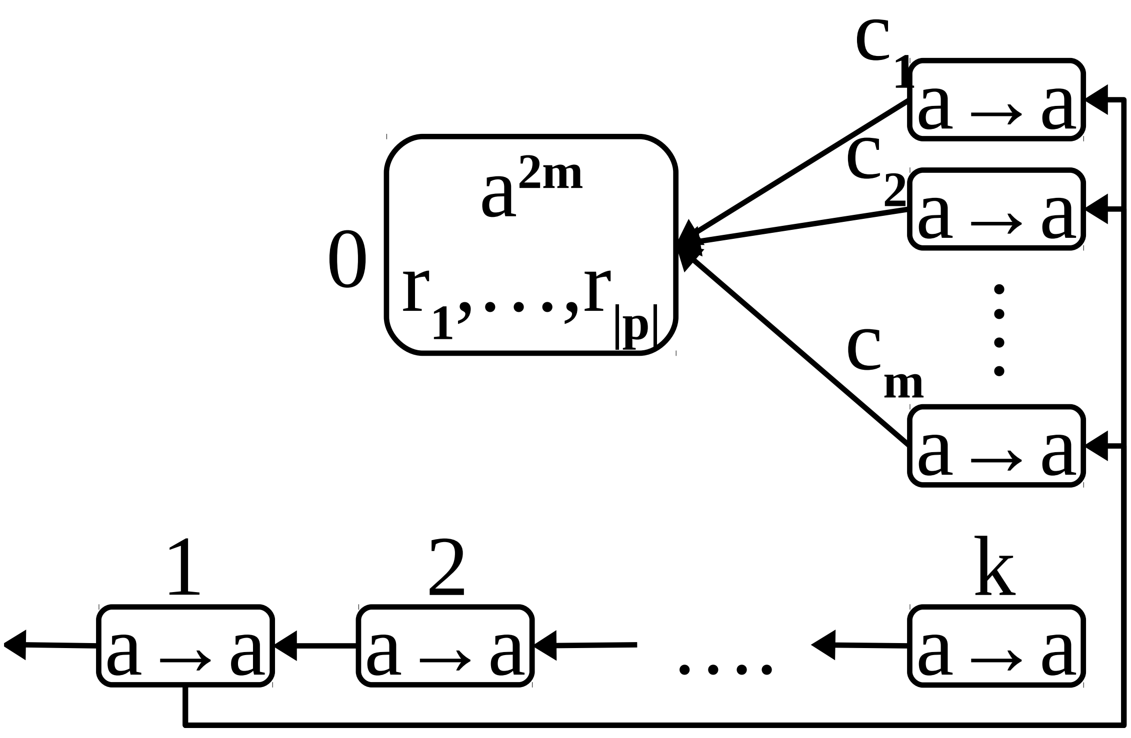
$V=\{a_1,\ldots,a_k\}$

a_s - 0^s1

$N_i \to 0^s1N_j$

$a^{m+i}/a^{m+i-j} \to a;s$

SNPSP Systems & REG Languages 2



SNPSP Systems & RE Languages

SNPSP Systems & RE Languages 1: Encoding

Encoding Strings: $val_k(x)$

$V=\{a_1,a_2,a_3,\ldots, a_k\}$ – Alphabet

$x = a_{i1}a_{i2}a_{i3}\ldots a_{im}$ – string over over V.

Symbol Encoding: $val_k(a_i)=i$ of symbol a_i.

$val_k(a_1) = 1$

$val_k(a_2) = 2$

....

$val_k(a_k) = k$

SNPSP Systems & RE Languages 1: Encoding

Example: $V=\{a_1=p,\ a_2=q,\ a_3=r,\ a_4=s\}$, $k+1 = 4+1 = 5$

$x = sqqrp$

$val_4(s) = 4_5$

$val_4(sq) = 10_5 \cdot val_4(s) + val_4(q) = \boxed{40_5 + 2_5} = \boxed{42_5}$

$val_4(sqq) = 10_5 \cdot val_4(sq) + val_4(q) = \boxed{420_5 + 2_5} = \boxed{422_5}$

$val_4(sqqr) = 10_5 \cdot val_4(sqq) + val_4(r) = \boxed{4220_5 + 3_5} = \boxed{4223_5}$

$val_4(sqqrp) = 10_5 \cdot val_4(sqqr) + val_4(p) = \boxed{42230_5 + 1_5} = \boxed{42231_5}$

SNPSP Systems & RE Languages 1

Theorem: For every alphabet $V=\{a_1,a_2,a_3,\ldots, a_k\}$, there is a morphism $h_1:(V\cup\{b,c\})^* \rightarrow B^*$ and a projection $h_2:(V\cup\{b,c\})^* \rightarrow V^*$ such that for each language $L \subseteq V^*$, $L \in RE$, there is an SNPSP system Π such that $L = h_2(h_1^{-1}(L(\Pi)))$

$x \in L$, $y \in L(\Pi)$

$x = a_{i1}a_{i2}a_{i3}\ldots a_{im}$

$y = 10^{i1}1 \mid 0^{j1}1 \mid 10^{i2}1 \mid 0^{j2}1 \mid \ldots \mid 10^{im}1 \mid 0^{(jm+s)}1.$

We define h_1 and h_2 as:

$h_1(a_i) = 10^i1$ for $1 \le i \le k$, $h_1(b)=0$, $h_1(c)=01$

$h_2(a_i) = a_i$ for $1 \le i \le k$, $h_2(b)=\lambda$, $h_2(c) = \lambda$

SNPSP Systems & RE Languages 1

h_1:

$h_1(a_i) = 10^i1$

$h_1(b) = 0$

$h_1(c) = 01$

h_2:

$h_2(a_i) = a_i$

$h_2(b) = \lambda$

$h_2(c) = \lambda$

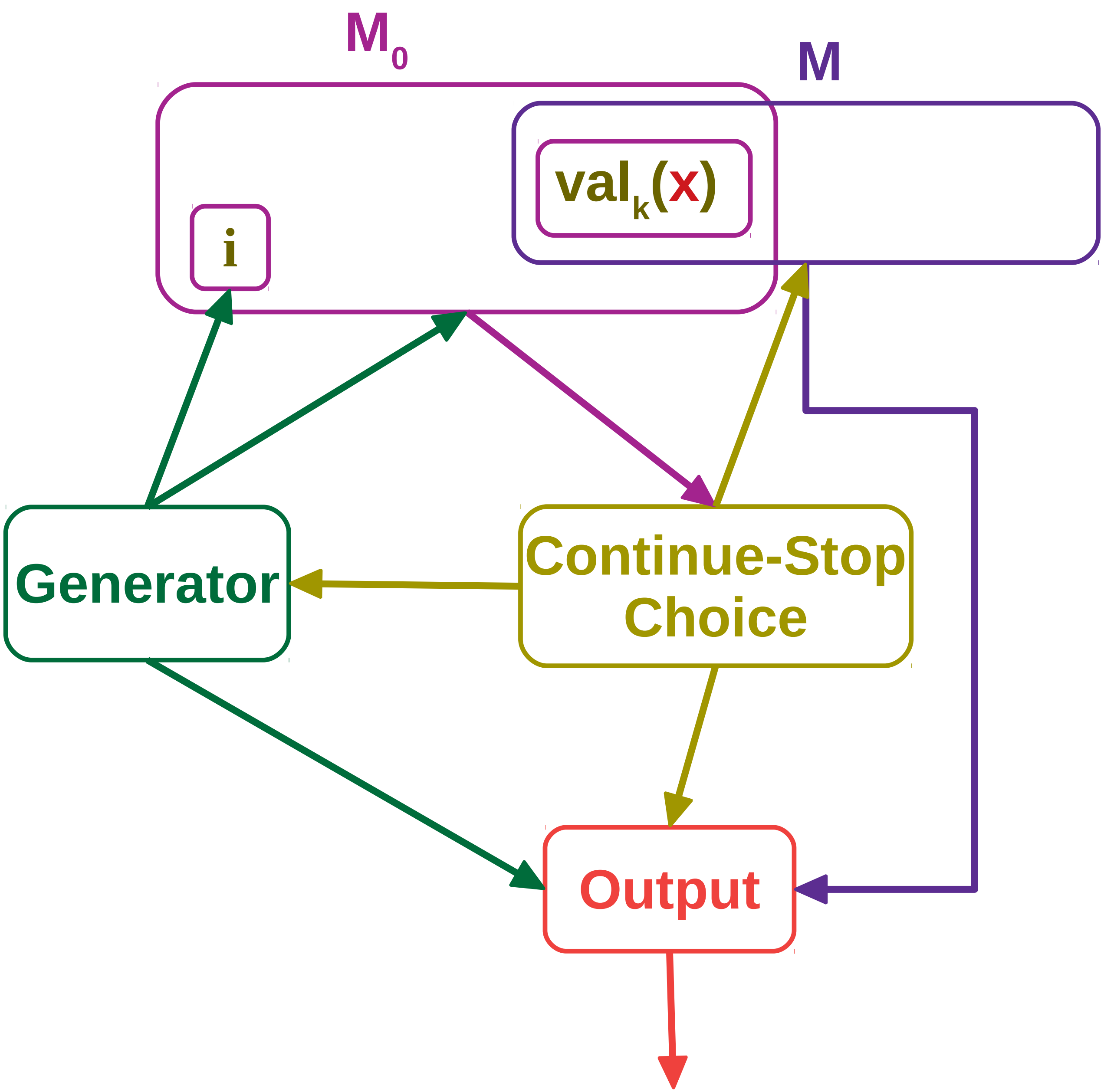
$$y = 10^{i1}1 \mid 0^{j1}1 \mid 10^{i2}1 \mid 0^{j2}1 \mid \ldots \mid 10^{im}1 \mid 0^{(jm+s)}1.$$

$$y' = h_1^{-1}(y) = a_{i1} \mid b^{j1-1}c \mid a_{i2} \mid b^{j2-1}c \mid \ldots \mid a_{im} \mid b^{jm+s-1}c$$

$$x = h_2(y') = a_{i1} \mid \lambda \mid a_{i2} \mid \lambda \mid \ldots \mid a_{im} \mid \lambda$$

$$x = h_2(h_1^{-1}(y)) = a_{i1}a_{i2}a_{i3}\ldots a_{im}$$

SNPSP Systems & RE Languages 1: System



$x = a_{i1}a_{i2}a_{i3}\ldots\mathbf{a_{it}}\ldots a_{im}$

Symbol (for x):

$\mathbf{a_{it}}$

Symbol Encoding:

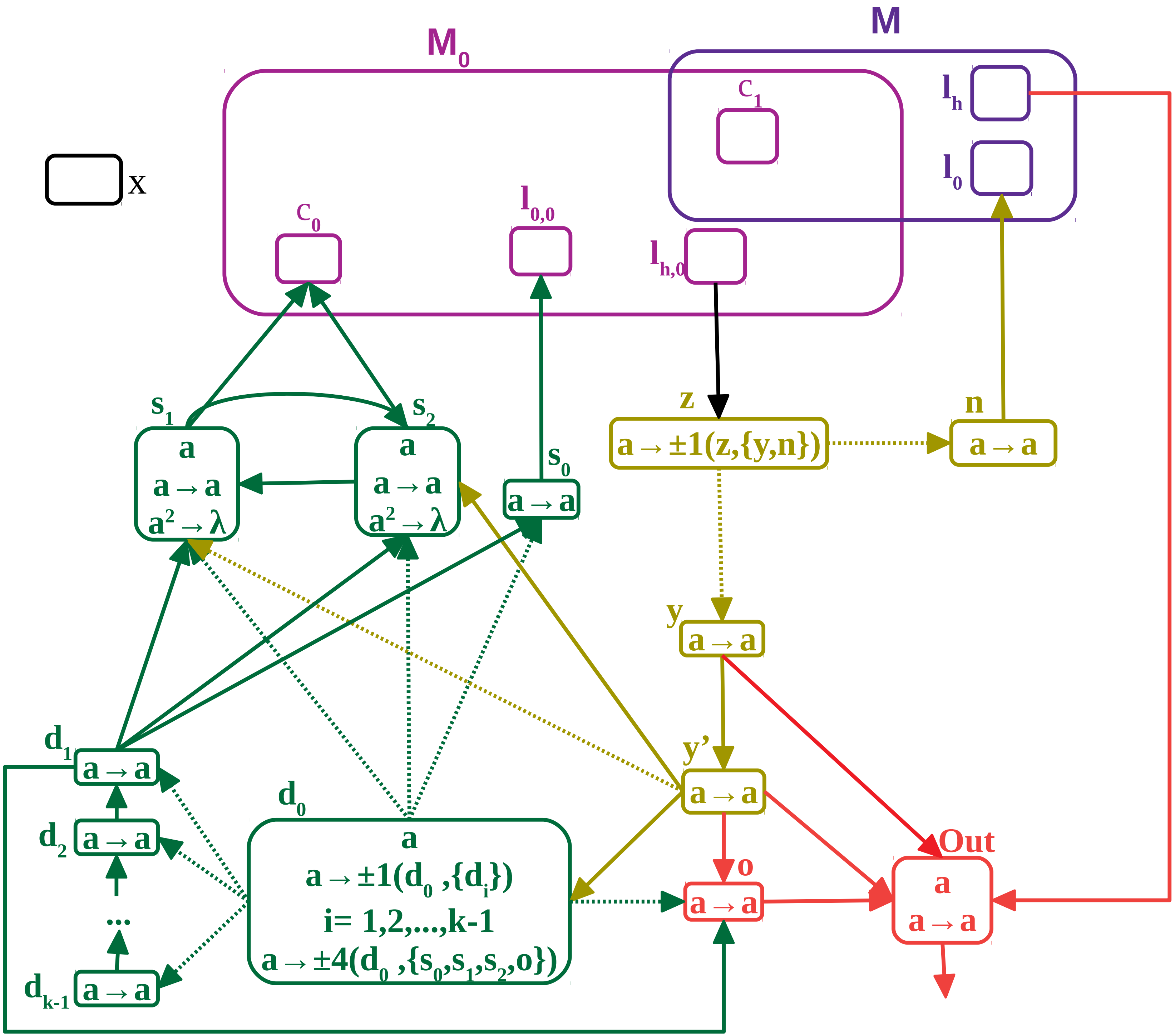
$\mathbf{i_t}$

Substring (for y):

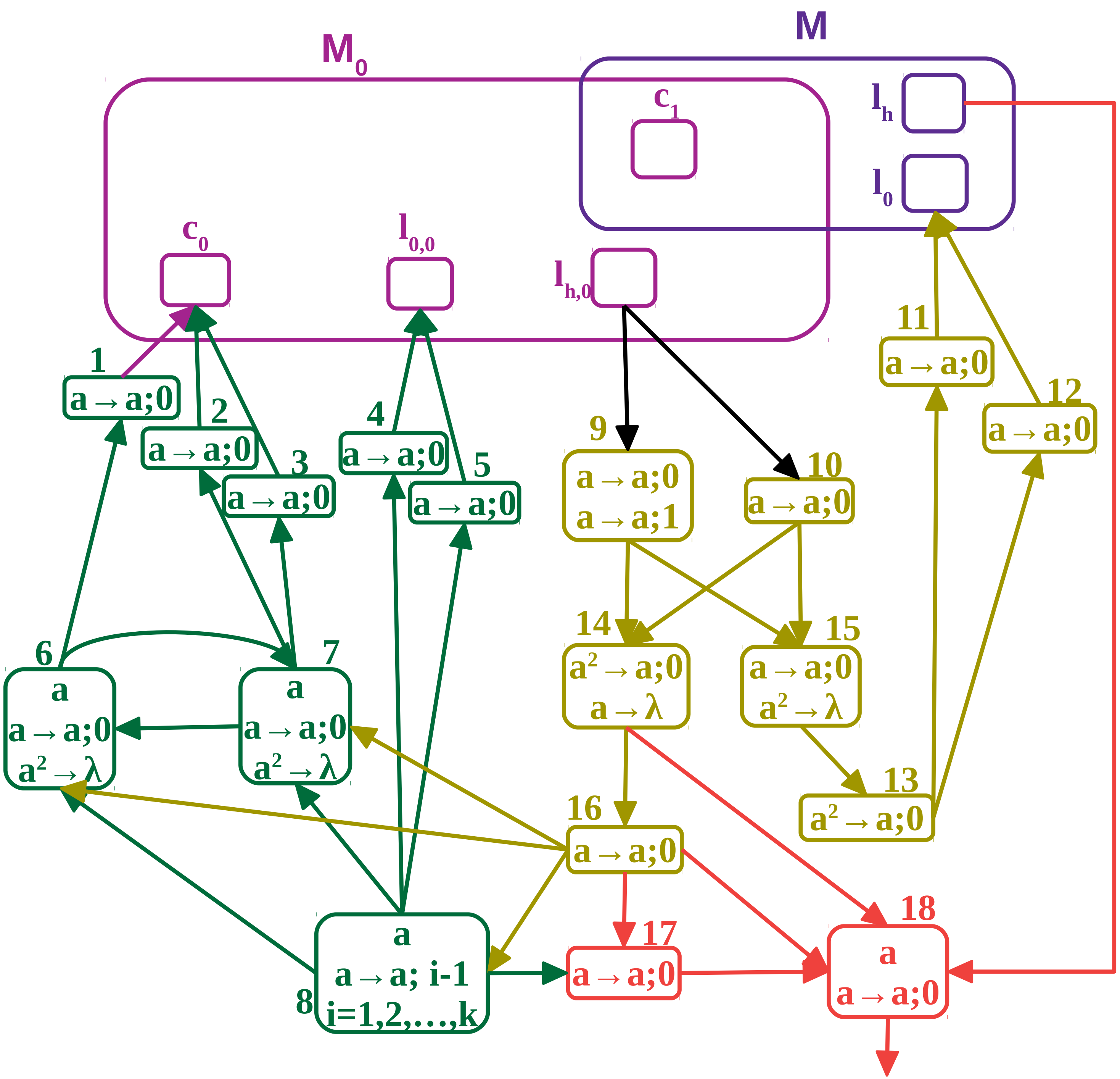
$\mathbf{10^{i_t}1}$

$$y = 10^{i1}1 \mid 0^{j1}1 \mid 10^{i2}1 \mid 0^{j2}1 \mid \ldots \mid 10^{im}1 \mid 0^{(jm+s)}1.$$

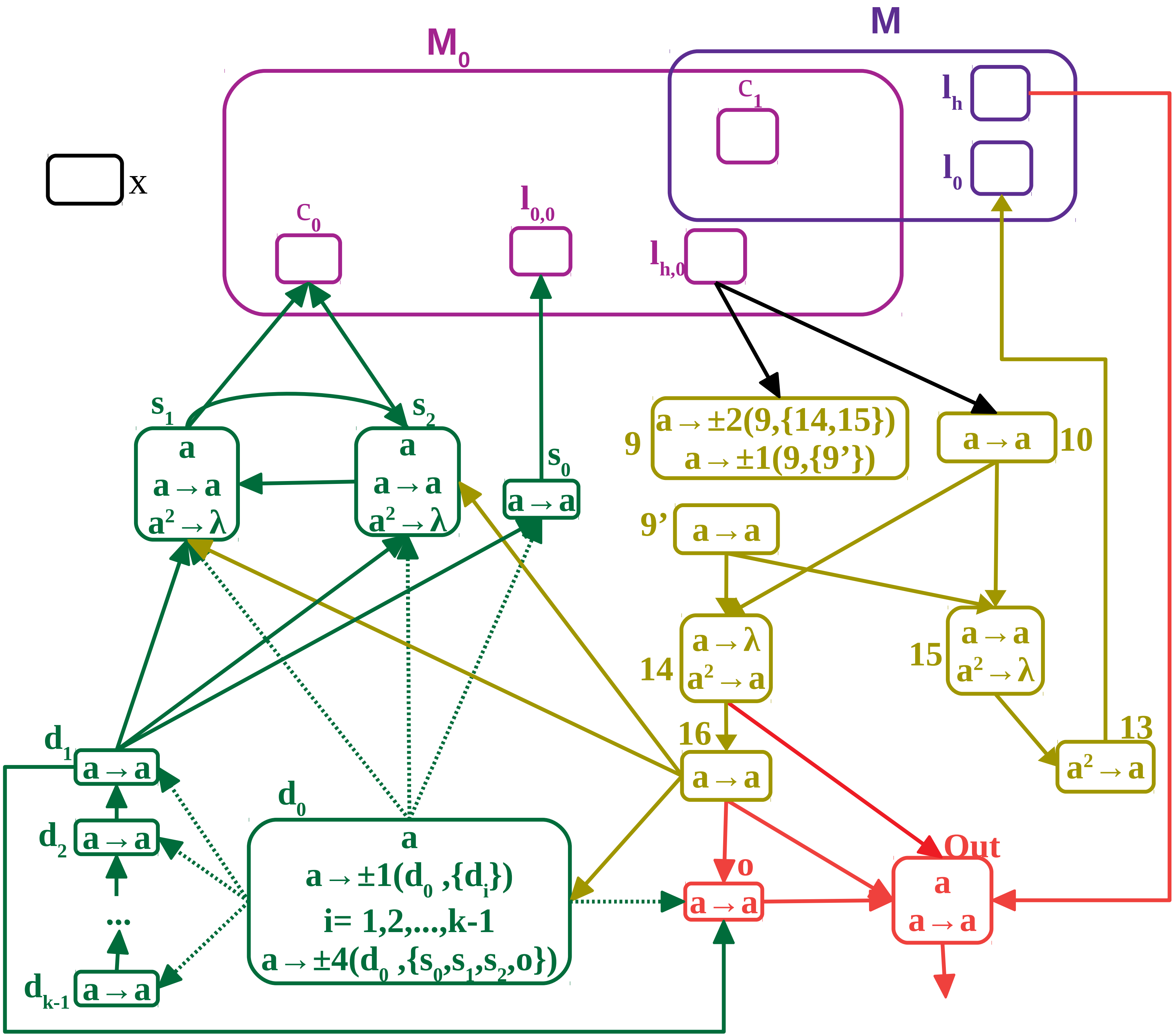
SNPSP Systems for RE Languages (Version 1)



SNP System for RE Languages



SNPSP System for RE Languages (Version 0)



SNPSP Systems & CF Languages

String Generation Algorithm

SNPSP + CFL: Context-free Grammar (GNF)

Greibach Normal Form Grammar: $G=(N,T,S,P)$

$N = \{n_1,n_2,\dots, n_x\}$, (non-terminal symbols)

$T=\{t_1,t_2,\dots,t_y\}$ (terminal symbols)

$S \in N$ (start symbol)

P is the set of production rules of the forms:

R_j: $n_{rj} \rightarrow t_{r'j}N_j$ where $n_{rj} \in N$, $t_{r'j} \in T$, $N_j \in N^*$, $1 \leq r_j \leq x$, $1 \leq r'_j \leq y$.

SNPSP + CFL: Context-free Grammar (GNF)

Example: $G=(N=\{A,B,C\},T=\{a,b,c\},A,P)$

$R_1: A \rightarrow aABB$

$R_2: A \rightarrow bBBC$

$R_3: A \rightarrow a$

$R_4: B \rightarrow b$

$R_5: C \rightarrow c$

Derive: aabbbcb

$$A \xrightarrow{R_1} aABB \xrightarrow{R_3} aaBB \xrightarrow{R_2} aabBBCB \xrightarrow{R_4} aabbBCB$$

$$\xrightarrow{R_4} aabbbCB \xrightarrow{R_5} aabbbcB \xrightarrow{R_4} aabbbcb$$

SNPSP + CFL: String Generation Algorithm

Grammar: **G=(N,T,S,P)**

1. Place initial symbol **S** on top of stack.

2. Pop: Get the **non-terminal** symbol on top of the stack.

3. Compare: Check the top symbol and see which production rule $\mathbf{R_j}$: $\mathbf{n_{rj} \rightarrow t_{r'j}N_j}$ can be applied on the top symbol. Non-deterministically apply one of the applicable rule.

4. Output + Push: Output the terminal $\mathbf{t_{r'j}}$ symbol of the rule and push the **non-terminal** symbols $\mathbf{N_j}$ back to the stack.

5. Repeat steps 2-4 until stack is empty.

SNPSP + CFL: String Generation Algorithm

No	Stack	Rule	New Stack	Output
1	A*	R_1:A → aABB	BBA*	a
2	BBA*	R_3:A → a	BB*	a
3	BB*	R_2:A → bBBC	BCBB*	b
4	BCBB*	R_4:B → b	BCB*	b
5	BCB*	R_4:B → b	BC*	b
6	BC*	R_5:C → c	B*	c
7	B*	R_4:B → b	*	b

SNPSP + CFL: Encoding

Example: $V=\{a_1=p, a_2=q, a_3=r, a_4=s\}$, $k+1 = 4+1 = 5$

$x = sqqrp$

$val_4(s) = 4_5$

$val_4(sq) = 10_5 \cdot val_4(s) + val_4(q) = \boxed{40_5 + 2_5} = \boxed{42_5}$

$val_4(sqq) = 10_5 \cdot val_4(sq) + val_4(q) = \boxed{420_5 + 2_5} = \boxed{422_5}$

$val_4(sqqr) = 10_5 \cdot val_4(sqq) + val_4(r) = \boxed{4220_5 + 3_5} = \boxed{4223_5}$

$val_4(sqqrp) = 10_5 \cdot val_4(sqqr) + val_4(p) = \boxed{42230_5 + 1_5} = \boxed{42231_5}$

SNPSP + CFL: Encoding

Example: $V=\{a_1=p, a_2=q, a_3=r, a_4=s\}$, $k+1 = 4+1 = 5$

$x = sqqrp$, $y=pqrs$

$val_4(sqqrp) = 42231_5$

$val_4(pqrs) = 1234_5$

$val_4(xy) = val_4(x)(10_5)^{|y|}+val_4(y)$

$= 42231_5(10_5)^4+1234_5$

$= 42231_5(10000_5)+1234_5$

$= 422310000_5+1234_5$

$= 422311234_5$

$=val_4(sqqrppqrs)$

SNPSP + CFL: Encoding

Example: $V=\{a_1=p,\ a_2=q,\ a_3=r,\ a_4=s\}$, $k+1 = 4+1 = 5$

$x = sqqrp$

$val_4(sqqrp) = 42231_5$

$val_4(sqqr) = val_4(sqqrp)/(10_5) = 4223_5$

$val_4(p) = val_4(sqqrp) \ \%\ (10_5) = 1_5$

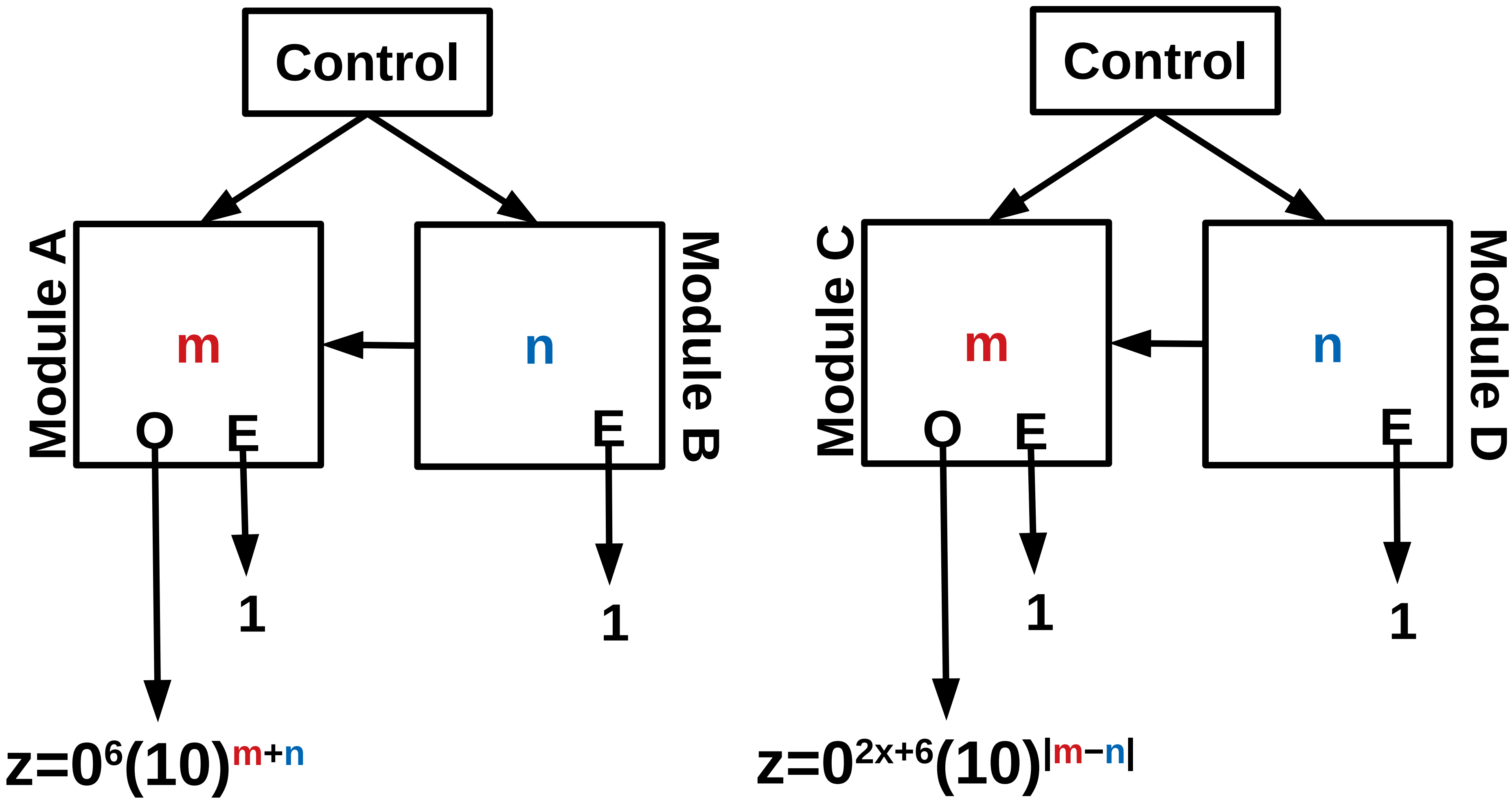
SNPSP + CFL: String Generation Algorithm

No	Stack	Rule	New Stack	Output
1	1_4	R_1:A → aABB	221_4	a
2	221_4	R_3:A → a	22_4	a
3	22_4	R_2:A → bBBC	2322_4	b
4	2322_4	R_4:B → b	232_4	b
5	232_4	R_4:B → b	23_4	b
6	23_4	R_5:C → c	2_4	c
7	2_4	R_4:B → b	0_4	b

SNPSP Systems & CF Languages

Arithmetic Memory Module

Arithmetic-Memory Module: Addition, Subtraction



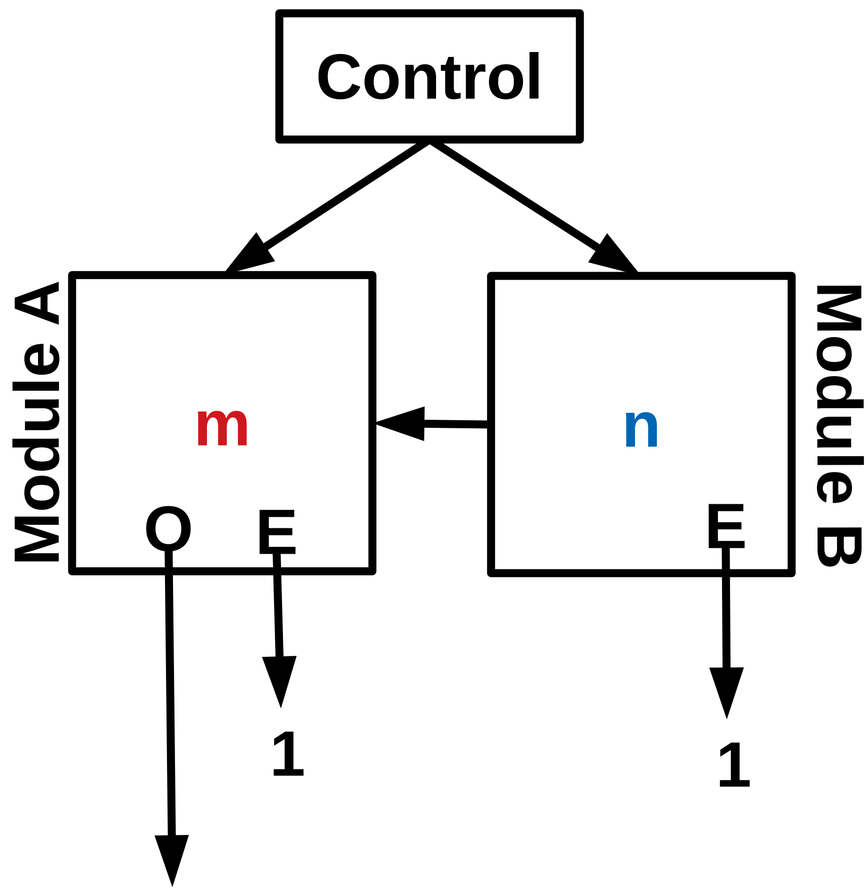
Addition: $|z|_1 = m + n$

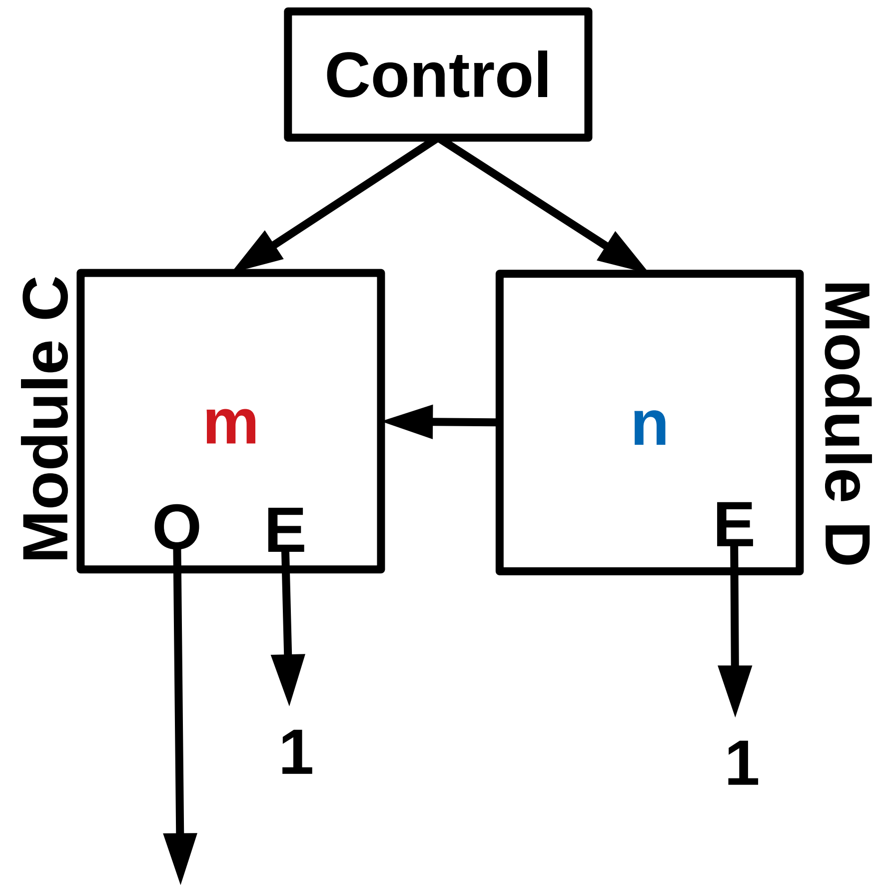
Difference: $|z|_1 = |m - n|$

Arithmetic-Memory Module: Multiplication, Division

$i = 1$ if $(m \% n) \neq 0$

$i = 0$ if $(m \% n) = 0$





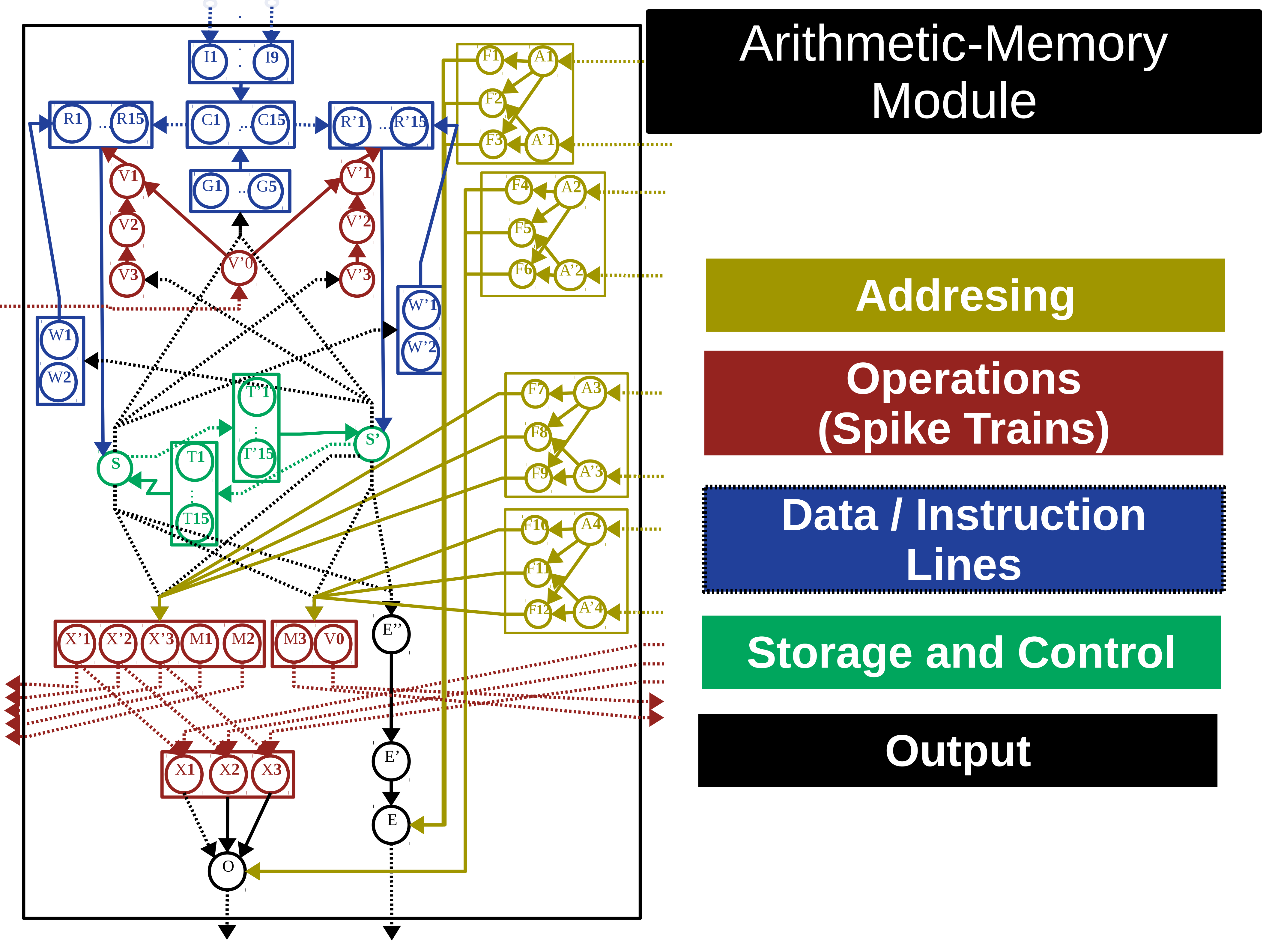
$$z=0^{11}((10)^{m}(0\ 8\ ))^{n-1}(10)^{m}$$

$$z=0^{6}((0^{5})^{n-1}\ (10^{4}))^{q}(0^{5})^{i\cdot n}$$

$$q=m/n$$

Multiplication: $|z|_1=m\cdot n$

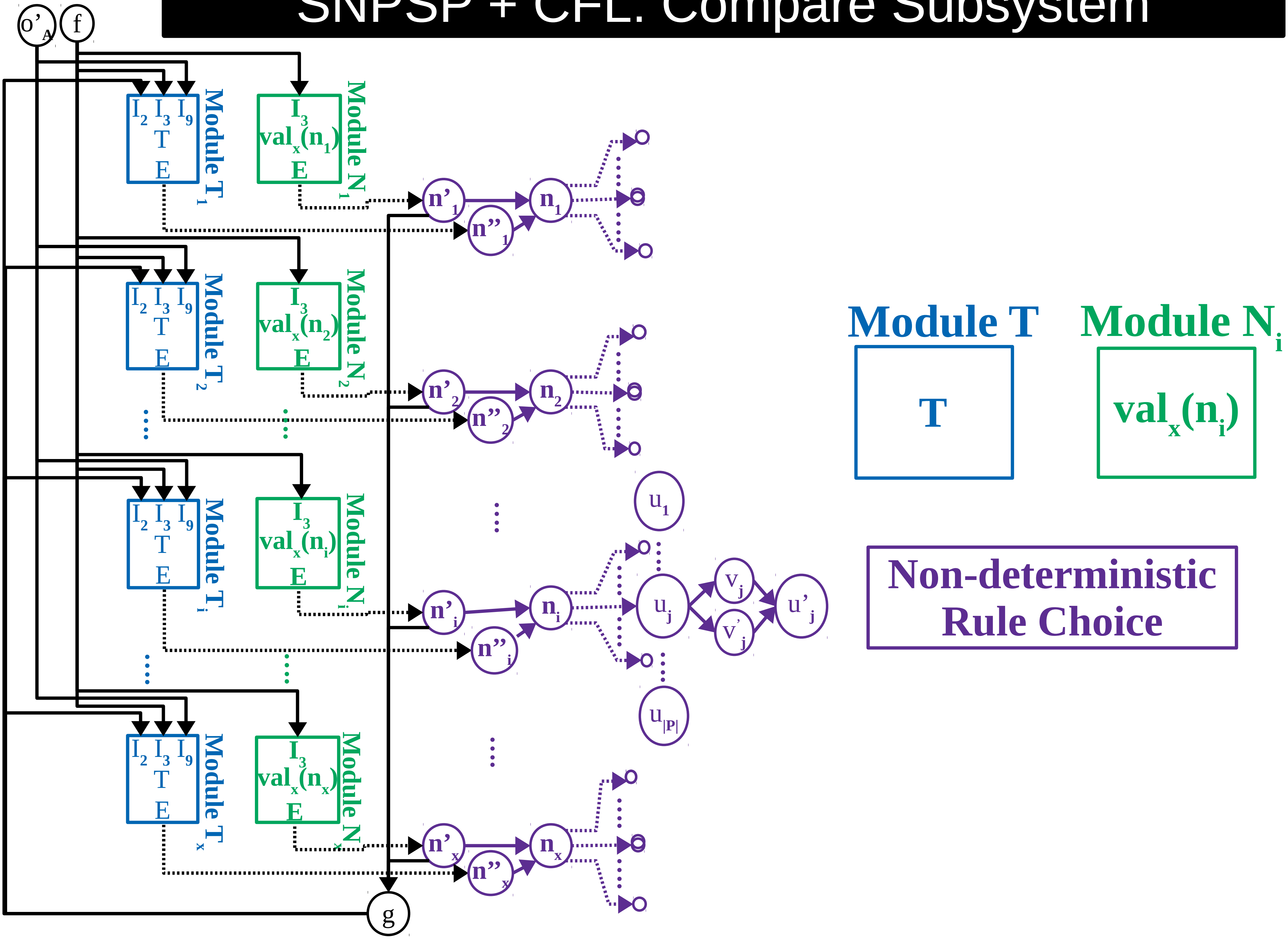
Division: $|z|_1=m/n$


Arithmetic-Memory Module
Addresing
Operations (Spike Trains)
Data / Instruction Lines
Storage and Control
Output
I1 I9
R1 R15 C1 C15 R'1 R'15
G1 G5
V1 V2 V3 V'0 V'1 V'2 V'3
W1 W2 W'1 W'2
T'1 T'15 T1 T15
S S'
F1 A1 F2 F3 A'1
F4 A2 F5 F6 A'2
F7 A3 F8 F9 A'3
F10 A4 F11 F12 A'4
X'1 X'2 X'3 M1 M2 M3 V0
E'' E' E
X1 X2 X3
O

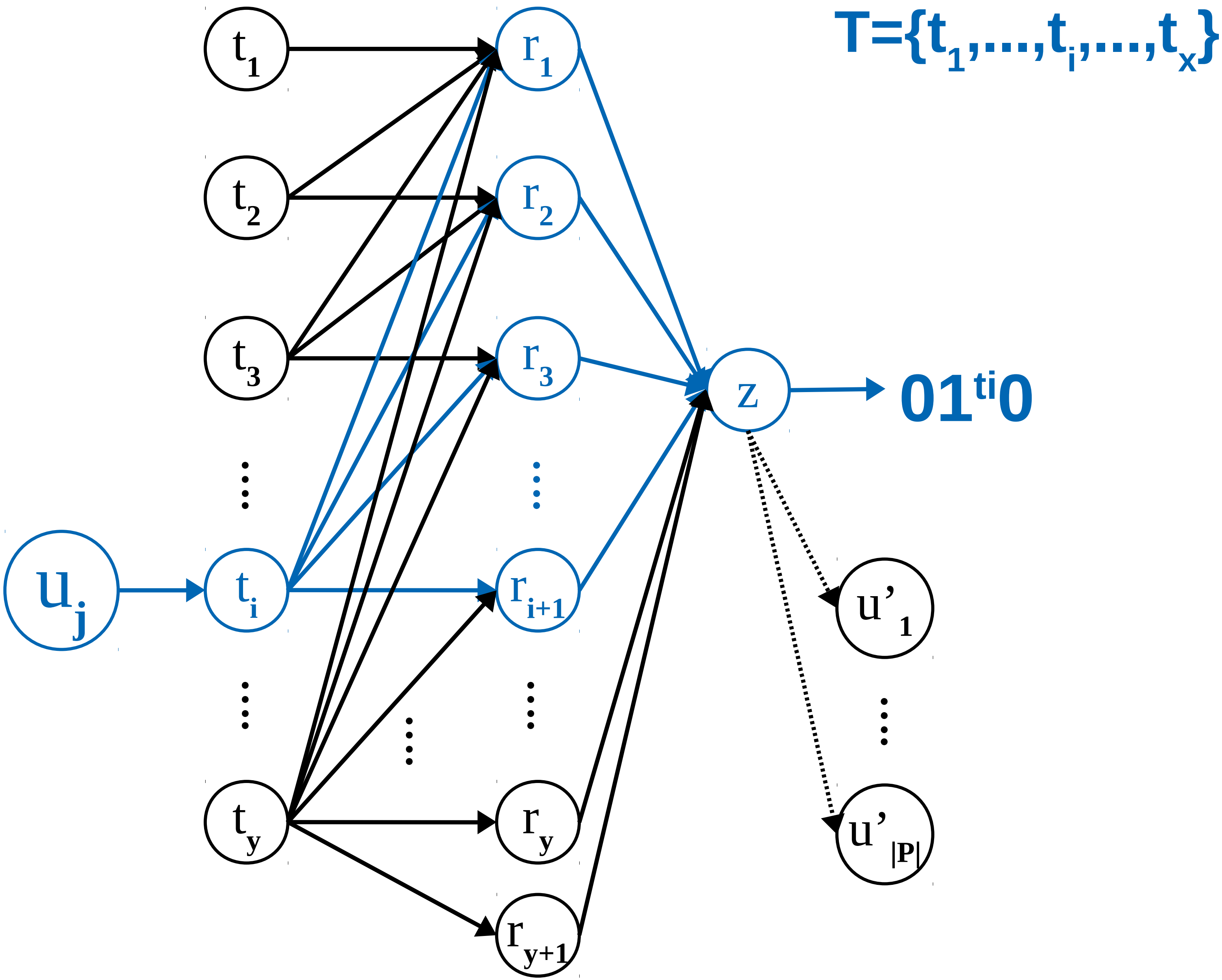
SNPSP Systems & CF Languages

Stack Operation Subsystems

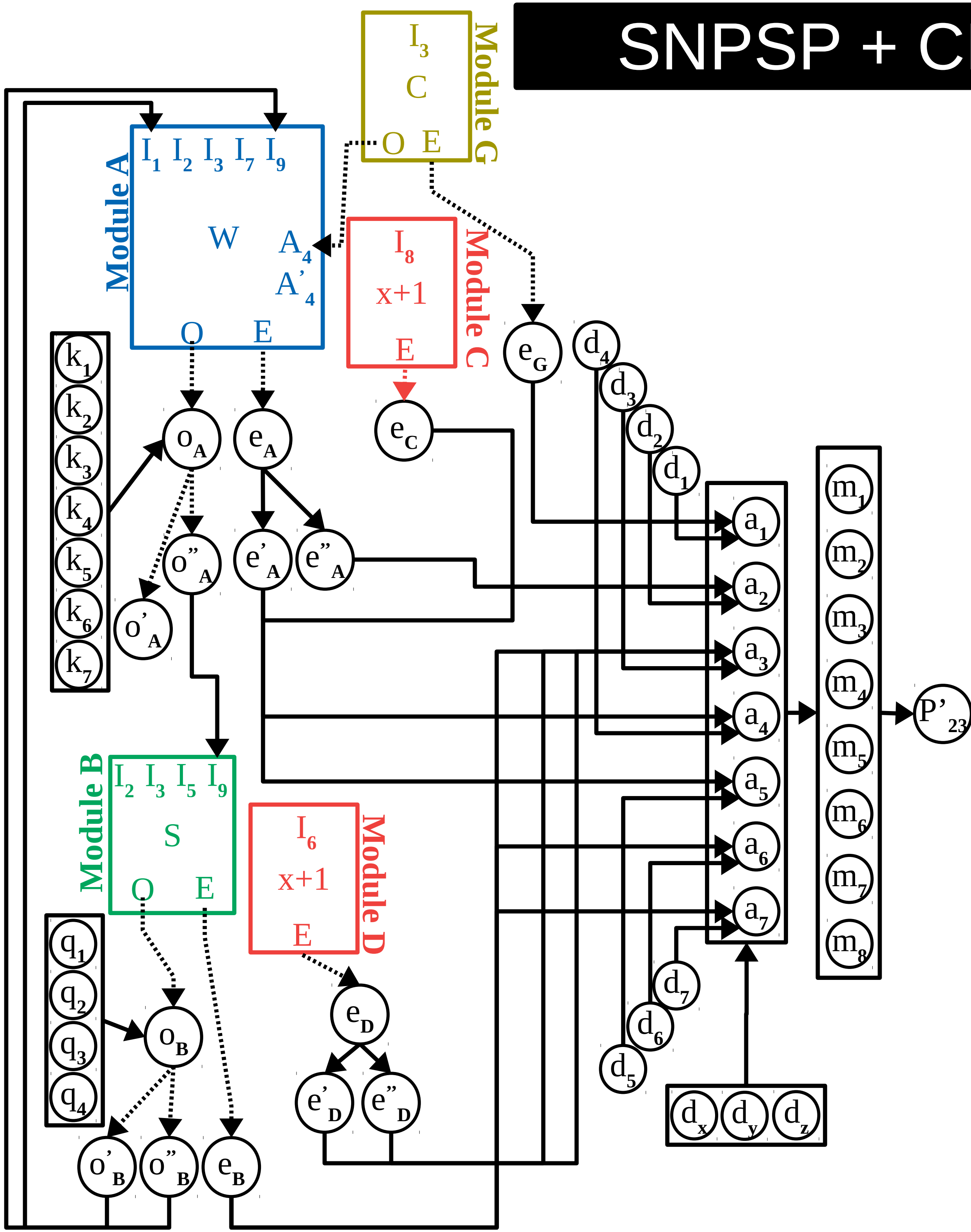
SNPSP + CFL: Compare Subsystem



SNPSP + CFL: Output Subsystem



SNPSP + CFL: Pop Subsystem



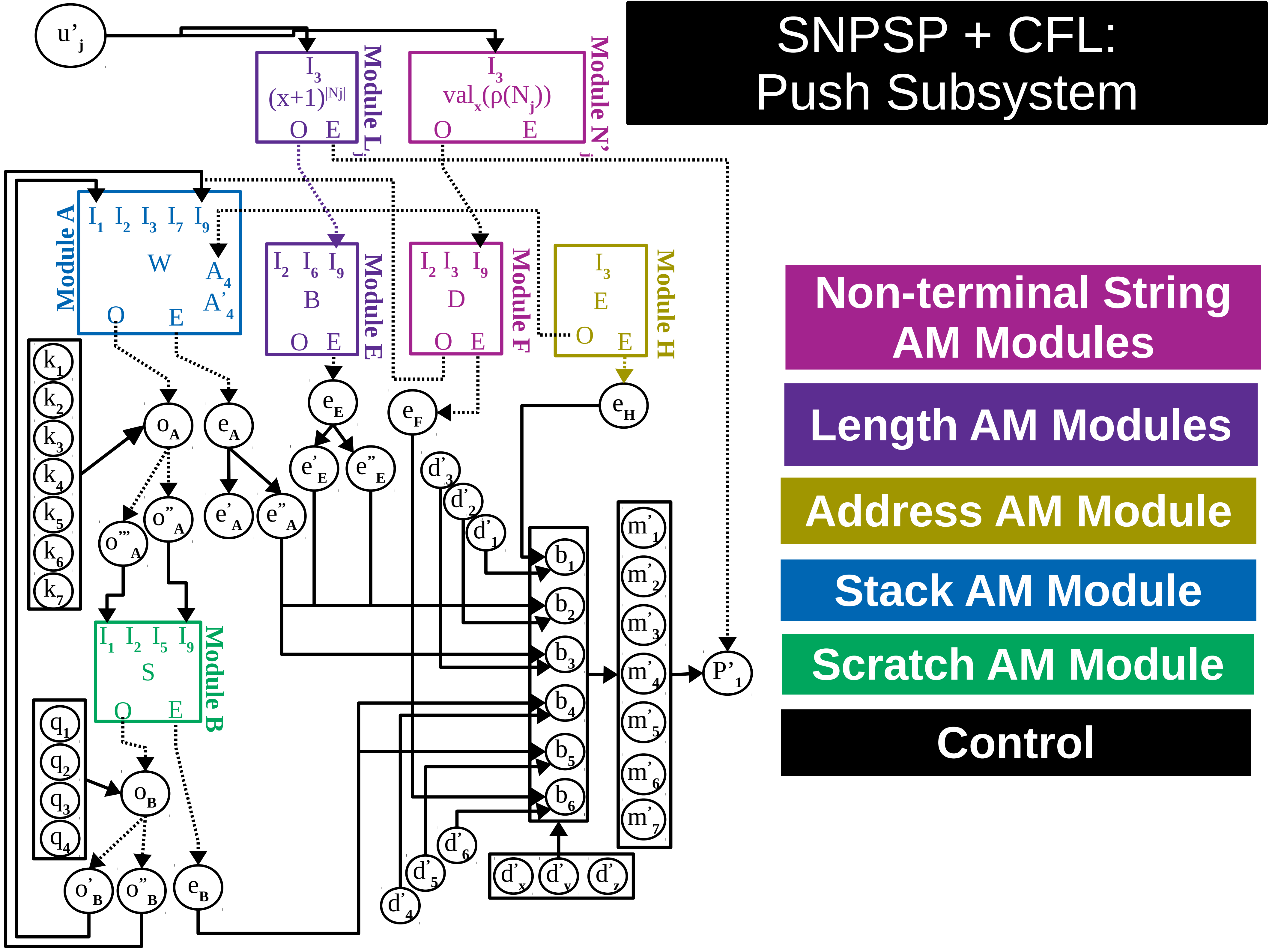
Address AM Module

Base AM Modules

Stack AM Module

Scratch AM Module

Control


SNPSP + CFL:
Push Subsystem
Non-terminal String AM Modules
Length AM Modules
Address AM Module
Stack AM Module
Scratch AM Module
Control
Module A
Module B
Module E
Module F
Module H
Module L_j
Module N'_j
u'_j
I_3
$(x+1)^{|N_j|}$
$val_x(\rho(N_j))$
W
S
B
D
E
O
A_4
A'_4
o_A
o''_A
o'''_A
e_A
e'_A
e''_A
e_E
e'_E
e''_E
e_F
e_H
o_B
o'_B
o''_B
e_B
P'_1

Summary + Future Work

Conclusion

Summary:

1. We were able to create procedures for constructing SNPSP systems that generate ***FIN***, ***REG***, ***RE*** languages These results were presented at *18th International Conference on Membrane Computing, 24-28 July 2017 - Bradford, UK.*

2. We implemented an **Arithmetic-Memory** module that can perform arithmetic operations and store numbers. This result was presented at *6th Asian Conference on Membrane Computing, 21-25 September 2017, Chengdu (P.R. China)*

3. We were able to create a procedure for constructing SNPSP systems that generate **CF** languages.

4. We we are able to simulate **forgetting rules** and **rules with delay** in SNPSP system and was able to show how, to some extent, SNP systems can simulate **plasticity rules**.

Conclusion

Possible Future Works:

1. Improve the design of the AM modules. Can it be made smaller with lost of functionality? Additional operators may be added.

2. Instead of language generation, compare the smallest know SNP to SNPSP systems. Does having plasticity rules decreases the size of the current smallest SNPSP system compared to the current smallest SNPSP systems?

3. Is it possible use only one type of plasticity rule action (i.e. ± only) and still have a Turing-complete model?

Thank you!

Acknowledgments

R.T.A. de la Cruz is grateful for the Department of Science and Technology (DOST)'s support through the Engineering Research and Development forTechnology (ERDT)'s graduate scholarship program.