





Basic Arithmetic Calculations Through Virus-Based Machines

Antonio Ramírez-de-Arellano¹(✉), David Orellana-Martín¹,
and Mario J. Pérez-Jiménez^{1,2}

¹ Research Group on Natural Computing, Department of Computer Science and Artificial Intelligence, Universidad de Sevilla, Avda. Reina Mercedes s/n, 41012 Seville, Spain

{aramirezdearellano,dorellana,marper}@us.es

² SCORE Lab, I3US, Universidad de Sevilla, 41012 Seville, Spain

Abstract. In Natural Computing, several models of computation based on processes occurring in nature exist. While some of them are well-established computing framework, there are some types of devices that are underdeveloped. This is the case of Virus Machines, framework inspired by the movement of viruses between hosts, and how can they be replicated while certain events happen. The relevance of this work lies in the formal definition of the framework and both the insights presented about the formal verification of the different designs and the possible new research lines.

In this work, Virus Machines are studied from a numerical point of view. In this sense, five different devices regarding the four basic arithmetic operators are created, and some insights about the proofs of their correctness are stated. While addition, subtraction and multiplication require only of one device, for division two different machines will be designed: one for the quotient of the division and the other for the remainder.

Keywords: Natural computing · Virus machine · Arithmetic calculator · Information fusion

1 Introduction

This work can be considered as a contribution to the area of *Natural Computing*, which is a field of research that investigates both human-designed computing inspired by nature and computing that occurs in nature.

In virology, a virus is a parasitic biological agent that can only reproduce after infecting a host cell. Every animal, plant, and protist species on this planet has been infected by viruses. Viruses can transmit from one host to another through various routes (e.g., conjunctival route, mechanical route, etc.). For additional details on viruses, refer to [2].

In this study, a new computing paradigm, introduced in [1], based on the transmissions and replications of viruses, is introduced. This paradigm provides non-deterministic computing models that consist of several cell-like *hosts* connected to one another by *channels*. Viruses are placed in the hosts and can

transmit from one host to another by passing through a channel, and can replicate itself while transmitting. These processes are controlled by several instructions, which are attached to the channels. These systems can be considered as a heterogeneous network that consists of:

- A *virus transmission network*: a weighted directed graph, wherein each node represents a *host* and each arc represents a *transmission channel* through which viruses can transmit between hosts or exit to the environment. In addition, each arc has associated a weight (natural number $w > 0$), which indicates the number of viruses that will be transmitted.
- An *instruction transfer network*: a weighted directed graph, wherein each node represents a *control instruction unit* and each edge represents an optional *instruction transfer path* with a positive integral weight.
- An *instruction-channel control network*: an undirected graph, wherein each node represents either a control instruction or a transmission channel and each edge represents a relationship between an instruction and a channel.

The computing models of this paradigm are universal (equivalent in power to Turing machines) when there is no limit on the number of viruses present in any host during a computation. The paper is organized as follows. Next section, the computing paradigm of virus machines is presented. In Sect. 3, the different modules concerning the basic arithmetic operators are presented. The paper ends with some open problems and concluding remarks.

2 Virus Machines

In what follows we formally define the **syntax** of the *Virus Machines*.

Definition 1. A *Virus Machine* of degree (p, q) , $p \geq 1, q \geq 1$ is a tuple $\Pi = (\Gamma, H, I, D_H, D_I, G_C, n_1, \dots, n_p, i_1, h_{out})$, where:

- $\Gamma = \{v\}$ is the singleton alphabet;
- $H = \{h_1, \dots, h_p\}$ and $I = \{i_1, \dots, i_q\}$ are ordered sets such that $v \notin H \cup I$, $H \cap I = \emptyset$ and $h_{out} \notin I \cup \Gamma$: either $h_{out} \in H$ or h_{out} represents the environment (denoted by h_0);
- $D_H = (H \cup \{h_{out}\}, E_H, w_H)$ is a weighted directed graph, where $E_H \subseteq H \times (H \cup \{h_{out}\})$, $(h, h) \notin E_H$ for each $h \in H$, $out-degree(h_{out}) = 0$ and w_H is a mapping from E_H onto $\mathbb{N} \setminus \{0\}$;
- $D_I = (I, E_I, w_I)$ is a weighted directed graph, where $E_I \subseteq I \times I$, w_I is a mapping from E_I onto $\mathbb{N} \setminus \{0\}$ and the out-degree of each node is less than or equal to 2;
- $G_C = (V_C, E_C)$ is an undirected bipartite graph, where $V_C = I \cup E_H$ being $\{I, E_H\}$ the partition associated with it: every edge connects an element from I with, at most, an arc from E_H ;
- $n_j \in \mathbb{N}(1 \leq j \leq p)$.

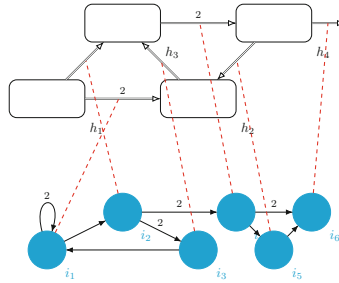


Fig. 1. Structure of a Virus Machine

A Virus Machine (VM, for short) $\Pi = (\Gamma, H, I, D_H, D_I, G_C, n_1, \dots, n_p, i_1, h_{out})$ of degree (p, q) , can be viewed as an ordered set of p hosts labelled with h_1, \dots, h_p , where host h_j initially contains exactly n_j viruses, and an ordered set of q control instruction units labelled with i_1, \dots, i_q . Symbol h_{out} represents the output region: it can be a host in the case that $h_{out} \in H$ or h_{out} can refer to the environment in the case that $h_{out} = h_0$. Arcs from the directed graph D_H represent transmission channels through which viruses can transmit from one host h_s (different from h_{out}) to another different host $h_{s'}$, or to the environment. If $s' = 0$, viruses may exit to the environment. In any moment, at most one instruction is activated and then the channel $(h_s, h_{s'})$ (arc in G_C) with weight $w_{s,s'}$ attached with it, will be opened. Then, $w_{s,s'}$ viruses will be transmitted/replicated from h_s to $h_{s'}$. By default, each channel is closed.

Arcs from the directed graph D_I represent instruction transfer paths, and they have associated within a weight. Finally, the undirected bipartite graph G_C represents the instruction-channel network by which an edge $\{i_j, (h_s, h_{s'})\}$ indicates a control relationship between instruction i_j and channel $(h_s, h_{s'})$.

Graphically, a virus machine of degree $(4, 6)$ with 4 hosts and 6 control instructions can be represented as a heterogeneous network consisting of three graph, as illustrated in Fig. 1. Each host is depicted as a rectangle and each instruction is depicted as a circle. Each arrow is either a virus transmission channel linking the hosts (or pointing to the environment), or an instruction transfer path linking the instructions; in both cases, each arrow is assigned with a positive integral weight (the weight 1 is not marked for simplicity). The control relationships between instructions and channels are represented as dotted lines.

In what follows, the semantics associated with the computing model of the virus machines is described. An instantaneous description or a configuration \mathcal{C}_t at an instant t of a virus machine is described by a tuple $(a_{0,t}, a_{1,t}, \dots, a_{p,t}, u_t)$ where $a_{0,t}, a_{1,t}, \dots, a_{p,t}$ are natural numbers, $u_t \in I \cup \{\#\}$, where $\# \notin H \cup h_0 \cup I$ is an object for characterizing halting configurations. The meaning of \mathcal{C}_t is the following: at instant t the environment contains exactly $a_{0,t}$ viruses and the host h_s contains exactly $a_{s,t}$ viruses, and if $u_t \in I$, then the instruction u_t will be activated at step $t + 1$ (otherwise, if $u_t = \#$, then no instruction will be activated). The initial config-

uration of the system $\Pi = (\Gamma, H, I, D_H, D_I, G_C, n_1, \dots, n_p, i_1, i_{out})$ is $\mathcal{C}_0 = (0, n_1, \dots, n_p, i_1)$. A configuration $\mathcal{C}_t = (a_{0,t}, a_{1,t}, \dots, a_{p,t}, u_t)$ yields configuration $\mathcal{C}_{t+1} = (a_{0,t+1}, a_{1,t+1}, \dots, a_{p,t+1}, u_{t+1})$ in one *transition step* if we can pass from \mathcal{C}_t to \mathcal{C}_{t+1} if we can pass from \mathcal{C}_t to \mathcal{C}_{t+1} in the following form.

- (a) First, given that \mathcal{C}_t is a non-halting configuration we have $u_t \in I$. Then the control instruction unit u_t is activated.
- (b) If u_t is attached to a channel $(h_s, h_{s'})$ then the channel will be opened and:
 - If $a_{s,t} \geq 1$ then only one virus is consumed from host h_s and $w_{s,s'}$ copies of v are produced in the region $h_{s'}$.
 - If $a_{s,t} = 0$ then no virus is consumed from host h_s and no virus is produced in the region $h_{s'}$.
- (c) If u_t is not attached to any channel then there is no transmission of viruses.
- (d) Object $u_{t+1} \in I \cup \{\#\}$ is obtained as follows:
 - If $out-degree(u_t) = 2$ then there are two different instructions $u_{t'}$ and $u_{t''}$ such that $(u_t, u_{t'}) \in E_I$ (with weight $w_{t,t'}$) and $(u_t, u_{t''}) \in E_I$ (with weight $w_{t,t''}$).
 - If instruction u_t is attached to a channel $(h_s, h_{s'})$:
 - * If $a_{s,t} \geq 1$ then u_{t+1} is the instruction corresponding to the *highest* weight path ($max\{w_{t,t'}, w_{t,t''}\}$). If $w_{t,t'} = w_{t,t''}$, the next instruction is selected in a non-deterministic way.
 - * If $a_{s,t} = 0$ then u_{t+1} is the instruction corresponding to the *lowest* weight path ($min\{w_{t,t'}, w_{t,t''}\}$). If $w_{t,t'} = w_{t,t''}$, the next instruction is selected in a non-deterministic way.
 - If instruction u_t is not attached to a channel, then the next instruction u_{t+1} ($u_{t'}$ or $u_{t''}$) is selected in a non-deterministic way.
 - If $out-degree(u_t) = 1$ then the system behaves deterministically and u_{t+1} is the instruction that verifies $(u_t, u_{t+1}) \in E_I$.
 - If $out-degree(u_t) = 0$ then $u_{t+1} = \#$, and \mathcal{C}_{t+1} is a halting configuration.

Definition 2. A *Virus Machine with input*, of degree (p, q, r) , $p \geq 1, q \geq 1, r \geq 1$ is a tuple $\Pi = (\Gamma, H, H_r, I, D_H, I, D_H, D_I, G_C, n_1, \dots, n_p, i_1, h_{out})$, where:

- $(\Gamma, H, I, D_H, D_I, G_C, n_1, \dots, n_p, i_1, h_{out})$ is a *Virus Machine of degree* (p, q) .
- $H_r = \{h_{i_1}, \dots, h_{i_r}\} \subseteq H$ is the ordered set of r input hosts and $h_{out} \notin H_r$.

If Π is a virus machine with input of degree (p, q, r) , $p \geq 1, q \geq 1, r \geq 1$ and $(\alpha_1, \dots, \alpha_r) \in \mathbb{N}^r$, the *initial configuration* of Π with input $(\alpha_1, \dots, \alpha_r)$ is $(0, n_1, \dots, n_{i_1} + \alpha_1, \dots, n_{i_r} + \alpha_r, \dots, n_p, i_1)$. Thus, each r -tuple $(\alpha_1, \dots, \alpha_r) \in \mathbb{N}^r$, is associated with an initial configuration $(0, n_1, \dots, n_{i_1} + \alpha_1, \dots, n_{i_r} + \alpha_r, \dots, n_p, i_1)$.

A computation of a virus machine Π with input $(\alpha_1, \dots, \alpha_r)$, denoted by $\Pi + (\alpha_1, \dots, \alpha_r)$, starts with configuration $(0, n_1, \dots, n_{i_1} + \alpha_1, \dots, n_{i_r} + \alpha_r, \dots, n_p, i_1)$ and proceeds as stated above. The result of a halting computation of $\Pi + (\alpha_1, \dots, \alpha_r)$ is the total number of viruses sent to the output region during the computation.

Next, a particular kind of virus machines providing function computing devices, is introduced.

Definition 3. Let $f : \mathbb{N}^k \rightarrow \mathbb{N}$ be a partial function. We say that function f is computable by a virus machine Π with k input hosts if for each $(x_1, \dots, x_k) \in \mathbb{N}^k$ we have the following:

- If $(x_1, \dots, x_k) \in \text{dom}(f)$ and $f(x_1, \dots, x_k) = z$, then every computation of $\Pi + (x_1, \dots, x_k)$ is a halting computation and the output is z .
- If $(x_1, \dots, x_k) \in \mathbb{N}^k \setminus \text{dom}(f)$, then every computation of $\Pi + (x_1, \dots, x_k)$ is a non-halting computation.

3 Arithmetic Operation Modules

In this section, we provide different modules for the different basic arithmetic operations; that is, addition, subtraction, multiplication and division (both quotient and remainder). In these types of machines, the encoding of natural numbers will be given by an unary encoding; that is, the input number $n \in \mathbb{N}$ will be encoded by v^n ; that is, n copies of the object v . In what follows, we introduce the different modules and some of their properties.

3.1 Add Module

This module is a Virus Machines with two input hosts, where the two terms of the addition are introduced as an input. The Virus Machine

$$\Pi_{add} = (\Gamma, H, H_r, I, D_H, D_I, G_C, n_1, n_2, i_1, h_{out})$$

of range $(2, 3, 2)$ where:

1. $\Gamma = \{v\}$, $H = H_r = \{h_1, h_2\}$, $I = \{i_1, i_2, i_3\}$;
2. $D_H = (\{h_0\} \cup H, E_H, w_H)$, where $E_H = \{(h_1, h_0), (h_2, h_0)\}$ and $w_H(h_1, h_0) = w_H(h_2, h_0) = 1$;
3. $D_I = (I, E_I, w_I)$, where $E_I = \{(i_1, i_1), (i_1, i_2), (i_2, i_2), (i_2, i_3)\}$ and $w_I(i_1, i_1) = w_I(i_2, i_2) = 2, w_I(i_1, i_2) = w_I(i_2, i_3) = 1$;
4. $G_C = (I \cup E_H, E_C)$, where $E_C = \{\{i_1, (h_1, h_0)\}, \{i_2, (h_2, h_0)\}\}$;
5. $n_i = 0, i \in \{1, 2\}$;
6. $h_{out} = h_0$.

A visual representation of this Virus Machine can be found in Fig. 2a. The idea is that all the viruses present in the initial configuration to the environment (i.e. the output region). Let (a, b) be the input of Π ; then the following invariants hold in this machine.

$$\phi(k) \equiv \mathcal{C}_k = (k, a - k, b, i_1), \text{ for } 0 \leq k \leq a$$

In the first k steps, a viruses go from h_1 to the environment. Then, in configuration a no viruses are available in h_1 , therefore the next instruction is i_2 .

$$\phi'(k) \equiv \mathcal{C}_{a+1+k} = (a + k, 0, b - k, i_2), \text{ for } 0 \leq k \leq b$$

From configuration \mathcal{C}_{a+1} , instruction i_2 is executed until no more viruses are available in the host h_2 , and thus i_3 will be selected as the next instruction.

Finally, in configuration \mathcal{C}_{a+b+2} , instruction i_3 , that is not attached to any channel, thus next instruction is $\#$ and the computation halts. Therefore, $\mathcal{C}_{a+b+3} = (a + b, 0, 0, \#)$.

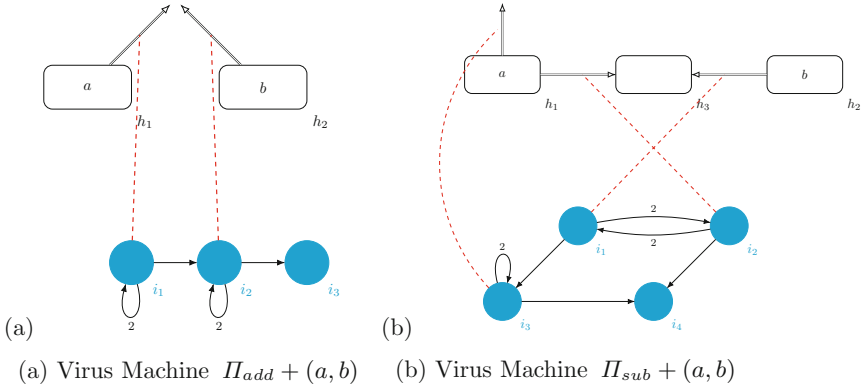


Fig. 2. Virus Machines for addition and subtraction operations

3.2 Sub Module

The Virus Machine presented computes the reduced difference, represented with the symbol $\overset{\bullet}{-}$ operation; that is, $a \overset{\bullet}{-} b = a - b$ if $a \geq b$, 0 otherwise. The Virus Machine

$$\Pi_{sub} = (\Gamma, H, H_r, I, D_H, D_I, G_C, n_1, n_2, n_3, i_1, h_{out})$$

of range $(3, 4, 2)$ where:

1. $\Gamma = \{v\}$, $H = \{h_3\} \cup H_r$, where $H_r = \{h_1, h_2\}$, $I = \{i_1, i_2, i_3, i_4\}$;
2. $D_H = (\{h_0\} \cup H, E_H, w_H)$, where $E_H = \{(h_1, h_3), (h_1, h_0), (h_2, h_3)\}$ and $w_H(h_1, h_0) = w_H(h_2, h_0) = 1$;
3. $D_I = (I, E_I, w_I)$, where $E_I = \{(i_1, i_2), (i_1, i_3), (i_2, i_1), (i_2, i_4), (i_3, i_3), (i_3, i_4)\}$ and $w_I(i_1, i_2) = w_I(i_2, i_1) = w_I(i_3, i_3) = 2, w_I(i_1, i_3) = w_I(i_2, i_4) = w_I(i_3, i_4) = 1$;
4. $G_C = (I \cup E_H, E_C)$, where $E_C = \{\{i_1, (h_2, h_3)\}, \{i_2, (h_1, h_3)\}, \{i_3, (h_1, h_0)\}\}$;
5. $n_i = 0, i \in \{1, 2, 3\}$;
6. $h_{out} = h_0$.

A visual representation of this Virus Machine can be found in Fig. 2b. The idea is that viruses from h_1 are “countered” by viruses from h_2 and, if any virus remains in h_1 , then goes to the environment. Let (a, b) be the input of Π , and let $mod(a, b)$ be the modulus operator; then the following invariants hold in this machine.

$$\phi(k) \equiv C_k = (0, a - \lfloor \frac{k}{2} \rfloor, b - \lceil \frac{k}{2} \rceil, k, i_{1+mod(k,2)}), \text{ for } 0 \leq k \leq \min\{2(a+1) - 1, 2b\}.$$

Here, two different cases can arise. If $a < b$, then the first $2(a+1) - 1$ steps, instructions i_1 and i_2 will alternate one after the other, in order to send to h_3 (a “garbage” host) one virus each host alternately. As h_1 will run out of viruses before h_2 , in the configuration $C_{2(a+1)-1}$, since no more viruses are available in host h_1 , instruction i_4 is selected as the next instruction, and configuration

$\mathcal{C}_{2(a+1)+1}$ is a halting configuration. Otherwise, if $a \geq b$, then the first $2b$ steps, instructions i_1 and i_2 will alternate one after the other. In this case, h_2 will run out of viruses before h_1 , thus in the configuration \mathcal{C}_{2b} , instruction i_3 will be the next instruction. Then, from that point, the following invariant holds:

$$\phi'(k) \equiv \mathcal{C}_{2b+k} = (k, a - b - k, 0, k, i_3), \text{ for } 0 \leq k \leq a - b$$

After $a - b$ steps, $a - b$ viruses will be sent to the environment and, since no more viruses are available in host h_1 , next instruction is i_4 , that will lead to a halting configuration $\mathcal{C}_{2b+a-b+2}$.

3.3 Mul Module

A Virus Machine capable of returning the multiplication of two given numbers is given below. The Virus Machine

$$\Pi_{mul} = (\Gamma, H, H_r, I, D_H, D_I, G_C, n_1, n_2, n_3, n_4, i_1, h_{out})$$

of range $(4, 5, 2)$ where:

1. $\Gamma = \{v\}$, $H = \{h_3, h_4\} \cup H_r$, where $H_r = \{h_1, h_2\}$, $I = \{i_1, i_2, i_3, i_4, i_5\}$;
2. $D_H = (\{h_0\} \cup H, E_H, w_H)$, where $E_H = \{(h_1, h_3), (h_2, h_4), (h_3, h_1), (h_3, h_0)\}$ and $w_H(h_1, h_3) = 2, w_H(h_2, h_4) = w_H(h_3, h_1) = w_H(h_3, h_0) = 1$;
3. $D_I = (I, E_I, w_I)$, where $E_I = \{(i_1, i_2), (i_1, i_5), (i_2, i_2), (i_2, i_3), (i_3, i_1), (i_3, i_1), (i_4, i_3)\}$ and $w_I(i_1, i_2) = w_I(i_2, i_2) = w_I(i_3, i_4) = 2, w_I(i_1, i_5) = w_I(i_2, i_3) = w_I(i_3, i_1) = w_I(i_4, i_3) = 1$;
4. $G_C = (I \cup E_H, E_C)$, where $E_C = \{\{i_1, (h_2, h_4)\}, \{i_2, (h_1, h_3)\}, \{i_3, (h_3, h_1)\}, \{i_4, (h_3, h_0)\}\}$;
5. $n_i = 0, i \in \{1, 2, 3, 4\}$;
6. $h_{out} = h_0$.

We can see the idea of this Virus Machine in Fig. 3. Let (a, b) the the input of Π . The following invariant holds:

$$\phi(k) \equiv \mathcal{C}_{k(3a+3)} = (a \cdot k, a, b - k, 0, k, i_1), \text{ for } 0 \leq k \leq b$$

From the first configuration, viruses from h_2 will be sent to host h_4 (a “garbage” host) in order to “count” the number of times that viruses from h_1 must be sent to the environment. Then, instruction i_2 will send two times the number of viruses in h_1 to h_3 in a steps. When it finishes, it goes to instruction i_3 , that will be executed alternately with i_4 until no viruses remain in h_3 . This will happen in $2a + 1$ steps. When this happens, the next instruction selected is, again, i_1 . This process will go on until no viruses remain in h_2 , where instruction i_5 is selected and leads to a halting configuration in the next step, in configuration $\mathcal{C}_{b(3a+3)+2}$.

3.4 Quotient Module

The division operation can be seen as a function $f : \mathbb{N}^2 \rightarrow \mathbb{N}^2$, such that $f(D, d) = (q, r)$, fulfilling the following requirement: $D = d \cdot q + r$. q is said

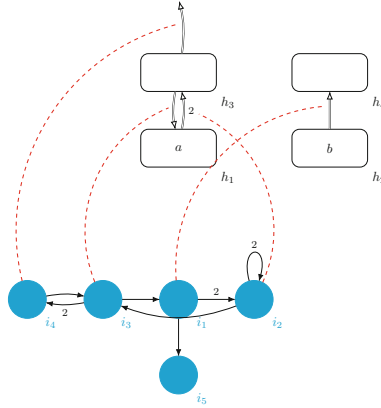


Fig. 3. Virus Machine $\Pi_{mul} + (a, b)$

to be the *quotient* of the division, while r is the *remainder*. The device returning the *quotient* is the Virus Machine

$$\Pi_{quo} = (\Gamma, H, H_r, I, D_H, D_I, G_C, n_1, n_2, n_3, n_4, i_1, h_{out})$$

of range $(4, 5, 2)$ where:

1. $\Gamma = \{v\}$, $H = \{h_3, h_4\} \cup H_r$, where $H_r = \{h_1, h_2\}$, $I = \{i_1, i_2, i_3, i_4, i_5\}$;
2. $D_H = (\{h_0\} \cup H, E_H, w_H)$, where $E_H = \{(h_1, h_3), (h_2, h_4), (h_3, h_0), (h_4, h_2)\}$ and $w_H(h_2, h_4) = w_H(h_1, h_3) = w_H(h_3, h_0) = w_H(h_4, h_2) = 1$;
3. $D_I = (I, E_I, w_I)$, where $E_I = \{(i_1, i_2), (i_1, i_3), (i_2, i_1), (i_2, i_5), (i_3, i_4), (i_4, i_1), (i_4, i_4)\}$ and $w_I(i_1, i_2) = w_I(i_2, i_1) = w_I(i_4, i_4) = 2, w_I(i_1, i_3) = w_I(i_2, i_5) = w_I(i_3, i_4) = w_I(i_4, i_1) = 1$;
4. $G_C = (I \cup E_H, E_C)$, where $E_C = \{\{i_1, (h_2, h_4)\}, \{i_2, (h_1, h_3)\}, \{i_3, (h_1, h_0)\}, \{i_4, (h_4, h_2)\}\}$;
5. $n_i = 0, i \in \{1, 2, 3, 4\}$;
6. $h_{out} = h_0$

This Virus Machine can be depicted as in Fig. 4a

By definition, we will say that $a/0 = 0, a \in \mathbb{N}$. Let (a, b) be the input of Π . Then the following invariant holds:

$$\phi(k) \equiv C_{k(3b+3)} = (k, a - b \cdot k, b, b \cdot k - k, 0, i_1), \text{ for } 0 \leq k \leq \lfloor \frac{a}{b} \rfloor$$

From the first configuration, instructions i_1 and i_2 will alternately be executed until h_1 or h_2 go out of viruses. If h_2 goes out of viruses first, then it means that we can add 1 to the temporary quotient (i.e. the number of viruses in the environment). If this is the case, from i_1 , the instruction i_3 is selected, and one virus is sent from h_3 to the environment (taking into account that viruses in h_3 are not useful in any other sense, therefore we use them as “counters”). When h_1 runs out of viruses, and instruction i_2 is executed, the instruction selected is i_5 , leading to a halting configuration in $\mathcal{C}_{\lfloor a/b \rfloor \cdot (3b+3)+2}$. Let us recall that in this case we say that, by definition, $a/0 = 0, a \in \mathbb{N}$.

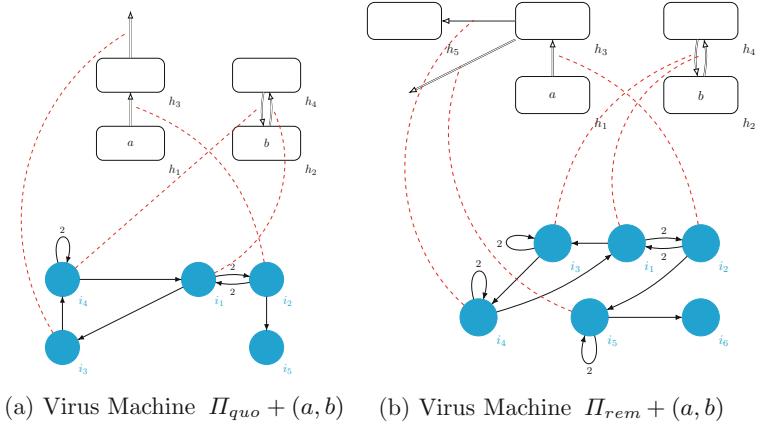


Fig. 4. Virus machines for the quotient and remainder of division operations

3.5 Remainder Module

While in the previous subsection, the Virus Machine answering the *quotient* of the division of two given numbers was given, below is presented one returning the *remainder* of such operation. The Virus Machine

$$\Pi_{rem} = (\Gamma, H, H_r, I, D_H, D_I, G_C, n_1, n_2, n_3, n_4, n_5, i_1, h_{out})$$

of range $(5, 6, 2)$ where:

1. $\Gamma = \{v\}, H = \{h_3, h_4, h_5\} \cup H_r$, where $H_r = \{h_1, h_2\}, I = \{i_1, i_2, i_3, i_4, i_5, i_6\}$;
2. $D_H = (\{h_0\} \cup H, E_H, w_H)$, where
 $E_H = \{(h_1, h_3), (h_2, h_4), (h_3, h_5), (h_3, h_0), (h_4, h_2)\}$ and $w_H(h_1, h_3) = w_H(h_2, h_4) = w_H(i_3, i_5) = w_H(h_3, h_0) = w_H(h_4, h_2) = 1$;
3. $D_I = (I, E_I, w_I)$, where
 $E_I = \{(i_1, i_2), (i_1, i_3), (i_2, i_1), (i_2, i_5), (i_3, i_3), (i_3, i_4), (i_4, i_1), (i_4, i_4), (i_5, i_5), (i_5, i_6)\}$ and $w_I(i_1, i_2) = w_I(i_2, i_1) = w_I(i_3, i_3) = w_I(i_4, i_4) = w_I(i_5, i_5) = 2, w_I(i_1, i_3) = w_I(i_2, i_5) = w_I(i_3, i_4) = w_I(i_4, i_1) = w_I(i_5, i_6) = 1$;
4. $G_C = (I \cup E_H, E_C)$, where
 $E_C = \{\{i_1, (h_2, h_4)\}, \{i_2, (h_1, h_3)\}, \{i_3, (h_4, h_2)\}, \{i_4, (h_3, h_5)\}, \{i_5, (h_3, h_0)\}\}$;
5. $n_i = 0, i \in \{1, 2, 3, 4, 5\}$;
6. $h_{out} = h_0$

This Virus Machine is represented visually in Fig. 4b.

In order to fulfil the basics of the division, that is, $D = d \cdot q + r$, we say that the remainder of the division $a/0, a \in \mathbb{N}$ is equal to a . Let (a, b) be the input of Π . Then the following invariant holds:

$$\phi(k) \equiv \mathcal{C}_{k(4b+3)} = (0, a - b \cdot k, b, 0, 0, b \cdot k, i_1), \text{ for } 0 \leq k \leq \lfloor \frac{a}{b} \rfloor$$

The idea is similar to the quotient module. First, we alternate between instructions i_1 and i_2 , until one of them run out of viruses. On the one hand, if

h_2 runs out of viruses first, then it means that b can be subtracted once again from a . When this happens, instruction i_5 is selected, that takes back all the viruses from h_4 to h_2 , when this process finishes, all the viruses from h_3 are sent to the host h_5 , that is a “garbage” collector, and it goes back to instruction i_1 . This happens each $4b + 3$ steps. On the other hand, if h_1 runs out of viruses first, the finalization protocol starts, first by activating the instruction i_5 , that sends all of the viruses from h_3 to the environment. This whole process leads to a halting configuration $\mathcal{C}_{\lfloor a/b \rfloor \cdot (4b+3) + \text{remainder}(a,b) + 3}$.

4 Conclusions and Future Work

The idea of this work is to establish some ideas for later applications in the framework of information fusion. In [3], the four basic arithmetical operators are used to give an explicit basic probability assignment (BPA) calculator by means of Spiking Neural P systems. Since the encoding is totally different in this framework, some changes must be introduced in order to obtain such kind of calculator.

Apart from that, and since the framework is in a very early stage, several research lines are open, as well as some ideas for extension of the model, in a similar way to extensions in the framework of Membrane Computing.

Acknowledgements. This work was supported by “Junta de Andalucía (Consejería de Economía, Conocimiento, Empresas y Universidad)” (P20.00486) – Desarrollo de modelos computacionales de especies invasoras en el Guadalquivir: herramientas de gestión para su control y prevención. D. Orellana-Martín also acknowledges Contratación de Personal Investigador Doctor. (Convocatoria 2019) 43 Contratos Capital Humano Línea 2. Paidi 2020, supported by the European Social Fund and Junta de Andalucía.

References

1. Chen, X., Pérez-Jiménez, M.J., Valencia-Cabrera, L., Wang, B., Zeng, X.: Computing with viruses. *Theoret. Comput. Sci.* **623**, 146–159 (2016)
2. Dimmock, N.J., Easton, A.J., Leppard, K.: *Introduction to Modern Virology*. Blackwell Publication, Malden (2007)
3. Zhang, G., Rong, H., Paul, P., He, Y., Neri, F., Pérez-Jiménez, M.J.: A complete arithmetic calculator constructed from spiking neural P systems and its application to information fusion. *Int. J. Neural Syst.* **31**(1), 2050055 (2001)