

Self Programming Machines (I).

J-P Moulin

Département d'Information et de Recherche Médicale.

Centre Hospitalier Interdépartemental de Clermont d'Oise.

2, rue des Finets.

60607 - Clermont d'Oise Cedex.

Keywords :

Functional modification

Self-modifying automaton

Stability by self-replication

Self-programming machine

Self-reference

Self-referential stability

ABSTRACT

The history of life on earth is full of examples of the responses of living creatures which are appropriate to the environment. This is particularly noticeable when living things are confronted with new conditions. Depending on the context, different names are given to these responses : learning, behaviour, adaptation. These responses all pose the problem of programming without a programmer or performing complicated tasks without human will or intervention. This self programming property seems ubiquitous and therefore must be implemented in various ways. Due to the diversity of its implementations, the most plausible hypothesis is that the self programming property results from a simple abstract law.

In this paper, a solution to this enigma of self programming is put forward. This solution is a simple abstract law : machines known as **self programming machines** or \mathbf{m}_{sp} , which associate two or more automata which change their internal organisation whenever the external data vary (**self-modifying automata** or \mathbf{a}_{sm}). After any transient steps, the \mathbf{m}_{sp} stabilises when it goes indefinitely in a limit cycle of length one (or fixed point). In this case the external value v is the index of a function f such as $f_v(v) = v$. It stabilises in a self replication process (stabilisation by self-replication) and the pair (f_v, v) is self referential.

Self programming machines suggest new contributions to the theories of replicating machines and self reference theory.

In this paper "Self Programming Machine (I)", the theory of such machines was studied, however in the second paper, "Self Programming Machine (II)", performances of an Ashby homeostat is compared when it is driven by a network of \mathbf{a}_{sm} , and a network of \mathbf{m}_{sp} .

1- Introduction.

The history of life on earth is full of examples of how living things have adapted their responses to the environment, in particular when they are confronted with new conditions. Depending on the context, different names are given to these responses : learning, behaviour, adaptation.

- In the plant world, there is the example of the orchid which can be pollinated because the calyx looks like a female wasp and the corolla exhales a smell identical to a female wasp (Kullenberg, 1956). This way the deluded male wasp carries the pollen from the stamen to the pistil.
- Among monocellular organisms, paramecia are able to learn and make appropriate reactions (Gelber, 1958).
- White ants are quickly able to find the shortest route in an extremely complicated man-made-maze when they work as a social group, whereas an isolated insect cannot achieve such a task (Goldberg, 1981).
- The immune system responds specifically to a new synthetic molecule (Lodish et al., 1995).
- Many authors consider that the performances of the genome for the evolution of species and that of the nervous system for behaviour, are similar. (Chauvin, 1985)

These examples all involve programming without a programmer or performing complicated tasks without human will or intention.

This self programming property seems ubiquitous and must therefore be implemented in various ways : Circuits of the nervous system, relationships between social insects, metabolic modifications or exchanges of chemical signals. Due to the diversity of its implementations, the most plausible hypothesis is that the self programming property results from a simple abstract law.

This self programming property gives rise to many arguments :

- If a living organism is exposed to new conditions, the activation of pre-existing circuits can be invoked in order to explain its response. This kind of hypothesis is often given when the nervous system recovers its functions after a lesion. This argument distances but does not refute the self programming capacity because the activated existing circuit must have been set up previously.
- When the pertinence of the response is mentioned, how much is real and how much is anthropomorphic illusion ?

There appears to be a paradox : Although living things seem very often to have appropriate responses when new situations arise, it is very difficult to find experiments which can be explained exclusively by the self programming property.

Three such experiments can be cited :

1 - A man permanently wears Dove prisms in front of his eyes. These prisms invert the right and the left side. After any days, the vestibulo ocular reflex is inverted (Gonshor et al., 1976).

2 – In another experiment, the attachment of the internal and external recti muscles of a monkey's eyeball were severed and re-attached in a crossed position so that a contraction of the external rectus would cause the eyeball to turn not outward but inward. When the wound was healed, he was surprised to discover that the two eyeballs still moved together, so that binocular vision was preserved (Marina, 1915).

3 - More recently, another author severed the nerves supplying the flexor and the extensor muscles of a spider monkey and rejoined them in a crossed position. After the nerves had regenerated the animal's arm movements were at first grossly uncoordinated but improved until an essentially normal mode of use was re-established (Sperry, 1947).

These results provide one solution to the enigma of self programming but there are perhaps many. The solution given here is a simple abstract law.

1 - In the first publication (Moulin, J-P., 1992), we studied automata which change their internal organisation whenever the external data vary. These automata (**self-modifying automata** or \mathbf{a}_{sm}) are deterministic. Their operating rules, the change in these rules and the initial conditions are randomly chosen once and definitively at the beginning.

2 - In the second paper (Moulin, J-P., 1999), the dynamics of chains of such automata were studied.

3 – In this paper “Self Programming Machines (I)”, (Moulin, J-P., 2000), was studied the theory of machines which associate two or more \mathbf{a}_{sm} or chain of \mathbf{a}_{sm} . These machines have the self programming property and are called **self-programming machines** or \mathbf{m}_{sp} . After a few transient steps, the \mathbf{m}_{sp} stabilises when it goes indefinitely into a limit cycle of length one (or fixed point). In this case the external value v is the index of a function f such as $f_v(v) = v$. It stabilises in a self replication process and the pair (f_v, v) is self referential.

4 – In the next paper “Self Programming Machines (II)”, we will compare performances of the Ashby homeostat when it is driven by a network of \mathbf{a}_{sm} , chain of \mathbf{a}_{sm} and \mathbf{m}_{sp} .

Finally, \mathbf{m}_{sp} provides new contributions to :

- Replicating machines (Von Neumann, 1966; Myhill, 1970; Codd, 1968) and gives a new stability model, namely stability by self replication.

- The theory of self referential machines (Thatcher, 1965). Many authors, in computer (Ashby, 1961) and natural sciences (Maturana & Varela, 1981; Zeleny, 1981) consider that self reference is a fundamental concept in comprehension of the mechanisms of perception, behaviour and associations in the brain (Bartlett & Suber, 1987).
- The theory of learning machines (Anthony, Biggs, 1991).

Comparison with others machines and automata.

- Some authors have studied the dynamics of automata, where connections and truth tables are randomly chosen once and definitively at the beginning, implying that internal rules remain fixed (Quenched model) (Kaufmann, 1969 ; Fogelman Soulié, 1985).
- Other authors have studied the dynamics of automata, the rules of which are randomly modified at each step. (Annealed model) (Derrida & Stauffer, 1986)

Finally, in order to simulate memory and pattern recognition in the brain, many authors have devised machines for which the internal operations are modified in order to minimise the difference between the output and goal value. Important work in this field includes : (Adaline Mattson, 1959; Widrow & Hoff, 1960), the association network (Anderson, 1972; Kohonen, 1972) the Cognitron (Fukushima, 1975) the Hopfield association Network (Hopfield, 1982) and the multi layered Back-propagation association network (Rumelhart et al, 1986).

2 - Formalism of self-programming machines.

Here, a self-programming machine is defined and the properties of its dynamics shown.

2.1 - Definition 1.

Let $\mathbf{a}_{sm} = (A, F_A, C, g)$ be a self-modifying automaton characterised by :

$$e_{n+1} = f_n(e_n) \quad (1)$$

$$f_{n+1} = C(e_{n+1}) \quad (2)$$

The trajectory which corresponds to the successive states of \mathbf{a}_{sm} will be denoted :

$$T_0 = \{(f_0, e_0), \dots, (f_{t-1}, e_{t-1}), (f_t, e_t), \dots, (f_{t+p-1}, e_{t+p-1}), (f_t, e_t), \dots\} \quad (7)$$

$$= Tt_0 \cup Tp_0 = \{(f_0, e_0), \dots, (f_{t-1}, e_{t-1})\} \cup \{(f_t, e_t), \dots, (f_{t+p-1}, e_{t+p-1}), (f_t, e_t), \dots\} \quad (8)$$

where Tt_0 and Tp_0 are the subset of transient states and the subset of states in limit cycle of \mathbf{a}_{sm} respectively.

The **self-programming machine** \mathbf{m}_{sp} associated to \mathbf{a}_{sm} is a dynamic system (e_t, g_t) . \mathbf{m}_{sp} is characterized by its input at time t $input(t)$ and a labeled oriented graph $\mathbf{G}_t = [V_t, arc_t, label_t]$, where $V_t \subset A$ is the set of vertices, $arc_t \subset A^2$ the set of arcs and $label_t : V_t \rightarrow F_A$ the vertices labelling of \mathbf{G}_t , defined as follows :

Let us denote $(e_0, g_0) = (e_0, f_0)$ the initial state of \mathbf{m}_{sp} , where (e_0, f_0) is some state on a trajectory of \mathbf{a}_{sm} . Then we define :

$$input(0) = e_0$$

$$V_0 = \{e_0, e_1, \dots, e_{t-1}, e_t, \dots, e_{t+p-1}\} \quad (9)$$

$$= Vt_0 \cup Vp_0 = \{e_0, e_1, \dots, e_{t-1}\} \cup \{e_t, \dots, e_{t+p-1}\} \quad (10)$$

$T_0 = \{(f_0, e_0), \dots, (f_{t-1}, e_{t-1}), (f_t, e_t), \dots, (f_{t+p-1}, e_{t+p-1}), (f_t, e_t), \dots\}$ is the trajectory of \mathbf{a}_{sm} starting at (e_0, f_0)

$$arc_0 = \{(e_0, e_1), \dots, (e_{t-1}, e_t), (e_t, e_{t+1}), \dots, (e_{t+p-2}, e_{t+p-1})\} \quad (11)$$

$$= arct_0 \cup arcp_0 = \{(e_0, e_1), \dots, (e_{t-1}, e_t)\} \cup \{(e_t, e_{t+1}), \dots, (e_{t+p-2}, e_{t+p-1})\} \quad (12)$$

$$label_0(e_i) = f_i, \quad \forall e_i \in V_i$$

We thus have : $\forall (e_n, e_{n+1}) \in arc_0 : e_{n+1} = [label_0(e_n)](e_n)$

At time $t+1$, define (e_{t+1}, g_{t+1}) as follows. If :

$$input(t) = e_n \quad (13)$$

$$V_t = \{e_j, \dots, e_n, e_k, e_l, \dots\} \quad (14)$$

$$arc_t = \{(e_j, e_j), \dots, (e_n, e_k), (e_k, e_l), \dots\}$$

$$label_t(e_i) = g_i, \forall e_i \in V_t \text{ with} \quad (15)$$

$$e_k = [label_t(e_n)](e_n) = g_n(e_n) \text{ and } e_l = [label_t(e_k)](e_k) = g_k(e_k) \quad (16)$$

then:

$$input(t+1) = e_l \quad (17)$$

$$V_{t+1} = \{e_j, \dots, e_n, e_l, \dots\} \quad (18)$$

$$arc_{t+1} = \{(e_j, e_j), \dots, (e_n, e_l), \dots\}$$

$$label_{t+1}(e_i) = label_t(e_i), \forall i \neq n$$

$$\text{and } label_{t+1}(e_n) = label_t(e_k) \circ label_t(e_n) \quad (19)$$

Figure 2-1 shows how, every time machine \mathbf{m}_{sp} passes through a succession of 2 states, \mathbf{m}_{sp} memorizes the direct transition from the first state to the last one. Index t measures the number of times \mathbf{m}_{sp} passes through a given state having the same input value. Progressively, \mathbf{m}_{sp} "skips" more and more states, thus obviously decreasing the length p of the original limit cycle of \mathbf{m}_{sm} until it remains an unique state called the self-referential state.

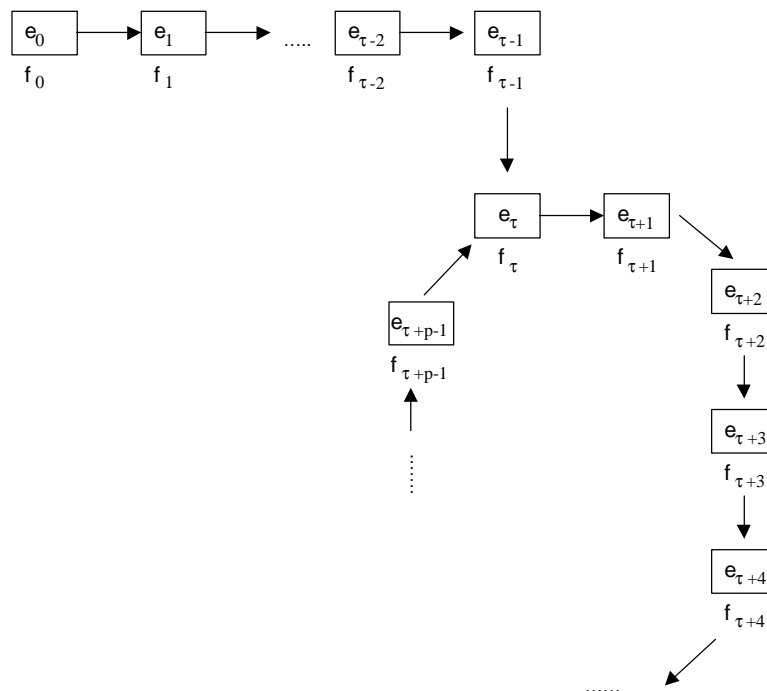


Figure 2-1 : Graph G_0 - Initial graph of the \mathbf{m}_{sp} .

Figure 2-1 : Initial graph of self-programming machine \mathbf{m}_{sp} associated to self-modifying automaton \mathbf{a}_{sm} .

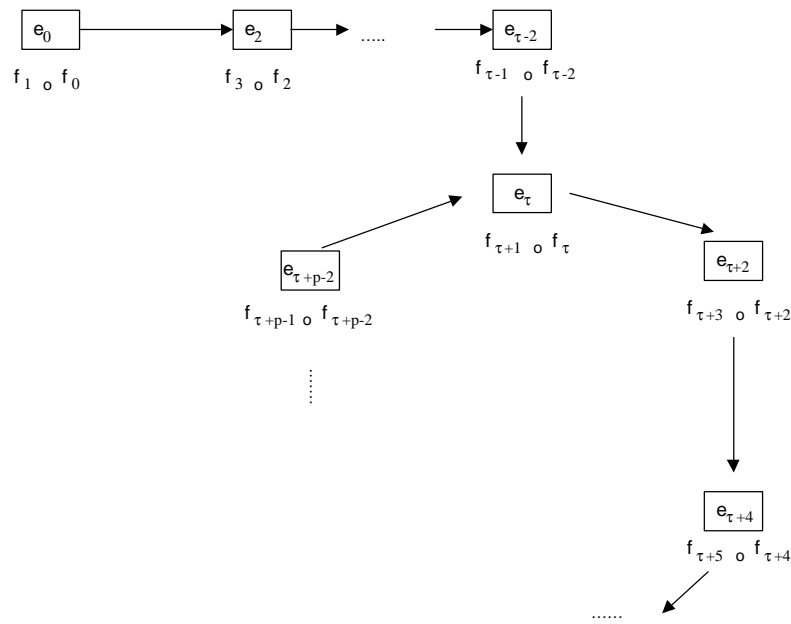


Figure 2-2 : Graph G - When the m_{sp} reaches for the second times the vertex e_t .

Figure 2-2 : Graph of self-programming machine m_{sp} when it has reached for the second times the vertex e_t . (for t and p even).

2.2 - Definition 2

Let $V = \{v_0, v_1, \dots, v_k\}$ be a set of objects and let $\mathbf{j} = \{\mathbf{j}_{v_0}, \mathbf{j}_{v_1}, \dots, \mathbf{j}_{v_k}\}$ be a set of functions such as $\forall i, \mathbf{j}_{v_i} : V \mapsto V$, the pair (\mathbf{j}_{v_n}, v_n) is said self-referential iff $\boxed{\mathbf{j}_{v_n}(v_n) = v_n}$.

2.3 - Theorem 1.

Let us compose m times the function \mathbf{j}_{v_n} , we obtain $\mathbf{j}_{v_n} \circ \mathbf{j}_{v_n} \circ \dots \circ \mathbf{j}_{v_n}$.

The pair $(\mathbf{j}_{v_n} \circ \mathbf{j}_{v_n} \circ \dots \circ \mathbf{j}_{v_n}, v_n)$ is self-referential.

Proof :

Substitute $\mathbf{j}_{v_n}(v_n)$ $m-1$ times by (v_n) in the expression $\mathbf{j}_{v_n} \circ \mathbf{j}_{v_n} \circ \dots \circ \mathbf{j}_{v_n}(v_n)$

2.4 - Theorem 2

The successor of a self-referential state is also a self referential state.

$input(t) = e_n$ and $[label_t(e_n)](e_n) = g_n(e_n) = e_n$ define the self referential state (g_n, e_n) .

The equality (19) allows us to write : $label_{t+1}(e_n) = label_t(e_n) \circ label_t(e_n) = g_n \circ g_n$, therefore $input(t+1) = g_n \circ g_n(e_n) = e_n$ and the successor of the state (g_n, e_n) is $([label_{t+1}(e_n)], e_n) = (g_n \circ g_n, e_n)$ which is self-referential according to the theorem 1.

Remark : A self-referential state and its successor have the same vertex and therefore if a vertex of V_t is self-referential, then $|V_{t+1}| = |V_t|$ and the vertex corresponding to the self-referential state is the last one.

2.5 - Definition 3.

Let \mathbf{a}_{sm} the self-modifying automaton associated to the self-programming machine \mathbf{m}_{sp} and its trajectory $T_0 = \{(f_0, e_0), \dots, (f_{t-1}, e_{t-1}), (f_t, e_t), \dots, (f_{t+p-1}, e_{t+p-1}), (f_t, e_t), \dots\}$. (7)

In the initial graph of the \mathbf{m}_{sp} , $V_0 = \{e_0, \dots, e_{t-1}, e_t, \dots, e_{t+p-1}\} = Vt_0 \cup Vp_0$, (10)

V_0 corresponds to vertices of the trajectory of the \mathbf{a}_{sm} and Vt_0 and Vp_0 correspond respectively to the vertices of the transient states and the states in p-cycles of the \mathbf{a}_{sm} . $|V_0| = |Vt_0| + |Vp_0| = t + p$, t being the number of transient states and p the number of states of the p-cycle of the \mathbf{a}_{sm} .

2.6 - Theorem 3.

\mathbf{m}_{sp} stabilise only building the self-referential objects : $(g_n, e_n) = ([label_t(e_n)], e_n)$

At each time, the number of vertices of Vp decreases from one : $|Vp_{t+1}| = |Vp_t| - 1$ (20)

Therefore, there is some t for which $|Vp_t| = 2$

For any vertex e_i at the time t, corresponds $label_t(e_i) = g_i$ and therefore the state (g_i, e_i)

With $input(t) = e_n$, we have two possibilities : $Vp_t = \{e_n, e_k\}, Vp_t = \{e_n, e_n\}$.

For $Vp_t = \{e_n, e_k\}$, we have two possibilities of arcs :

$$arc_t = \{(e_n, e_k), (e_k, e_k)\} \quad (21)$$

$$arc_t = \{(e_n, e_k), (e_k, e_n)\}. \quad (22)$$

For $Vp_t = \{e_n, e_n\}$, we have two possibilities of arcs :

$$arc_t = \{(e_n, e_n), (e_n, e_k)\} \quad (23)$$

$$arc_t = \{(e_n, e_n), (e_n, e_n)\}. \quad (24)$$

First case :

$$Vp_t = \{e_n, e_k\}, arc_t = \{(e_n, e_k), (e_k, e_k)\}, input(t) = e_n$$

$$e_k = [label_t(e_n)](e_n) = g_n(e_n) \text{ and } e_k = [label_t(e_k)](e_k) = g_k(e_k)$$

and $label_{t+1}(e_n) = label_t(e_k) \circ label_t(e_n)$, therefore :

$$Vp_{t+1} = \{e_k\}, arc_{t+1} = \{(e_k, e_k)\} \text{ and } input(t+1) = e_k$$

$$\text{therefore } e_k = [label_{t+1}(e_k)](e_k) = g_k(e_k) \text{ and } e_k = [label_{t+1}(e_k)](e_k) = g_k(e_k),$$

and $label_{t+2}(e_k) = label_{t+1}(e_k) \circ label_{t+1}(e_k) = g_k \circ g_k$, therefore :

$Vp_{t+2} = \{e_k\}$, $arc_{t+2} = \{(e_k, e_k)\}$, $input(t+2) = e_k$ and $e_k = g_k \circ g_k(e_k)$, therefore the object $([label_{t+2}(e_k)], e_k)$ is self-referential.

Second case : (22)

$$Vp_t = \{e_n, e_k\}, arc_t = \{(e_n, e_k), (e_k, e_n)\}, input(t) = e_n$$

$$e_k = [label_t(e_n)](e_n) = g_n(e_n) \text{ and } e_n = [label_t(e_k)](e_k) = g_k(e_k)$$

and $label_{t+1}(e_n) = label_t(e_k) \circ label_t(e_n) = g_k \circ g_n$, therefore :

$$Vp_{t+1} = \{e_n\}, arc_{t+1} = \{(e_n, e_n)\} \text{ and } input(t+1) = e_n, \text{ therefore } e_n = g_k \circ g_n(e_n) \text{ and}$$

The object $([label_{t+2}(e_n)], e_n)$ is self-referential.

Third case : (23)

$$Vp_t = \{e_n, e_n\}, arc_t = \{(e_n, e_n), (e_n, e_k)\}, input(t) = e_n$$

This case is not possible because

$$[label_t(e_n)](e_n) = g_n(e_n) \text{ can't be simultaneously equal to } e_n \text{ and } e_k .$$

Fourth case : (24)

$$Vp_t = \{e_n, e_n\}, arc_t = \{(e_n, e_n), (e_n, e_n)\}, input(t) = e_n$$

$$e_n = [label_t(e_n)](e_n) = g_n(e_n) \text{ and } e_n = [label_t(e_n)](e_n) = g_n(e_n)$$

and $label_{t+1}(e_n) = label_t(e_n) \circ label_t(e_n) = g_n \circ g_n$, therefore

$$Vp_{t+1} = \{e_n\}, arc_{t+1} = \{(e_n, e_n)\} \text{ and } input(t+1) = e_n, \text{ therefore } e_n = g_n \circ g_n(e_n) \text{ and}$$

The object $([label_{t+1}(e_n)], e_n)$ is self-referential.

Therefore, in all the possible cases (i.e. 1,2 and 4), \mathbf{m}_{sp} stabilises in a self-referential way, all their states have one and only one predecessor and all the states are different.

2.7 - Theorem 4.

if t is even, \mathbf{m}_{sp} stabilises after $(t/2) + p - 1$ times and it remains $(t/2)$ vertices.
If t is odd, \mathbf{m}_{sp} stabilises after $(t+1)/2 + p - 2$ times and it remains $(t+1)/2$ vertices.

Proof :

At each time, the number of vertices of the graph of \mathbf{m}_{sp} decreases from one until the \mathbf{m}_{sp} reaches a self-referential state :

- According to the equalities $V_t = \{e_j, \dots, e_n, e_k, e_l, \dots\}$ (14) and $V_{t+1} = \{e_j, \dots, e_n, e_l, \dots\}$ (18), we deduce : $|V_{t+1}| = |V_t| - 1$ (20)
- According to the theorem 2, when the \mathbf{m}_{sp} has reached a self-referential state, $|V_{t+1}| = |V_t|$.

The \mathbf{m}_{sp} goes through the t transient vertices Vt_0 only once and it goes through the vertices Vp_0 a sufficient number of times in order to reach a self-referential state.

- If t is even, $V_{t/2} = \{e_0, e_2, \dots, e_{t-2}, e_t, \dots, e_{t+p-1}\}$, $input(t/2) = e_t$ and $|V_{t/2}| = (t/2) + p$, therefore $|V_{(t/2)+p-1}| = (t/2) + 1$
- If t is odd, $V_{(t+1)/2} = \{e_0, e_2, \dots, e_{t-1}, e_{t+1}, \dots, e_{t+p-1}\}$, $input((t+1)/2) = e_{t+1}$ and $|V_{(t+1)/2}| = (t+1)/2 + p - 1$, therefore $|V_{(t+1)/2+p-2}| = (t+1)/2$.

2.8 - Lemma.

Definitions :

- In a graph, a Eulerian tour Θ is a closed trail which contains each arc exactly once.
- Let us call $TP_0 = \{(f_t, e_t), \dots, (f_{t+p-1}, e_{t+p-1}), (f_t, e_t), \dots\}$ the trajectory of \mathbf{a}_{sm} having no transient states.
- Let us associate \mathbf{m}_{sp} having TP_0 as initial conditions.
- Let us call TP_∞ the trajectory of \mathbf{m}_{sp} whenever $(f_{t+i}, e_{t+i}) \in TP_\infty$, (f_{t+i}, e_{t+i}) being a self-referential state.
- Let Gp_∞ and Vp_∞ be respectively the graph and the set of vertices associated to TP_∞ .

Lemma :

$$Vp_\infty = \Theta_1 \cup \dots \cup \Theta_k \cup \Theta_{k+1} \cup \dots \cup \Theta_i.$$

- Let us suppose $input(t) = e_{t+a}$ corresponding to the vertex e_{t+a} . Sooner or later, the \mathbf{m}_{sp} will reach a vertex e_{t+b} it has already gone through. All the arcs of the closed trail $L_r = \{e_{t+a}, \dots, e_{t+b}, \dots, e_{t+b-1}, e_{t+b}\}$ are distinct.
- Let us characterise this Eulerian tour by the set of distinct vertices $\Theta_r = \{e_{t+a}, \dots, e_{t+b}, \dots, e_{t+b-1}\} = L_r - \{e_{t+b}\}$. $|\Theta_r| = |L_r| - 1$ is the number of Eulerian tour and $\forall i, j : \Theta_i \cap \Theta_j = \emptyset$.
- Let us call Θ_1 the first Eulerian tour.
- According to the theorem 3, \mathbf{m}_{sp} stabilises in a self-referential state (f_{t+i}, e_{t+i}) and the successor of the vertex e_{t+i} is e_{t+i} , therefore the last loop is $L_i = \{e_{t+i}, e_{t+i}\}$ and $\Theta_i = \{e_{t+i}\}$.

All these considerations allow us to write :

$$Vp_\infty = \Theta_1 \cup \dots \cup \Theta_k \cup \Theta_{k+1} \cup \dots \cup \Theta_i.$$

2.9 - Computation of i of the index $t+i$ of a self-referential state

$$(g_{t+i}, e_{t+i}) = ([label_{t+i}(e_{t+i})], e_{t+i}).$$

- Let us call $\Theta_k = \{e_a, e_{a+1}, e_{a+2}, \dots, e_{a+p_k-2}, e_{a+p_k-1}\}$ a Eulerian tour Θ_k of Vp_∞ and let be $p_k = |E_k|$ the number of the arcs of Θ_k .
 - If p_k is even, $\Theta_{k+1} = \{e_a, e_{a+2}, \dots, e_{a+p_k-2}\}$ and $p_{k+1} = p_k - (p_k/2) = p_k/2$.
 - if p_k is odd, $\Theta_{k+1} = \{e_{a+2}, \dots, e_{a+p_k-3}, e_{a+p_k-1}\}$ and $p_{k+1} = p_k - (p_k + 1)/2 = (p_k - 1)/2$.

- Let us denote d_k the difference between two successive indices of Θ_k , $d_{k+1} = 2d_k$. $d_0 = 1$, therefore $d_k = 2^k$.
 - If p_k is even, the first element of Θ_k and Θ_{k+1} are the same and $i_{k+1} = i_k$.
 - If p_k is odd, the first element of Θ_k and Θ_{k+1} differs from d_{k+1} and $i_{k+1} = i_k + 2^{k+1}$.

We can sum up all these results by the following algorithm :

Initial conditions : $p_0 = p$, $i_0 = 0$

Until $p_i = 1$

If p_k is even do :

$$p_{k+1} = p_k / 2$$

$$i_{k+1} = i_k$$

If p_k is odd do :

$$p_{k+1} = (p_k - 1) / 2$$

$$i_{k+1} = i_k + 2^{k+1}$$

$i = i_i$ is the result.

Usually, the initial trajectory T_0 of \mathbf{m}_{sp} includes transient states $T\mathbf{t}_0$

$T_0 = T\mathbf{t}_0 \cup Tp_0 = \{(f_0, e_0), \dots, (f_{t-1}, e_{t-1})\} \cup \{(f_t, e_t), \dots, (f_{t+p-1}, e_{t+p-1}), (f_t, e_t), \dots\}$ with $|T\mathbf{t}_0| = t$.

- If t is even, \mathbf{m}_{sp} reaches the first vertex of Vp_0 at time $t/2$: $input(t/2) = e_t$, $V_{t/2} = \{e_0, e_2, \dots, e_{t-2}, e_t, \dots, e_{t+p-1}\} = \{e_0, e_2, \dots, e_{t-2}\} \cup \{e_t, \dots, e_{t+p-1}\} = V\mathbf{t}_{t/2} \cup Vp_0$ and we take $p_0 = p$.
- If t is odd, \mathbf{m}_{sp} reaches the second vertex of Vp_0 at time $(t+1)/2$: $input((t+1)/2) = e_{t+1}$ $V_{(t+1)/2} = \{e_0, e_2, \dots, e_{t-1}, e_{t+1}, \dots, e_{t+p-1}\} = \{e_0, e_2, \dots, e_{t-1}\} \cup \{e_{t+1}, \dots, e_{t+p-1}\} = V\mathbf{t}_{(t-1)/2} \cup (Vp_0 - \{e_t\})$.
And we take $p_0 = p - 1$ and $i = i_i + 1$.

Remark: $i = i_i$ can be computed in a straightforward way : One subtracts the greater power of two less than p . Let 2^n be this power. $i = \mathbf{d}(p) = 2(p - 2^n)$.

The figure 2-3 gives the values $i = \mathbf{d}(p)$ of distribution \mathbf{d} for $p \in [1, \dots, 62]$.

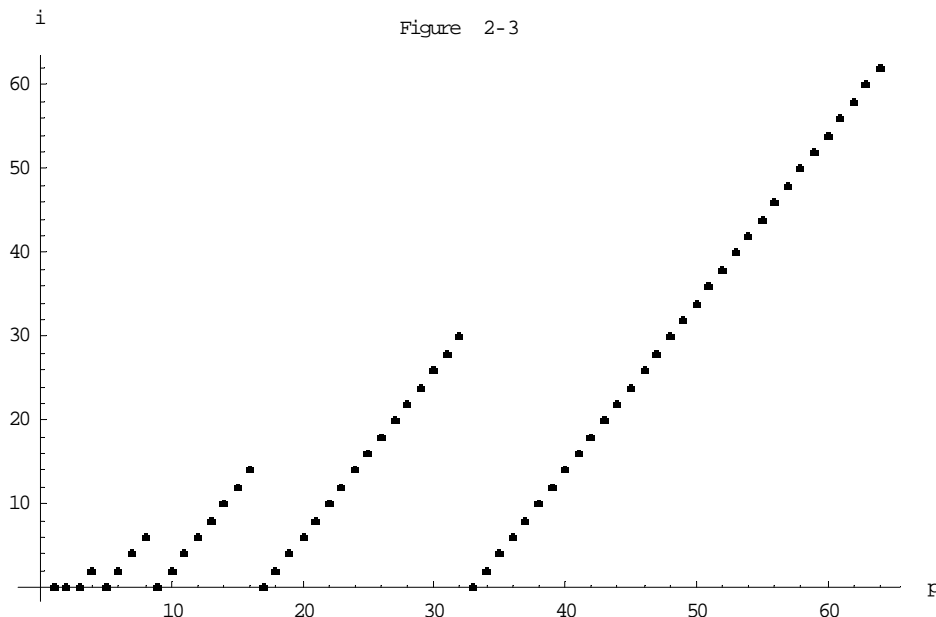


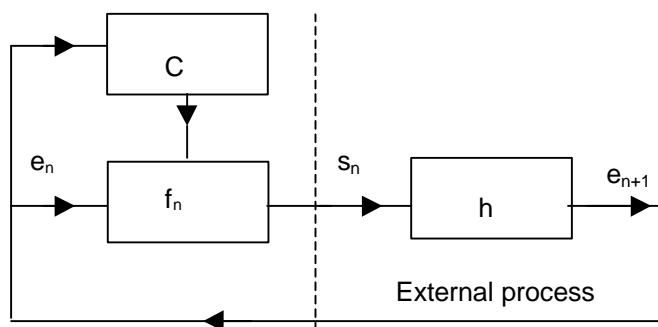
Figure 2-3.

Self-programming machine connected with an external process.

- A \mathbf{m}_{sp} connected to itself (1) : $e_{n+1} = f_n(e_n)$ builds self-referential states (g_n, e_n) such as $g_n(e_n) = e_n$. The interest of this is restricted to perception mechanism models. A \mathbf{m}_{sp} connected to a deterministic process as function h would be of greater interest.

3 . 1-First case :

The \mathbf{m}_{sp} is connected with an external process modelled by function h as shown in figure 3-1 :



- $input(t) = e_n$ (13)

$s_n = f_n(e_n), e_{n+1} = h(s_n),$ therefore

- $label_i(e_n) = h \circ f_n$ (15)

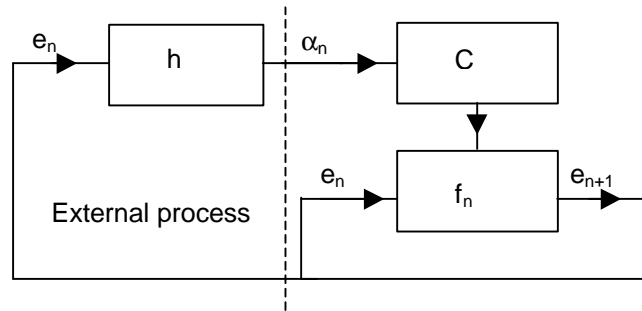
- $e_{n+1} = [label_t(e_n)](e_n) = h \circ f_n(e_n)$ and
 $e_{n+2} = [label_t(e_{n+1})](e_{n+1}) = h \circ f_{n+1}(e_{n+1})$ (16)

- $label_{t+1}(e_n) = label_t(e_{n+1}) \circ label_t(e_n) = h \circ f_{n+1} \circ h \circ f_n$ (19)

These equalities show an impossibility because the values of $h(x)$ are known but not the function h itself. Therefore the expression $label_t(e_n) = h \circ f_n$ cannot be calculated and the \mathbf{m}_{sp} connected on a external function h as displayed in figure 3-1 does not work.

3.2 - Second case :

The \mathbf{m}_{sp} is connected with a external process modelled by a function h as shown in figure 3-2 :



- $input(t) = e_n$ (13)

- $label_t(h(e_n)) = f_n$ (15)

- $e_{n+1} = [label_t(h(e_n))](e_n) = f_n(e_n)$ and
 $e_{n+2} = [label_t(h(e_{n+1}))](e_{n+1}) = f_{n+1}(e_{n+1})$ (16)

- $label_{t+1}(h(e_n)) = label_t(h(e_{n+1})) \circ label_t(h(e_n)) = f_{n+1} \circ f_n$ (19)

The function $f_n = label_t(h(e_n)) = label_t(\mathbf{a}_n)$ can be considered as a function of two variables \mathbf{a}_n and e_n . Let us write $f_n(e_n) = f(\mathbf{a}_n, e_n)$.

- $e_{n+1} = f(\mathbf{a}_n, e_n)$ (22)

- $\mathbf{a}_{n+1} = h(e_{n+1})$ therefore $\mathbf{a}_{n+1} = h \circ f(\mathbf{a}_n, e_n)$ (23)

If we substitute in the equality $f(\mathbf{a}_{n+1}, e_{n+1}) = e_{n+2}$ the right terms of the equalities (22) and (23), we obtain :

- $e_{n+2} = f \circ [f(\mathbf{a}_n, e_n), h \circ f(\mathbf{a}_n, e_n)] = [label_{t+1}(h(e_n))](e_n)$ (24)

4 - Self-programming machines functioning with no calculation :

One can define functions $f_n = label_t(e_n)$ by stating a rule that specifies the value $f_n(e_n)$ of f_n for each $e_n \in \mathbb{N}$. The formalism defined in definition 1 of the paragraph 2.1 is thus modified :

- $label_t(e_i) = e_{i+1}, \forall e_i \in V_i$ instead of (15) and (16) (25)
- $label_t(e_{i+1}) = e_{i+2}$

$$label_{t+1}(e_i) = e_{i+2} \quad \text{instead of (19)} \quad (26)$$

5 - Asynchronous network of self-programming machines.

Consider an asynchronous network of self programming machines for which each machine has its local time (Zelonka). Such a network can be simulated by choosing at random the next self programming machine which will be activated. Therefore we have to study a self programming machine when its input value is randomly chosen from an uniform distribution.

5-1 Definition 1

Let $\mathbf{a}_{sm} = (A, F_A, C, g)$ be a self-modifying automaton characterised by :

$$e_{n+1} = f_n(e_n) \quad (1)$$

$$f_{n+1} = C[e_{n+1}] \quad (2)$$

The trajectory which corresponds to the successive states of \mathbf{a}_{sm} will be denoted :

$$T_0 = \{(f_0, e_0), \dots, (f_{t-1}, e_{t-1}), (f_t, e_t), \dots, (f_{t+p-1}, e_{t+p-1}), (f_t, e_t), \dots\} \quad (7)$$

$$= Tt_0 \cup Tp_0 = \{(f_0, e_0), \dots, (f_{t-1}, e_{t-1})\} \cup \{(f_t, e_t), \dots, (f_{t+p-1}, e_{t+p-1}), (f_t, e_t), \dots\} \quad (8)$$

where Tt_0 and Tp_0 are respectively the subset of transient states and the subset of states in limit cycle of \mathbf{a}_{sm} .

The **asynchronous self-programming machine** \mathbf{m}_{asp} associated to \mathbf{a}_{sm} is a dynamic system (e_t, g_t) . \mathbf{m}_{asp} is characterized by its input at time t $input(t) = e_t$ randomly chosen and a labelled oriented graph $\mathbf{G}_t = [V_t, arc_t, label_t]$, where $V_t \subset A$ is the set of vertices, $arc_t \subset A^2$ the set of arcs and $label_t : V_t \rightarrow F_A$ the vertices labelling of \mathbf{G}_t , defined as follows :

Let us denote $(e_0, g_0) = (e_0, f_0)$ the state of \mathbf{m}_{asp} corresponding to the initial state of the \mathbf{a}_{sm} .

Then we define :

$$V_0 = \{e_0, e_1, \dots, e_{t-1}, e_t, \dots, e_{t+p-1}\} \quad (9)$$

$$= Vt_0 \cup Vp_0 = \{e_0, e_1, \dots, e_{t-1}\} \cup \{e_t, \dots, e_{t+p-1}\} \quad (10)$$

$T_0 = \{(f_0, e_0), \dots, (f_{t-1}, e_{t-1}), (f_t, e_t), \dots, (f_{t+p-1}, e_{t+p-1}), (f_t, e_t), \dots\}$ is the trajectory of \mathbf{a}_{sm} starting at (e_0, f_0)

$$arc_0 = \{(e_0, e_1), \dots, (e_{t-1}, e_t), (e_t, e_{t+1}), \dots, (e_{t+p-2}, e_{t+p-1})\} \quad (11)$$

$$= arct_0 \cup arcp_0 = \{(e_0, e_1), \dots, (e_{t-1}, e_t)\} \cup \{(e_t, e_{t+1}), \dots, (e_{t+p-2}, e_{t+p-1})\} \quad (12)$$

$$label_0(e_i) = f_i, \quad \forall e_i \in V_i$$

We thus have : $\forall (e_n, e_{n+1}) \in arc_0 : e_{n+1} = [label_0(e_n)](e_n)$

At time $t+1$, define (e_{t+1}, g_{t+1}) as follows. If :

$$input(t) = e_n \quad (13)$$

$$V_t = \{e_j, \dots, e_n, e_k, e_l \dots\} \quad (14)$$

$$arc_t = \{(e_j, e_j), \dots, (e_n, e_k), (e_k, e_l) \dots\}$$

$$label_t(e_i) = g_i, \quad \forall e_i \in V_t \quad \text{with} \quad (15)$$

$$e_k = [label_t(e_n)](e_n) = g_n(e_n) \quad \text{and} \quad e_l = [label_t(e_k)](e_k) = g_k(e_k) \quad (16)$$

then:

$$input(t+1) = e_l \quad (17)$$

$$V_{t+1} = \{e_j, \dots, e_n, e_k, e_l \dots\} = V_t \quad (20)$$

$$arc_{t+1} = \{(e_j, e_j), \dots, (e_n, e_l), (e_k, e_l) \dots\} \quad (21)$$

$$label_{t+1}(e_i) = label_t(e_i), \quad \forall i \neq n$$

$$\text{and } label_{t+1}(e_n) = label_t(e_k) \circ label_t(e_n) \quad (19)$$

5-2 Dynamics of asynchronous self-programming machines : qualitative study.

Let's call e_j the successor of the vertex e_i and e_i a predecessor of e_j .

5-2-1 Theorem 1 : A vertex has one and only one successor.

Proof : Due to the definition of function itself :

If $g_i(e_i) = e_j$ and $g_i(e_i) = e_m$ then $e_j = e_m$.

5-2-2 Theorem 2 : A vertex belongs to one and only one cycle.

Proof : Let's suppose a vertex belongs to two cycles. In this case, the vertex has two successors. According to the theorem 1, it is impossible.

For clarity, we write the successor of e_i , $Suc(e_i)$ and the p th successor of e_i , $Suc.Suc \dots Suc(e_i)$ or $Suc^p(e_i)$, and the predecessor of e_i , $Pre(e_i)$.

5-2-3 Definition 1 : A vertex e_i belongs to a p -cycle of \mathbf{G}_t or is p -cyclic, iff $Suc^p(e_i) = e_i$.

5-2-4 Theorem 3 : The successor (e_j) of a p -cyclic vertex (e_i) is a p -cyclic vertex.

$$\text{We have } Suc(e_i) = e_j \quad (27)$$

$$\text{and } Suc^p(e_i) = e_i. \quad (28)$$

Proof : From (28) $Suc^{p-1}.Suc(e_i) = e_i$, (from 26) $Suc^{p-1}(e_j) = e_i$ (29),

(from 29) $Suc.Suc^{p-1}(e_j) = Suc(e_i)$, therefore $Suc^p(e_j) = e_j$.

Therefore a cyclic vertex has at least one predecessor which is cyclic and no predecessor of a transient vertex is cyclic.

Let e_n be the randomly chosen value of the $input(t)$ at time t and the directed graph $\mathbf{G}_t = [V_t, arc_t, label_t]$ defining the asynchronous self-programming machine \mathbf{m}_{asp} at time t .

From the formalism given in the paragraph 5-1, it will be recalled that :

$$\mathbf{V}_{t+1} = \mathbf{V}_t \quad (20) \text{ and}$$

$$arc_t = \{(e_j, e_{j'}), \dots, (e_n, e_k), (e_k, e_l) \dots\} \text{ and}$$

$$arc_{t+1} = \{(e_j, e_{j'}), \dots, (e_n, e_l), (e_k, e_l) \dots\} \quad (21),$$

therefore :

$$\text{In graph } \mathbf{G}_t, \quad Suc(e_n) = e_k, \quad Suc(e_k) = e_l.$$

$$\text{In graph } \mathbf{G}_{t+1}, \quad Suc(e_n) = e_l, \quad Suc(e_k) = e_l.$$

Depending on whether the vertices e_n, e_k and e_l are p-cyclic or transient, we have eight cases to study :

- **Case 1** : e_n, e_k and e_l are p-cyclic vertices of \mathbf{G}_t (**figure 5-1**). Therefore $Suc^2(e_n) = e_l$, $Suc^p(e_l) = e_l$, $Suc^{p-2}.Suc^2(e_n) = Suc^{p-2}(e_l)$ and finally $e_n = Suc^{p-2}(e_l)$

In \mathbf{G}_{t+1} , e_k is no longer the successor of e_n , therefore it is no longer successor of cyclic vertex becomes a transient vertex (from theorem 3).

In \mathbf{G}_{t+1} , $Suc(e_n) = e_l$ and since $e_n = Suc^{p-2}(e_l)$, $e_n = Suc^{p-2}.Suc(e_n) = Suc^{p-1}(e_n)$ and $e_l = Suc.Suc^{p-2}(e_l) = Suc^{p-1}(e_l)$.

e_n and e_l become p-1 cyclic in \mathbf{G}_{t+1} .

Remark :

If in \mathbf{G}_t , $Suc(e_n) = e_n$ and $Suc^2(e_n) = e_n$, corresponding to p=1 (fixed point).

In \mathbf{G}_{t+1} , $Suc(e_n) = e_n$ and $Suc^2(e_n) = e_n$, and the 1-cycle remains an 1-cycle.

- **Case 2** : (e_n p-cyclic, e_k p-cyclic, e_l transient) in graph \mathbf{G}_t
- **Case 3** : (e_n p-cyclic, e_k transient, e_l p-cyclic) in graph \mathbf{G}_t
- **Case 4** : (e_n p-cyclic, e_k transient, e_l transient) in graph \mathbf{G}_t
- **Case 6** : (e_n transient, e_k p-cyclic, e_l transient) in graph \mathbf{G}_t

Cases 2, 3, 4, and 6 are impossible because a transient vertex cannot be the successor of a cyclic vertex (from the theorem 3).

- **Case 5** : (e_n transient, e_k p-cyclic, e_l p-cyclic) in graph \mathbf{G}_t

The predecessors of e_n are the same in graph \mathbf{G}_t and in graph \mathbf{G}_{t+1} , therefore e_n is transient in graph \mathbf{G}_{t+1} . The successor of e_k and e_l remains identical in \mathbf{G}_t and \mathbf{G}_{t+1} , therefore e_k and e_l are p-cyclic in \mathbf{G}_{t+1} .

- Case 7 : (e_n and e_k transient, e_l p-cyclic) in graph \mathbf{G}_t

The predecessors of e_n and the predecessors of e_k are the same in graph \mathbf{G}_t and in graph \mathbf{G}_{t+1} , therefore e_n and e_k are transient in graph \mathbf{G}_{t+1} . The successors e_l remain identical in \mathbf{G}_t and \mathbf{G}_{t+1} , therefore e_l is p-cyclic in \mathbf{G}_{t+1} .

- Case 8 : (e_n, e_k and e_l transient) in graph \mathbf{G}_t (figure 5-2)

The predecessors of e_n are the same in graph \mathbf{G}_t and in graph \mathbf{G}_{t+1} , therefore e_n is transient in graph \mathbf{G}_{t+1} .

In \mathbf{G}_{t+1} e_k has no predecessors and therefore is transient.

In \mathbf{G}_{t+1} e_l has an unique predecessor e_n which is transient and is itself transient.

All these results can be summarised in the following array :

	\mathbf{G}_t e_n	\mathbf{G}_t e_k	\mathbf{G}_t e_l	\mathbf{G}_{t+1} e_n	\mathbf{G}_{t+1} e_k	\mathbf{G}_{t+1} e_l
Case 1	p-cyclic ($p \neq 1$) 1-cycle	p-cyclic	p-cyclic	p-1 cyclic 1-cycle	transient 1-cycle	p-1 cyclic 1-cycle
Case 2	p-cyclic	p-cyclic	transient	<- Impossibility		
Case 3	p-cyclic	transient	p-cyclic	<- Impossibility		
Case 4	p-cyclic	transient	transient	<- Impossibility		
Case 5	transient	p-cyclic	p-cyclic	transient	p-cyclic	p-cyclic
Case 6	transient	p-cyclic	transient	<- Impossibility		
Case 7	transient	transient	p-cyclic	transient	transient	p-cyclic
Case 8	transient	transient	transient	transient	transient	transient

This array can itself be summarised by the following rule : If the randomly chosen input of the \mathbf{m}_{asp} corresponds to a p-cyclic vertex ($p \geq 2$) in graph \mathbf{G}_t , it becomes p-1 cyclic vertex in \mathbf{G}_{t+1} , and its successor, which was p-cyclic in \mathbf{G}_t , becomes transient in \mathbf{G}_{t+1} . If $p = 1$, the vertex corresponding to a fixed point in \mathbf{G}_t remains a fixed point in \mathbf{G}_{t+1} .

In all other cases, the property of the vertices to be p-cyclic or transient in \mathbf{G}_t remains the same in \mathbf{G}_{t+1} .

As the number of vertices is finite, the number of p-cycle is finite and sooner or later all the p-cycle becomes a fixed point.

5-2-6 Theorem 5 :

A synchronous self-programming machines converge only at fixed points.

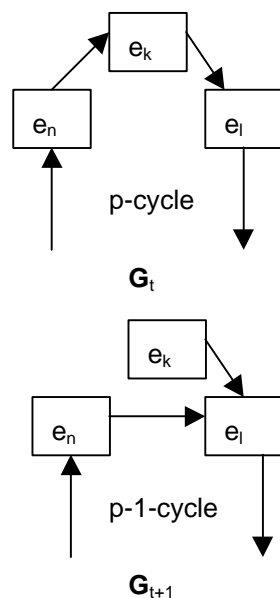


Figure 5-1

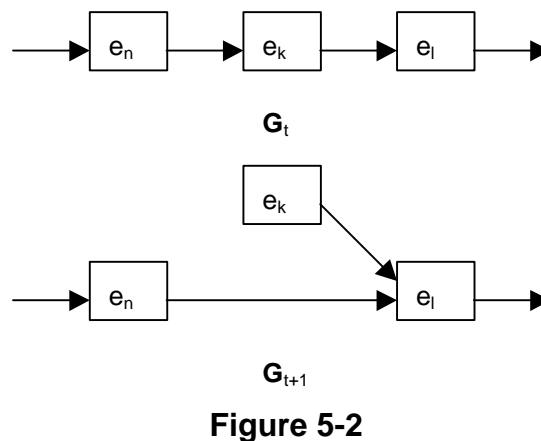


Figure 5-2

5-3 Dynamics of asynchronous self-programming machines : quantitative study.

In paragraph 2-1, it has been shown that every self-programming machine is associated with a self-modifying automaton which constitutes the initial graph of the self-programming machine

Thus we consider the class of all self-modifying automata where the number of vertices is n .

5-3-1 Theorem 1 : The total number of \mathbf{a}_{sm} having n vertices is n^n .

Proof : n^n is the number of functions which apply the set $\{0, \dots, n-1\}$ into itself.

5-3-2 Theorem 2: The number of different p -cycle is $P(n, p) / p$.

$P(n, p) = n! / (n - p)!$ is the number of p -permutations from a n -set.

Proof : The number of polygons with p vertices is $P(n, p)$, but p circular permutations of the vertices correspond to the same p -cycle, therefore we have to divide $P(n, p)$ by p to obtain the number of p -cycles.

5-3-3 Theorem 3 : The number of \mathbf{a}_{sm} having different p -cycle different is $P(n, p) \cdot n^{n-p} / p$.

Proof : The \mathbf{a}_{sm} have n vertices. P vertices correspond to the p -cycles, it remains $n-p$ vertices corresponding to n^{n-p} possibilities, n^{n-p} being the number functions which apply a n -set into a $n-p$ -set.

5-3-4 Theorem 4 : The probability that a vertex belongs to a p-cycle is $\wp(p) = P(n, p) / n^{p+1}$.

Proof : The number of vertices of all the a_{sm} is $n.n^n = n^{n+1}$,

The number of vertices of all the p-cycles is $p.P(n, p).n^{n-p} = P(n, p).n^{n-p}$,

The probability is $P(n, p).n^{n-p} / n^{n+1} = P(n, p) / n^{p+1}$.

5-3-5 Theorem 5 :

The probability that a vertex is transient is $\wp(t) = 1 - \sum_{p=1}^n P(n, p) / n^{p+1}$

A vertex is transient if it is not p-cyclic with $p \in \{1, \dots, n\}$.

The probability is $1 - \sum_{p=1}^n P(n, p) / n^{p+1}$

Simulation of dynamics of m_{asp} .

We will compute the number s of steps necessary for all the m_{asp} converge only at fixed points.

We have defined a random variable and its probability distribution of $n+1$ events : $\wp(t), \wp(1), \wp(2), \dots, \wp(n)$.

The quantity $1/n^{n+1}$ is the probability of chose a given vertex among the n^{n+1} of all the vertices of all the m_{asp} .

Applying the rule described in the paragraph 5.2 :

- Initial conditions : $\wp(t), \wp(1), \wp(2), \dots, \wp(n)$, $s = 0$
 - Until all the p-cyclic vertices are 1-cyclic vertices, repeat :
 - A test giving the random variable RV from the set $\{t, 1, 2, \dots, n\}$
 - If $RV = p \in \{2, \dots, n\}$
 - then replace
 - $\wp(p)$ by $\wp(p) - p/n^{n+1}$
 - $\wp(p-1)$ by $\wp(p-1) + (p-1)/n^{n+1}$
 - $\wp(t)$ by $\wp(t) + 1/n^{n+1}$
 - replace s by $s + 1$
 - else replace s by $s + 1$
 - end

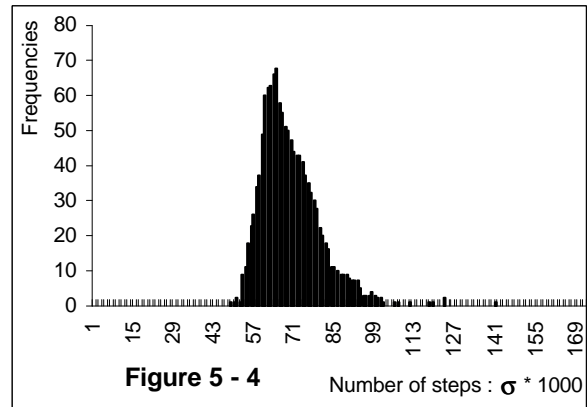
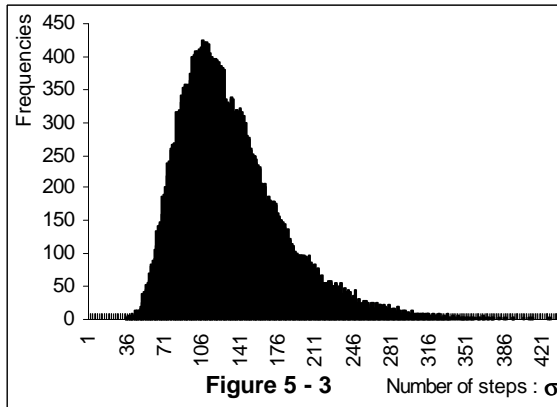
Unfortunately, the complexity of this algorithm is non-polynomial and the convergence of asynchronous self programming automata can only be computed for $n \leq 5$.

On **figures 5-3** and **5-4**, we show bar charts giving the results of 42,000 computations for $n=3$ and 1,500 computations for $n=5$. X-axis represents the number of steps s necessary for all m_{asp} converge only in fixed points and Y-axis represents the frequencies.

Let $E_{\varphi}(\mathbf{s})$ denote the expected number \mathbf{s} : $E_{\varphi}(\mathbf{s}) = \sum_{i=1}^{\infty} i \cdot \varphi(i)$ and k denote the number of computations.

We obtain $\tilde{\mathbf{s}}$, average value of \mathbf{s} : $\tilde{\mathbf{s}} = \frac{E_{\varphi}(\mathbf{s})}{k \cdot n^n}$

For $n=3$ and $n=5$ we obtain respectively $\tilde{\mathbf{s}} = 4.95$ and $\tilde{\mathbf{s}} = 22.26$.



5-4 – Asynchronous self-programming machines connected with an external process.

The \mathbf{m}_{asp} is connected with a external process modelled by a function h as shown in **figure 3-2**.

We take up the formalism of paragraph 5.1.. If h is the external process, $label(e_n)$ is replaced by $label(h(e_n))$ in all the expressions.

Conclusion.

In this paper, we have demonstrated that machines which associate two or more automata which change their internal organisation whenever the external data vary, converge exclusively at fixed points, have the self programming property and give a solution to the biological enigma of the fact that appropriate responses occur in the absence of either a programmer or any human intention.

We have studied several properties of these machines : The dynamics when they are connected to themselves and connected to an external process, machines operating without any calculation and asynchronous self programming machines.

The theory of these machines represents a new contribution to the theory of self replicating machines, to the theory of self reference and put forward a new type of stability, the stability by self replication.

Bibliographical references

- Anderson, J. A. (1972). A simple Neural Network Generating an Interactive Memory. *Mathematical Biosciences* 14: 197-220.
- Ashby, W.R. (1961). Principles of self-organising systems. Proceedings of the western joint computer conference, 255-278.
- Bartlett, S.J. and Suber, P. (1987). *Self Référence*. Martinus Nijhoff Publishers.
- Chauvin, R. (1968). Facteurs de direction et d'excitation au cours de l'accomplissement d'une tâche chez formica polyctena. *Insectes sociaux* 22: 199-206.
- Codd, E.F. (1968). *Cellular automata*. Academic press.
- Derrida, B., Stauffer, D. (1986). Phase transition in two dimensional Kaufmann cellular automata. *Europhysics letters*, 2 (10): 739-745.
- Desbiez, M.O., Bourgeade, P., Boyer, N., de Jaegher, G. and Frachisse, J.M. (1991). Réponse à des signaux mécaniques : Communication inter et intra cellulaires chez les végétaux. *Acta Biotheoretica*. 39, Nos 3 / 4: 289.
- Delmas, A. (1962). *Voies et centres nerveux*. Introduction à la neurologie. Masson.
- Fogelman Soulié, F. (1985). Contribution à une théorie du calcul sur réseaux. Thèse. Grenoble.
- Fukushima, K. (1975). Cognitron : A Self-Organizing Multilayered Neural Network. *Biological Cybernetics* 20: 121-136.
- Gelber, B.(1958). Retention in paramecium aurelia. *Journal of Comparative Physiology*. 51: 10-115. Springer Verlag. Berlin.
- Gonshor, A. and Melvill, J. G., (1976). Extreme vestibulo ocular adaptation induced by prolonged optical reversal of vision. *Journal Physiology*. London. 256: 381-414.
- Hopcroft, J.E. and Ulmann, J.D. (1972). *Introduction to the theory of languages, automata and computation*. Addison Wesley.
- Hopfield, J.J. (1982). Neural Network and Physical System with Emergent Collective Computational Abilities. *Proceedings of the National Academy of Sciences*. 79: 2554-2558.
- Kaufmann, S.A. (1969). Metabolic stability and epigenesis in randomly constructed genetic nets. *Theoretical biology* 22: 437-467.
- Kohonen, Teuvo (1972). *Self-Organization and Associative Memory*. Springer Verlag. Berlin.
- Kréveras, G. (1992). *During a conversation*.
- Lodish H., Baltimore D., Berk A., Zipursky S.L., Matsudaira P., and Darnell J. (1995). *Molecular Cell Biology*. 3rd edition. Scientific American Books.
- Mattson, R.L.. (1959). A Self-Organizing Logical System. Eastern Joint Computer Conference Record, Institute for Research and Education. New-York.
- Maturana, H.R., and Varela F.J. (1980). *Autopoiesis and cognition*. Boston studies in the philosophy of sciences, vol 42. D. Reidel.
- Moulin, J-P. (1992). Modifiable automata, Self-modifying automata. *Acta Biotheoretica*. 40 numéro 2/3: 195.
- Moulin, J-P. (1999). Very simple models, the self-modifying automata and chain of self-modifying automata, can explain self referential properties of living beings. *Acta Biotheoretica*. 47: 353-365.

- Myhill, J. (1970). The abstract Theory of self-reproduction. Essay on cellular automata. University of Illinois Press. Urbana .
- Nagel, E., Newman, J.R., Gödel, K. and Girard, J-Y. (1989). Le théorème de Gödel. Edition du seuil.
- Neumann J. von. Theory of self-reproducing automata. Ed. A.W. Burks. University Of Illinois Press. Urbana.
- Rumelhart, D. D., Hinton, G. E. and William, R. J. (1986). Learning Representations by Back-Propagating Errors. Nature 323: 533-536.
- Thatcher, J.W. (1965). Self-describing Turing machine and self reproducing cellular automata. Essays on cellular automata, 4: 103-131. A.W. Burks editor. University of Illinois Press.
- Widrow, B., and Hoff, M. E. (1960). Adaptive Switching Circuits. Wetscon Convention Record. Institute for Research and Education. New York.
- Zeleny, M. (1981). Autopiesis: A theory of living organization. D. Reidel.
- Zelonka, W. (1987). Notes on finite asynchronous automata. RAIRO, Informatique Thorique et Applications. 21(2) : 99-135.