```
;; Yann Ics <by.cmsc@gmail.com>
;**********************************
;       2013<>2024@COPYLEFT
;       ALL WRONGS RESERVED
;**********************************
; SLIME 2.27
CL-USER> (require package 'NEUROMUSE3)
loaded: OSC
loaded: NEUROMUSE3
NIL
CL-USER> (in-package :N3)
#<PACKAGE "N3">
N3> (version)
3.0.24
NIL
N3>
```

"Any process that approaches instant interrelation of a total field tends to raise itself to the level of conscious awareness, so that computers seem to 'think'. In fact, they are highly specialized at present, and quite lacking in the full process of interrelation that makes for consciousness. Obviously, they can be made to simulate the process of consciousness, just as our electric global networks now begin to simulate the condition of our central nervous system. But a conscious computer would still be one that was an extension of our consciousness."

– *Understanding Media*, [McLuhan, 1964] p. 351.

Demystification

"Intelligence is not what you know, but what you do when you don't know."[1]

"If you understand others you are intelligent.
If you understand yourself you are enlightened."

– *Lao Tse*

What is an Artificial Intelligence ?
And more specifically, what is Intelligence ?
Is it the capacity to solve any kind of situations ?
Is it the adaptation capacity to face any new situations ?
And what kind of situations ?
And what about Artificial ?
Is it something made by humans as in the imitation of nature ?
Or something unreal ?

Statement 1 : 'The *Terminator** syndrome' would seem to combine two pure facts of science fiction, such as the time travel – which is a kind of an extension of the teleportation but within the time domain – and the supremacy of the Artificial Intelligence, notably through the 'technological singularity' – if indeed this may happen** [Damasio, 2010].

* *Terminator* is an American media franchise created by James Cameron and Gale Anne Hurd from the initial movie *The Terminator* released in 1984.
** The same phenomenon happened in the eighties with the spatial conquest. People thought that they would live on the moon in the year 2000 and to colonize the other planets. This did not happen, notably because of the matter of distances in relation to the travel speed and the delay for communication beyond the moon – even if we reach the light speed – to name one of the main issues. It was a bet, and history showed another face. AI is the same kind of gamble or belief. One day people will realize that complexity has its own limit at the computing level – even if one reaches the hypothetic 'technological singularity' – and also that hardware has its own limitations like for instance the operating temperature, the radiation damage, the RAM capacity, the energy requirements, etc. And they will bet on something else, a new scientific hope or threat, like a new god to adore or to fear.

Statement 2 : AI engineers try to impress the audience to attract investment capital. Their naive approach betrays a lack of cross-disciplinary knowledge and probably, overestimates the 'prophetic' power of the programmers to make the ultimate AI that will control everything even itself. It goes without saying that this is for the sake of our supposed welfare and security – *Big Brother* is going to watch you [Orwell, 1949], if it is not already done.

1. The source of this quotation is not mentioned but ascribed to Jean Piaget (1896 – 1980), a Swiss psychologist known for his work on child development.

Statement 3 : AI will remain a tool – even if it is an adaptive tool destined to assist humans in harmony as far as possible – as an extension of ourselves [McLuhan, 1964], and as such, it can be very useful or very dangerous. Then, we have to choose between understanding or undergoing.

Statement 4 : The machine will always need human to maintain code, physical components and security. A bit like any human might need other human to improve himself/herself. Maybe we can imagine another machine doing the maintenance, but at which level will we make the difference between another machine and the machine itself.

Statement 5 : AI tends toward reproduce human intelligence, but with a significant lack of knowledge about brain biology in enactivism terms and its modeling.

Statement 6 : The symbiose between human and machine – i.e. transhumanism – might erase boundaries or at least will make boundaries unclear, but we are far from that mainly for ethical reasons [Wells, 1896]. But who knows ? Some people devoid of scruples might try – if it is not already ongoing – to make it real ...

Statement 7 : AI is more a philosophical reflexion about our own functioning, our own understanding of the world than a modelisable reality.

Statement 8 : Probably the ultimate limitation of the AI is about '*solution ontologique versus solution legislative*' [Klein, 2020], Induction versus Deduction [Demoures et al., 2005], Discrimination ('horizontal' qualitative estimation) versus Discretisation ('vertical' quantitative analysis) and the paradigmatic context [Herzog, 1974].

Statement 9 : *"... if I had access to a human-level system I'd ask it for help in understanding the consequences for humanity of various policies about AI."* [McCarthy, 2007] p. 1181.

Statement 10 : AI is by definition empirical. Like any biological system, all tend to optimise their adaptive capacity. Any emergent behaviour is hence unpredictable insofar as this is dependant of the architecture and the experience of the system, the context and the nature of the situation that the system has to face.

Statement 11 : *"... Si on prend l'I.A. pour ce qu'elle est, ça fait plus d'une décennie qu'elle surpasse l'humain, c'est un peu son but en tant que programme.*
Si on prend l'I.A. pour une simulation de cerveau et d'Intelligence humaine alors on risque de tomber de haut.
Je pense que le domaine de l'I.A. souffre de son nom, car ça

n'a rien à voir avec une 'intelligence' artificielle. C'est juste un genre d'algorithme mais fait de manière assez astucieuse pour rechercher des solutions potables dans un espace de possibilités gigantesques.
Prétendre que ce qu'on arrive à faire faire à une I.A. peut rivaliser avec ce que fait un cerveau humain (dans sa globalité j'entend, pas juste des calculs à la con ou des reconnaissances de formes très spécifiques, etc...) est assez prétentieux. Ça équivaux à penser qu'il suffit d'un ordi pour simuler le cerveau, et qu'en plus notre algorithme est en mesure de le faire...
*Bref ... c'est une forme de foi, de croyance mais rien de très scientifique."**

* Comment of *zoonel* on the article *« L'intelligence artificielle va surpasser l'humain dans 5 ans », d'après Elon Musk*, Patrick Ruiz, published the 05/03/21 on https://intelligence-artificielle.developpez.com/actu/313154/

Contents

List of figures

List of tables

```
N3> (format t "~A~&;~v@{~A~:*~}~2&" *PART-1* 36 #\*)
INTRODUCTION
;************************************

NIL
N3> (format t "~A" *CHAPTER-1.1*)
PRESENTATION
NIL
N3>
```

In 1999, Frédéric Voisin initiates a project called *Neuromuse*[2, 3] based on the artificial neural network, in order to experiment an ‘organic’ interactivity human/machine in a musical context and in real time.

Neuromuse3[4] aims to develop this idea in a holistic way in term of cliques and *tournois*, inspired by the reading of the book *Petite mathématique du cerveau* [Berrou et al., 2012] (and also [Berrou et al., 2011], [Berrou et al., sept 2012] and [Jiang et al., 2012]). Then, the object is to create an associative memory able to store and to retrace learned musical sequences, and to create original sequences from this ‘network of memory’ algorithmically.

The real time interactivity consists to a supervised learning according to previous learning as an acquis and a circumstantial recalling process(es) in terms of interactivity and interrelationship.

The formalisation of *Neuromuse3* is reduced in three main classes defined as NEURON, MLT (superclass of SOM, itself superclass of RNA) and AREA.

The overall mode of operation of *Neuromuse3* is to stimulate one or more SOM in order to activate a region defined by the AREA and to interpret the response, which can be summarised by the diagram on figure 1.

2. http://www.fredvoisin.com/spip.php?article7
3. http://www.neuromuse.org/
4. https://github.com/yannics/N3

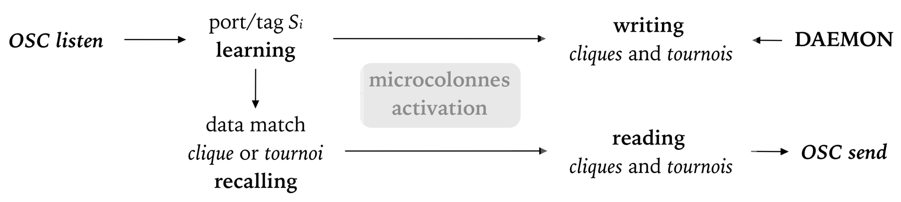


FIGURE 1 – Synoptic diagram of the I/O management.

The AREA network can also be interpreted as it is with its inner connectionist structure.

```
N3> (format t "~A" *CHAPTER-1.2*)
SELF-ORGANIZING-MAP
NIL
N3>
```

The classes SOM, RNA and NEURON are a minimalist adaptation of the code source *Neuromuse*[5] which aims to allow a flexibility to manage parameters. In short, it is question to initiate a Self Organizing Map in terms of learning and activation.

The Self Organizing Map is a concept formalised and modeled in 1980 by Teuvo Kohonen. It is a model of an artificial neural network working by competition for a set of given stimuli. This allows to create a topology for classification, interpolation or displaying multidimensional data.

> The map consists of a regular grid of processing units, 'neurons'. A model of some multidimensional observation, eventually a vector consisting of features, is associated with each unit. The map attempts to represent all the available observations with optimal accuracy using a restricted set of models. At the same time the models become ordered on the grid so that similar models are close to each other and dissimilar models far from each other.
>
> Fitting of the model vectors is usually carried out by a sequential regression process, where $t = 1, 2, ...$ is the step index. For each sample $x(t)$, first the winner index c (best match) is identified by the condition :

$$\forall i, ||x(t) - m_c(t)|| \leq ||x(t) - m_i(t)||$$

5. https://github.com/FredVoisin/neuromuse

After that, all model vectors or a subset of them that belong to nodes centered around node $c = c(x)$ are updated as :

$$m_i(t+1) = m_i(t) + h_{c(x),i}(x(t) - m_i(t))$$

Here $h_{c(x),i}$ is the 'neighborhood function', a decreasing function of the distance between the ith and cth nodes on the map grid. This regression is usually reiterated over the available samples. [6]

Here are some guidelines about the available parameters in N3.

Functions required for the class SOM as slots :

— The map function – slot `carte` – to initiate the neurons of the SOM.

— The neighborhood function – slot `voisinage` – which defines the type of influence that the winning neuron will have on its neighbors allowing to update the network during the learning process (usually a gaussian or a 'mexican hat' function).

— The proximity function to define the type of relationship between two neurons – slot `distance-in` – or between a set of stimuli and its response – slot `distance-out` – when the SOM is activated in order to set and to minimalize errors for the winning neuron selection.

The SOM initiation is done either with the defmethod `init-som` or simply with the N3 function `create-mlt` with as arguments the name of the SOM, the number of stimuli and the number of neurons in the map.

The activation consists to compute for each neuron the error defined by $||x(t) - m_i(t)||$.

Note that the data input has to be adjusted such as each component has a value between 0 and 1 in order to match the range of the output as weight values during initialisation.

The learning step refers to the class RNA according to the slots :

— `learning-rate` as the maximum correction value applied to the winning neuron ;

— `radius` as influence of the winning neuron on its neighbors.

See Table 1 concerning functions which can be set in a SOM as described previously.

See also Table 2 on page 14 for all parameters and properties.

6. `http://www.cis.hut.fi/research/som-research/som.shtml`

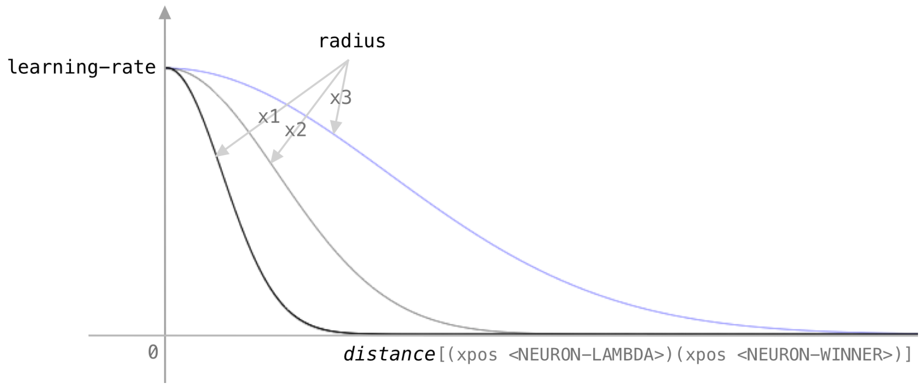


FIGURE 2 – The neighborhood function gauss with its parameters.

TABLE 1 – The package N3 provides some basic functions related to the SOM.

Mapping	quadrare, rnd-map
Proximity	euclidean
Neighborhood	gauss, fn-mex
Decreasing	exp-decay

```
N3> (format t "~A" *CHAPTER-1.3*)
TOPOLOGY
NIL
N3>
```

To map a SOM, the main principle is to initiate a learning process with a significative dataset, using the decreasing function exp-decay applied on the learning rate and on the radius of neighbourhood influence in relation with the number of iteration, defined by $\alpha_{(t)} = \alpha_0 e^{(-t/T)}$ with α_0 as the initial value and $T = n/|ln(\alpha_n/\alpha_0)|$ with n as the number of iterations.

See code on the Annex 1 on page 41.

Note that the illustration (c) on figure 3 the modulo considers each neuron as the middle of the map. Thus, a more accurate representation of this mapping takes the form of a 'donut' (i.e. a torus).

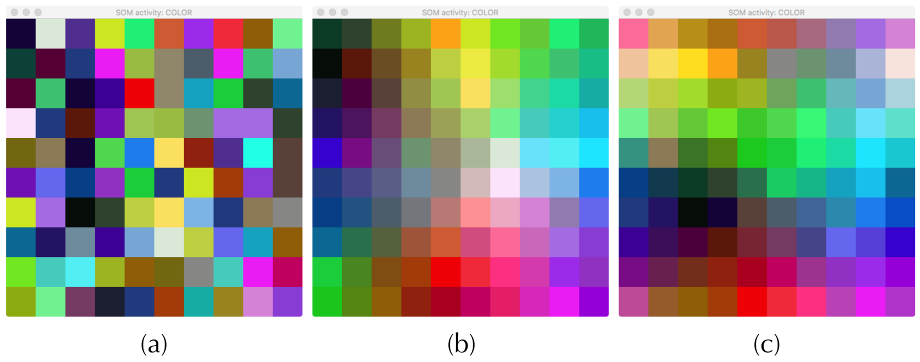


FIGURE 3 – (a) A map of 100 neurons initialised with random colors. (b) The same map after 3000 iteration. (c) Same process than (b) but with modulo.

```
N3> (format t "~A~&;~v@{~A~:*~}~2&" *PART-2* 36 #\*)
PROLEGOMENA
;************************************

NIL
N3> (format t "~A" *CHAPTER-2.1*)
HIERACHICAL CLUSTERING
NIL
N3>
```

In the framework of N3, the discrimination of the SOM in *fanaux* can be carried out by hierarchical ascending classification – noted CAH – according to the Ward's method. The choice of the method remains empirical, but this appears the most appropriate within the framework of an automatic classification.

Here are some lineaments of the CAH and the Ward's method, which are described more precisely in the articles *The automatic classification of quantitative data* [Chavent, 2013] and *Distances between Clustering, Hierarchical Clustering* [Shalizi, 2009].

Note that `dendrogram` generates a newick [7] file by default allowing to display or else the tree with a third-party application.

7. To know more about the Newick tree format see the Web page description by Joseph Felsenstein and by Gary Olsen at :
`http://evolution.genetics.washington.edu/phylip/newicktree.html`

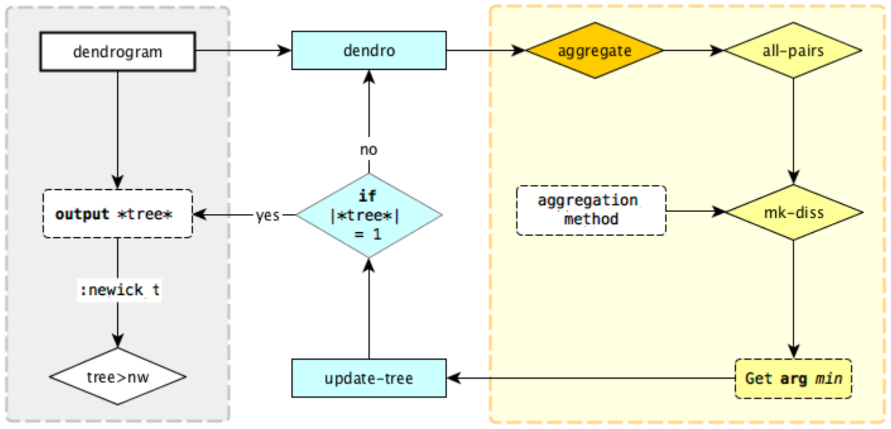


FIGURE 4 – Diagram of the CAH algorithm of dendrogram.

Ward's method

With the Ward's method, the distance between two classes A and B is computed as follow :

$$d(A,B) = \frac{|A||B|}{|A|+|B|} d(g_A, g_B)^2$$

with the mean value :

$$g_{\mathcal{P}} = \frac{1}{|\mathcal{P}|} \sum_{i \in \mathcal{P}} s_i = \overline{x}$$

In some case, the dendrogram hierarchy may have inversions. This can be avoided by applying the following relation :

$$\forall A, B \in H, h(A \cup B) = max(d(A,B), h(A), h(B))$$

Finally, the intraclass inertia is computed for each node as a potential class C such that :

$$I_C = \frac{1}{|C|} \sum_{e \in C} d(e, g_C)^2$$

```
N3> (format t "~A" *CHAPTER-2.2*)
CLIQUE
NIL
N3>
```

In graph theory, the clique – also called complete graph – is a set of nodes such that each node is adjacent to all others. This is expressed in the present context by a set called *macrocolonne* consisting of neurons or group of neurons called *microcolonne* interconnected according to the type of information concerned – see figures 5(a), 5(b) and 5(c).

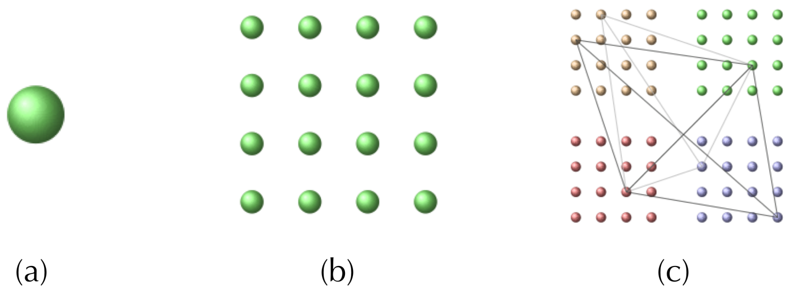


FIGURE 5 – (a) *Microcolonne* : neuron or group of neurons carrying information in terms of activation potential – e.g. a pitch of a sound. (b) *Colonne* : region of *microcolonnes* carrying the same type of information – e.g. a set of pitches as a scale or other. (c) *Macrocolonne* : a region of *colonnes* able to carry 'complete' information called '*infon*' as a clique – e.g. the characteristics of a sound defined by its pitch, intensity, duration and timbre.

On the figure 5(c), we can observe two cliques, one of which is dark gray overlays partially on the second in light gray.

From now on, we name indifferently as synonym *microcolonnes* as *fanaux* (i.e. a list of NEURON), *colonne* as MLT, *macrocolonne* as AREA and *infon* as a clique at the AREA level.

```
N3> (format t "~A" *CHAPTER-2.3*)
TOURNOI
NIL
N3>
```

In graph theory, the *tournoi* is an oriented clique. This currently inscribes a sequence in a temporal configuration, where the *fanaux* – by nature timeless – will then create links oriented in the form of cliques, in other words *tournoi*.

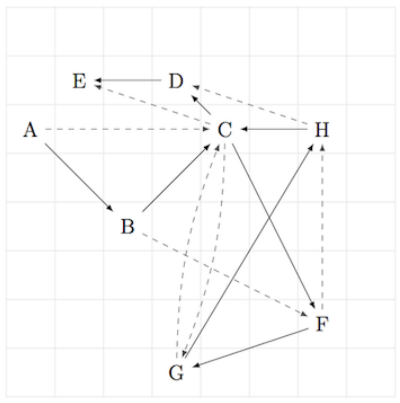


FIGURE 6 – Example of application of *tournois* [Berrou et al., 2012], op. cit., page 117.

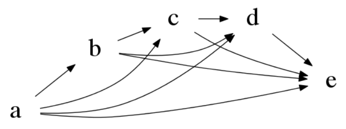


FIGURE 7 – Here is an example of a 5 order *tournoi*. This one corresponds to the followings arcs : ((a b) (a c) (a d) (a e) (b c) (b d) (b e) (c d) (c e) (d e)) and which will be translated simply by : (a b c d e).

To illustrate an application of *tournois*, the figure 6 shows a directed graph, in which we seek to restore the path A-B-C-F-G-H-C-D-E unambiguously – represented with full arrows. Each node symbolizes an *infon*. At first, the problem lies at node C, where two options are possible, namely F or D. A *tournoi* of order 3 solves this problem, covering 2 successive nodes – represented by dashed arrows –, anticipating thus the path from C to F signaled by the arrows from B to F when we come from B.

Thus, the temporal overlap depends on the order[8] of the *tournoi*, i.e. $n - 1$ and will determine the anticipation capacity of the network.

```
N3> (format t "~A~&;~v@{~A~:*~}~&" *PART-3* 36 #\*)
SHORT-TERM MEMORY
;************************************
NIL
N3>
```

For a given sequence as a learning process, the *tournois* are indexed for each *colonne* at the level of the class MLT, whose order will be defined by the slot `cover-value`. The *tournois* (with `remanence`[9] *true*) and the arcs constituting

8. In the graph theory, the order is the number of nodes interconnected. For both the *tournoi* and the clique as a complete graph, the order allows to determinate the number of edges needed such as for a graph of order n, the number of edges is equal to $n(n-1)/2$ – see example figure 7.

9. The remanence means if we consider the *tournois* with the cover-value as they are recorded (i.e. the order of the *tournois*) or if we only consider the arcs.

		Slot	**Input**	**Note**
NEURON		`name`	NEURON	
		`net`	RNA	
		`xpos`	*list*	as coordinates
		`ind`	*integer*	`neurons-list` index
		`synapses-list`	*list*	
	⚙	`temperature`	*number*	
		`erreur`	*number*	
		`output`	*list*	
RNA	↻	`name`	SOM	
	↻	`nbre-neurons`	*integer*	
		`neurons-list`	*list*	
	↻	`nbre-input`	*integer*	
	⚙	`input`	*list*	
	⚙	`radius`	*number*	
	⚙	`learning-rate`	*number*	
		`date-report`	*hash-table*	
		`epoch`	*integer*	
	🕸	`udp-list`	*list*	IP [string] + port(s) [integer]
	🕸	`daemons`	*thread*|*list*	
	🕸	`superdaemon`	*bool*|*object*	
SOM		`neuron-gagnant`	NEURON	
	⚙	`distance-in`	*function*	
	⚙	`distance-out`	*function*	
	⚙	`voisinage`	*function*	
	↻	`carte`	*function*	
	↻	`field`	*number*|*list*	if *list* → ||`field`|| = `topology`
	↻	`topology`	*integer*	
		`ghost`	NEURON	
MLT		`net`	AREA	
	</>	`fanaux-list`	*list*	→ `update-fanaux`
	</>	`cover-value`	*integer*	→ `update-cover-value`
		`mct`	*list*	
		`onset`	*hash-table*	*keys* = `(... first)`
		`fine`	*hash-table*	*keys* = `(last ...)`
		`trns`	*hash-table*	
		`arcs`	*hash-table*	
		`mem-cache`	*object*	
AREA	↻	`name`	AREA	
	↻	`soms-list`	*list*	
		`fanaux-length`	*integer*	
		`current-clique`	*list*	
	⚙	`sensorial-rate`	*number*	
		`arcs`	*hash-table*	
		`date-report`	*hash-table*	
	🕸	`udp-list`	*list*	

Table 2 – List of N3 classes with their respective slots.

the *tournois* (with `remanence` *nil*) will be retained in a separate hash table associated with each SOM.

Note that `cover-value` must imperatively be an integer and greater than or equal to 3 (since a *tournoi* is defined only from 3 nodes).

The *microcolonnes* of each SOM are listed and indexed according to their symbolic representation – i.e. the type of information conveyed, triggered by *stimuli* – in the slot `fanaux-list` as a list of neurons. In other words, the *microcolonnes* is the SOM reduced to the fanaux-list.

```
(update-fanaux SOMNAME <int>) ; if needed
(dolist (k SEQUENCCE)
  (setf (input SOMNAME) k)
  (learn SOMNAME :seq t))
```

N3 allows if needed to update by surjection [10] the fanaux-list with the method `update-fanaux` according to the 'old-fanaux-list' – if this latest is not nil. This means the modification of the arcs (at the SOM level) and of the involved edges (at the AREA level) if applicable. Mind that this will inevitably lead to an alteration – more or less significant depending on the degree of change – of what has been learned so far.

There are two ways to read the acquis of the MLT. The first one consists to retrace one (or more) *tournoi(s)* from a partial sequence temporally ordered according to the remanence or not. The method `locate-tournoi` allows to do that and gives as result a list of potential *tournoi(s)* sorted according to their weight.

```
(locate-tournoi SOMNAME '(3 ? 0) :remanence nil)
; ((0.28400004 (3 2 0)) (0.22000018 (3 3 0)) (0.2 (3 0 0))
   (0.15999967 (3 1 0)) (0.13600005 (3 4 0)))
(locate-tournoi SOMNAME '(3 ? 0) :remanence t)
; ((0.5 (3 2 0)) (0.5 (3 3 0)))
```

Note that with the remanence case, the given *tournoi* is compared with the learned *tournois* of MLT, and each match is retained with its weight, then the sum of all these weights is normalised. In the case of the length of the given *tournoi* is superior to the `cover-value`, this one is decomposed as follow : for instance consider the tournoi `(a b c d e)` and the `cover-value` equal to 3, then the computation is done on `(a b c) (b c d)` and `(c d e)`.

10. The surjection algorithm consists to explore the distance matrix of the 'new-fanaux-list' from 'the old-fanaux-list' in order to retain only the neurons with the smallest distance per row and by column. The conjunction of the last two is listed according to the index of the 'old-fanaux-list', adding if necessary the intermediate distances.

Also, to limit the search space – which increases exponentially – each *tournoi* composed with only wild cards is avoided. In that case, there are two undocumented functions to do the job as `all-tournoi` – which lists all the possible *tournois* according to a given order and the remanence as keys – and `rnd-tournoi` – which chooses randomly but weighted from the previous result.

The second one expresses the previous result in terms of probability by 'postpended' a wild card. Thus, the method `next-event-probability` allows to compute the probability of occurrences for a *fanal* to occur after a given subsequence of *fanaux* – as a list of indexed *fanaux* temporally ordered (with the possibility to add '?' as a wild card).

Then, the purpose of this function allows to estimate an occurrence – with the key `:result :eval` (set by default) – through the key function `:compute` (set with the function `rnd-weighted` by default).

The function `rnd-weighted` allows to choose a number at random between 0 and 1 such as this interval is conformed to the ordered repartition of probabilities (see the coloured stripes on figure 8 corresponding to the following code snippets).

```
(next-event-probability '(3 ? 0) SOMNAME :remanence nil
    :result :verbose)
; 2 => 30.749 %
; 3 => 22.460 %
; 0 => 16.756 %
; 1 => 16.043 %
; 4 => 13.993 %
```

```
(next-event-probability '(3 ? 0) SOMNAME :remanence t
    :result :verbose)
; 0 => 28.571 %
; 2 => 23.810 %
; 1 => 19.048 %
; 3 => 19.048 %
; 4 =>  9.524 %
```

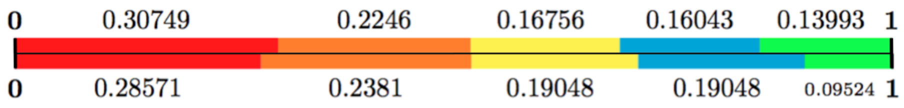


FIGURE 8 – Coloured stripes of the distributed probabilities without remanence on the top and with remanence at the bottom.

```
N3> (format t "~A~&;~v@{~A~:*~}~&" *PART-4* 36 #\*)
LONG-TERM MEMORY
;************************************
NIL
N3>
```

This is one of the main part of *Neuromuse3*. Indeed, the aim here is to create connections between each *microcolonne* – or *fanal* – as *tournois* inside a neuronal AREA as cliques.

Then, the learning process consists to connect a set of SOM together in order to create an associative memory.

The *macrocolonne* is managed by the class AREA, establishing the connections of each clique from each identified *fanal* involving relationships with incoming and outgoing arcs for each *fanal*. Then, each arc as the key of the hash table of AREA will be associated to a weight.

As seen previously, the method `locate-tournoi` gives as result a list of potential *tournois*, which by cross-checking with the *tournois* of the other *colonnes* – correlated in terms of cliques – allow the (re-)construction [11] of a sequence according to the user 'condition(s)'.

In the same vein, the method `locate-clique` allows to retrace clique(s) from partial data. Thus, from this step, the (re-)construction of a sequence which is connected to the reading of *tournois*, is done in a field of arcs and edges according to a deliberate heuristic by consensus and iteration.

The (re-)construction is effective when a learned sequence can be retraced as recollection or retrieving memory or as creation. In all case, this implies modalities defined by the user – see diagram on figure 9 [12].

The method `locate-clique` allows a first estimation of possible clique(s) according to two possible inputs :

11. The (re-)construction must be understood in terms of reminder and invention.
12. Concerning the diagram of the figure 9 :
    - The constraints or rules allow to limite the field of search, or in other words the constraints aims to reduce the field of possibilities. This can be done with as examples, harmonic rules, a specific scale or whatever. It would be possible to define the 'rules' on an interactive way, as a live coding for instance – involving in both case the user with its own code –, allowing the adaptability in terms of reaction to the immediate environment.
    - The bayesian inference can play a role for the estimation of a response in terms of probability focusing on the acquis – that is to say the highest probability.
    - The resolution is done on the filtered probabilities in order to get an optimal solution, which one can be oriented with some a posteriori rule(s) of the user.
    - Note that for now, the user interaction allows to manage the interrelationship between the agent and its environment in terms of affect.

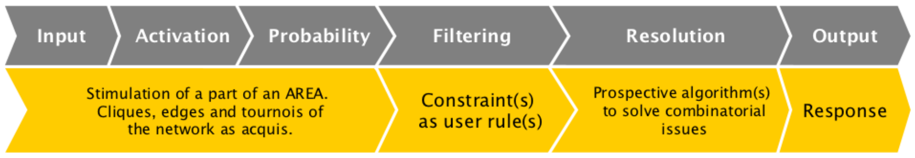

Input
Activation
Probability
Filtering
Resolution
Output
Stimulation of a part of an AREA. Cliques, edges and tournois of the network as acquis.
Constraint(s) as user rule(s)
Prospective algorithm(s) to solve combinatorial issues
Response

FIGURE 9 – Synoptic diagram of resolution I/O at the AREA level.

1. isolated node(s) : `((microcolonne_indice colonne_indice) ...)`
2. ordered node(s) : list of *microcolonne* indice(s) and/or wild card ordered according to the soms-list.

All connected *microcolonnes* to known elements are listed. This is what we observe on the figure 10 with the nodes (4 0) and (1 3), respectively the fifth *fanal* of SOM 1 and the second *fanal* of SOM 4. All microcolonnes whose connection number is equal to the number of known elements are collected. These connections are represented in solid lines.

Thus, we obtain a list of potential cliques ordered according to their respective weights in terms of probability.

```
(locate-clique AREANAME '((4 0) (1 3))) ;; isolated nodes
;; or
(locate-clique AREANAME '(4 ? ? 1)) ;; ordered nodes
; ((0.25050917 (4 2 2 1)) (0.22403261 (4 2 3 1))
;  (0.1965377 (4 2 0 1)) (0.120162934 (4 1 2 1))
;  (0.104887985 (4 1 3 1)) (0.10386966 (4 1 0 1)))
```

As seen before in the part 3 *Short-Term Memory*, it is possible at the AREA level to compute a potential clique according to some previous – can be partial – cliques interpreted in time as *tournois* with or without remanence with the function `next-event-probability`. The latter takes as the *head* argument a list of temporally ordered clique(s) and these are a list of ordered nodes in accordance with the second point seen previously referring to `locate-clique`.

To estimate the weight of a potential clique one takes the mean value of the weights of all edges constituting such clique.

At the AREA level, the probability for a given clique is done according to the previous cliques in terms of *tournois* in relation to the remanence as :

$$P_{C(T)} = \prod_{i \in SL} P(E_{n_i} | E_{p_i})$$

With SL as the `soms-list`, E_n as the next event – in other words the next clique or '*infon*' – and E_p as the previous events.

14. Generated with `(N3::>dot AREANAME '((4 0) (1 3)))`.

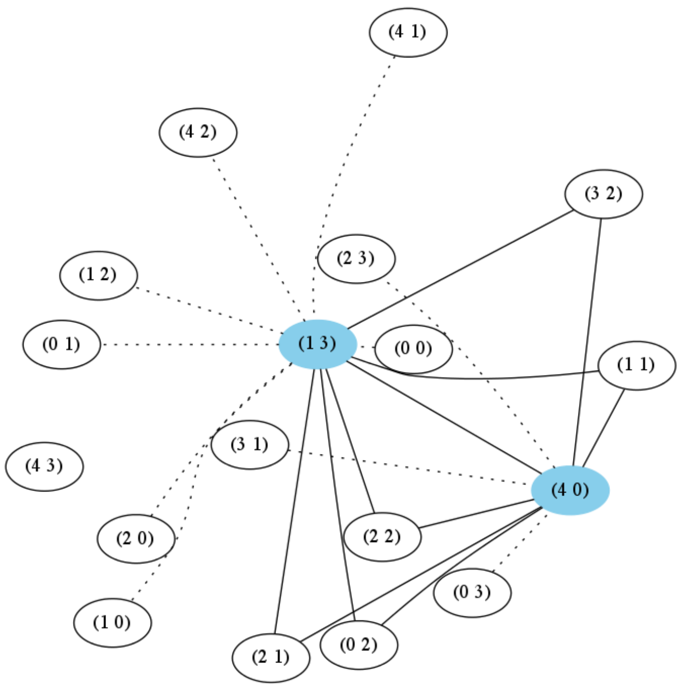


FIGURE 10 – Cliques estimation on a network AREA (partial view [14]) composed of four SOM of five *fanaux*.

Then the result is the product of the probability 'in time' and the weight of the resultant clique. Note that each probability is normalised.

AREA instantiation

To create an AREA, it needs first of all to set all needed SOM requisite for the '*infon*'. Then, the method `create-area` connect all these SOM according to their order in the soms-list.

Let the following procedure be an illustration of the principle of the learning processes at the AREA level.

```
;; Create AREA
(create-mlt 'SOMNAME1 116 100 :carte #'quadrare)
(create-mlt 'SOMNAME2 5 100 :carte #'quadrare )
   ...
(create-area 'AREANAME (list SOMNAME1 SOMNAME2 ...))

;; Mapping
```

```
(loop
   for i in '(SOMNAME1 SOMNAME2 ...)
   for data in '("~/dat1" "~/dat2" ...)
   ;; ------------------
   ;; set SCALING key(s) as a list of key+val for each SOM
   for kv in '((keyword val ...) ...)
   ;; ------------------
   do
   ;; if needed
   (setf (distance-in (id i))
     (lambda* (a b) (euclidean a b :position t :modulo t)))
   ;; ------------------
   (mapping (id i) 50000 (read-data (id i) data) :ds kv))

;; Learn the sequence `in time'.
(learn AREANAME
:seq (read-data AREANAME '("~/dat1" "~/dat2" ...) :scale t))
```

Following this procedure, mind the ordered arrangement of the sequence between the `soms-list` of the AREA and the list of data.

Note that each element of the sequence can be a path as a string or as a pathname or a data list.

Before launching the learning process of the AREA, it is necessary to set the `fanaux-list` either *a priori* with the key `n-fanaux` within the function `create-mlt` or *a posteriori* with the function `update-fanaux`. In the last case, one can estimate the optimum number of *fanaux* according to the user constraint(s) by computing the `dendrogram`. Then `(open-graph *tree*)` allows displaying the corresponding curve as the sum of the intra-class inertia by trimming with lines as peaks of the curve – meaning the minimum distance from the parent node (as the second number in the square bracket, the first number is the number of classes at this point). Note that in the graph, the impulse segments are displayed in a logarithmic scale for better readability. At this point, the approach is rather empirical, but it should be done once and for all.

About scaling

The bias in N3 is to normalise the input data in order to fit in a range between 0 and 1. The function `scaling` allows managing any input data according to

15. Note that :
- the learning is dependant of the encoding in relation to the desired environment or the kind of required 'control';
- each sequence is identified upstream and downstream by a '*tout-à-zéro*' marker;
- the learning at the AREA level consists to activate the current clique (via *microcolonnes* activation) in order to record it as arcs. At the same time, each MLT records arcs and *tournois* connected to the previous clique.

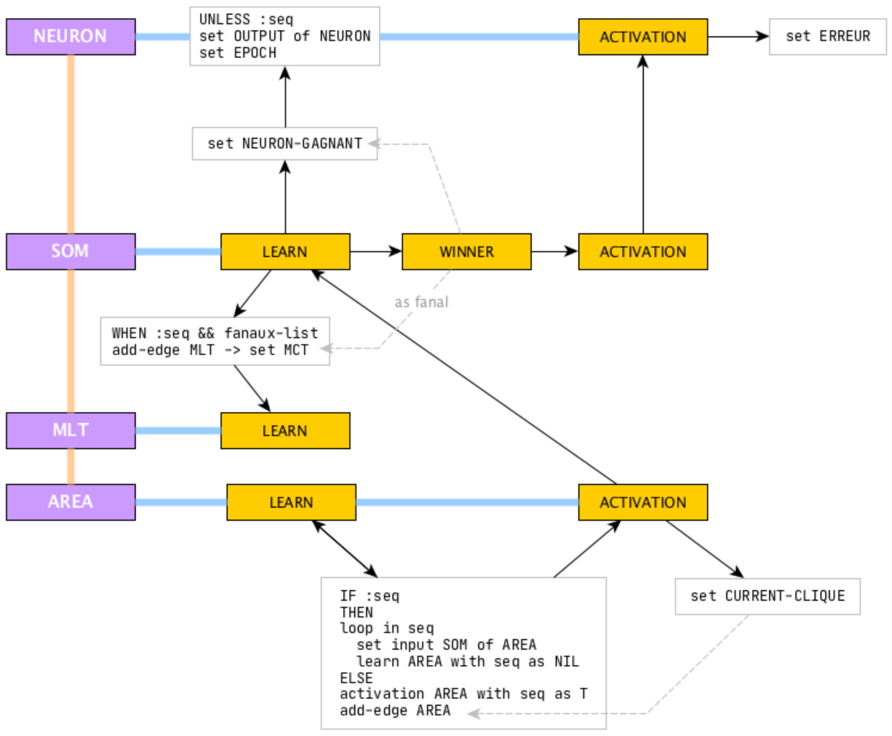


FIGURE 11 – Summary of the learning process in N3[15].

some modalities applied at the SOM level (key `:mlt`). By default, the 'data-scale' is bypassed. Then, to initiate or to interpret the data inputs, one needs to consider the following parameters :

the range which is estimated by default between the minimal value and the maximal value of the dataset (any other values outside the range will be clipped) ;

the curve allows to set the scale of the input data such as if this value equal to 0 (linear) then :

$$dataOut = \frac{(dataIn - inMin)(outMax - outMin)}{inMax - inMin} + outMin$$

and if this value < 0 (concave, negatively curved as exponential scale with the value $1 - e$) or > 0 (convex, positively curved as a logarithmic scale with the value $e - 1$) then :

$$dataOut = ln\left(\frac{b - dataIn}{a}\right)\frac{outMax - outMin}{curve} + outMin$$

$$\text{with } b = inMin + a \text{ and } a = \frac{inMax - inMin}{1 - e^{curve}}$$

the value of the curve should be set between $+18$ and -16[16] to avoid division by zero due to some rounding caused by computer systems about floating point numbers [Goldberg, 2001];

the type of the data input as :

- **norm** valid for all numbers, the argument is the range as a list with ‘minval’ and ‘maxval’ plus optionally the value of the curve (0 by default), or a number as the curve value;
- **dim** allows scaling data according to its dimension rank as a list of **norm** value by dimension.

The key `:update` set to *t* records the current settings of `scaling` in the history of the SOM and will be the default scaling settings. This information can be read and set ‘externally’ as follow :

```
;; read
(the-ds SOMNAME)
;; or
(history SOMNAME)
;; set
(set-ds SOMNAME <key> <arg> ...)
```

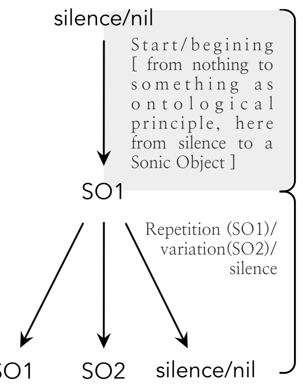
There is an alternative to scale or encode data for MLT – meaning to set the key `:mlt` –, which requires the macro function `lambda*` as the argument of the key `:bypass`. Thus, any kind of data can be formatted in the respect of the number of input of the considered MLT as a list.

```
N3> (format t "~A~&;~v@{~A~:*~}~&" *PART-5* 36 #\*)
DEVELOPMENTAL LEARNING
;************************************
NIL
N3>
```

Project launched in October 2014 by Olivier Georgeon on the internet platform MOOC[17] consisting of learning and developing a new concept in artificial intelligence called IDEAL (**I**mplementation of **DE**velopment**A**l **L**earning) – defined as an “approach to simulate the early mechanisms of emergent cognition based on theories of enactive cognition and on constructivist epistemology” [Georgeon et al., 2011].

16. These are some empirical limit values and, these can be different depending on the operating system and the lisp implementation.
17. http://liris.cnrs.fr/ideal/mooc/

Then, the idea in the context and in the continuation of *Neuromuse3*[18] is to create a kind of emergent intelligence from the network previously described, composed of cliques and *tournois*, by including additional learning process defined by IDEAL based on the experimentation and on the experience.


silence/nil
Start/begining [from nothing to something as ontological principle, here from silence to a Sonic Object]
SO1
Repetition (SO1)/ variation(SO2)/ silence
SO1
SO2
silence/nil

It is opportun to underline the philosophical aspect of this approach in terms of modeling including a causal device as computational cognition.

Thus, each agent develops its own understanding of its environment – i.e. an ontological conception in epistemological terms.

Also, IDEAL implies non-ontological interactional presuppositions – the ontological aspect will be defined by the agent as it experiences its interaction with the environment.

With that in mind, the following short description as paragraphs should help the understanding of the different concepts and processes used in this AI context.

1. There is no a priori knowledge of the environment.

'*It allows us to specify inborn behavioral preferences without specifying a predefined goal.*' [→ 41. Introduction, Georgeon, 2014, *loc. cit.* note 17]

In other words, only primitive interactions – i.e. innate – are defined through the initial function, which can evolve over time through the function `agent-result`.

'*This design choice follows from constructivist epistemology which suggests that sensorimotor patterns of interaction constitute the basic elements from which the agent constructs knowledge of the world.*' [→ 43. Architecture, Georgeon, 2014, *ibid.*]

The environment is therefore understood by the AI in terms of reading/writing relevant informations through neurons stimulation device and can be managed by programming constraints.

Glossary

Abbreviations : *fr.* = French ; *acc.* = *acception.*

Abstract : is an experiment or a result which refer to an interaction.

Enaction : interaction with the environment.

Primitive : low-level interaction defined by the couple experiment/result.

18. https://github.com/yannics/N3D

Proclivity : [*fr. propension*] inner, innate, natural force, which directs spontaneously or voluntarily towards an action, a behavior.

Valence : [*acc.* psychology] refers to the intrinsically pleasant or unpleasant quality of a stimulus or situation.

NOTE

Summary description

`E010` When the expected or desired result is effective, according to the conditions of `result010`, the qualitative character self-satisfied is 'activated' (frustrated otherwise). After n identical, similar, equivalent or other interactions, this is the bored character that is activated, and the system look for another experience to avoid 'boredom'.

`E020` The character is quantified with the valence expressing the motivation. This is applied to the primitive interactions.

- valence $\geq 0 \Rightarrow$ pleased
- valence $< 0 \Rightarrow$ pained

`E030` Continuation of `E020` including composite interactions – i.e. first level abstractions.

`E031` Variation of `E030` involving weights in terms of proclivity.
The proclivity is estimated as the product of the weight and the valence of the considered interaction.

Index

`*PROCLIVITY*` *<t/nil>*

`*REMANENCE*` *<integer>*

`*VERBOSE*` *<t/nil>*

`ADD-OR-GET-EXPERIMENT` *<existence>* *&key* label *<symbol>* string *<string>*

`ADD-OR-GET-INTERACTION` *<existence>* *<experiment/pre-interaction>* *<result/post-interaction>* *&key* valence *<number>* learn *<t/nil>*

`ADD-OR-GET-RESULT` *<existence>* *&key* label *<symbol>* string *<string>*

`CREATE-AGENT` *<name>*

`GET-PRIMITIVE-EXPERIENCE` *<experiment/interaction>*

`PICK-EXPERIMENT` *<existence>*

`PICK-OTHER-EXPERIENCE` *<existence>* *<experiment>*

`PREDICT` *<existence>* *<experiment/interaction>*
From *existence-interactions*, the evaluation is a *result.*

>REMANENCE *<existence> <experiment/interaction>*
Update of *existence-previous* by adding an *experiment* or an *interaction* in the MCT (short-term memory) with |MCT| = ***REMANENCE***

RUN-AGENT *<name> &key* duration *<integer>* step *<function>*
result *<function>*

SELECT-ANTICIPATE *<existence> <experiment/interaction/null>*
From *existence-interaction*(s), the evaluation is an *interaction*.

```
N3> (format t "~A~&;~v@{~A~:*~}~&" *PART-6* 36 #\*)
SEQUENCING
;************************************
NIL
N3>
```

NOTE

Important

— The duration expressed in pulse for a clique must be understood as indice, that is to say 0 for 1 pulse, 1 for 2 pulses, and so on. Conventionally, this is the first value of the clique or the list destinated to fill the buffer of the sequencing.

— The net of 'sequencing' concerns for now only the AREA.

— The synchronisation is effective only when the 'leader' is running, and match the beginning of the measure of this 'leader' according to the `anacrusis` value of the 'follower'. Also, the synchronisation is effective to resume a routine paused or interrupted by a subroutine. Note the 'leader' can be silent as a conductor as long as it is activated.

For convenience :

— the 'dub' sequencing is set as a global variable, so in order to be recognise as such, its name should be wrapped between asterisks ;

The sequencing is managed with the class of the same name described in the following table 4. Each sequencing as an object is instantiated generating its own sequence with its own rule(s). Each parameter and function is accessible from the name or the 'dub' (i.e. nickname) of this object leading to interact at any time with code or other sequencing instantiation.

The principle consists to estimate an occurrence according to the context defined for instance by what I called a 'dynamic buffer' – as ordered events [19] – which is updated each time the computation is done by the function(s) set in routine. In that case, the computation consists mainly to apply the function `next-event-probability` – taking as key argument the odds function `:compute` – according to the rule defined by the slot of the same name.

		Slot	**Input**	**Note**
SEQUENCING		`name`	SEQUENCING	
	⚙	`dub`	*symbol*	'nickname'
	⚙	`description`	*string*	
	⚙	`net`	MLT|AREA	or `:corpus` [20]
		`buffer-out`	*list*	
	⚙	`buffer-size`	*integer*	maximum size
	⚙	`buffer-thres`	*integer*	minimum size
	⚙	`pulse`	*number*	for one beat
	</>	`sync`	SEQUENCING	→ `set-sync`
	⚙	`meter`	*number*	number of beat
	⚙	`anacrusis`	*integer*	number of pulse [21]
	⚙	`pattern`	*list*	list of events
	</>	`rule`	*function*	→ `set-rule`
	</>	`routine`	*function*	→ `set-routine`
	</>	`subroutine`	SEQUENCING	→ `set-subroutine`
	⬠	`ip`	*string*	localhost by default
	⬠	`port`	*integer*	
	⬠	`tag`	*symbol*	`/N3` by default
		`mem-cache`	*hash-table*	

TABLE 3 – The sequencing class.

The rule should return what it is expected as argument if required of the routine. Mainly, the rule predefines the event as a partial clique or event. Thus, the rule(s) is(are) embedded in a `lambda*` function with as argument the sequencing object – conventionally named *self*.

Regarding the function(s) involve(s) in `set-routine`, the idea is to update the 'dynamic buffer' as a complete clique or event. In case of more than one function, the processes are sequential.

Concerning the routine, when it set, one thread as deamon is created in order to do the computation, which consists to fill the buffer-out according to the buffer-size. This 'deamon' thread is located in the mem-cache with the key

19. The order is in time-inverse understanding that each computed event is pushed into the buffer as the first item of this buffer.

20. An alternative to the AREA or MLT is to set a specific corpus as a list, a data file, or a function, initiated with the function `set-corpus`. The corpus itself is a data file stored in `*n3-backup-directory*/data/`.

21. Kind of delay according to the starting point of the measure as modulo of the 'leader' `meter` (which is defined by the slot `sync`).

compute. When the routine is acting, a second thread is created to manage the buffer(s) involved in time by sending data via OSC – see `+routine+` function. This ‘main’ thread is located in the mem-cache with the key *routine*.

	Key	Value	Note
	`last-event`	*list*	keep track of the last event sent
	`compute`	*thread*	fill the buffer
	`routine`	*thread*	send OSC message
	`routine-initform`	*function*	`+routine+`
	`subroutine-state`	*boolean*	
</>	`corpus`	*function*	`lambda*` function of the corpus
</>	`thirdpart`	*list*	of `lambda*` function
⚙	`ind`	*integer|key*	modulate OSC message [22]
⚙	`pattern-counter`	*number*	
	`init-clock`	*number*	`+clock+` at the first OSC
	`next-pulse`	*number*	`+clock+` to trigger next OSC

TABLE 4 – The `mem-cache` of the sequencing class.

About synchronisation

First, some definitions in context :

`*latency*` – time step of the `+clock+` as *dt* in second.

`+beat+` – tempo as the ratio of the beats defined by the `meter`, i.e. the duration in second of one beat as a global variable allowing to control globally the speed of the play. The value is the bpm (beat per minute) divided by 60.

`pulse` – number of equal pulsations for one beat as the minimal duration or as the Greatest Common Divisor (GCD) of the intended or expected rhythm.

`meter` – pattern of beats as the numerator of the time signature.

`anacrusis` – delay in number of pulse prepending the beginning of the measure.

The computation of duration in number of `*latency*` as *dt* is done with the number of pulse divided by the `+beat+` times `*latency*`.

The synchronisation of one sequencing (the ‘follower’) is subordinated to another one (the ‘leader’). Then the setting of the synchronisation of a ‘follower’ (slot `sync`) requires the ‘leader’ sequencing name and if needed the value of delay defined by the slot `anacrusis`.

22. Optional argument of internal function `bo`. The function `bo` returns either the message to send via OSC (basically a clique), or an information about this message. Thus, the optional `ind` can be an integer as indice to select an specific item of the clique (or as item of the buffer) ; or the keyword `:pos` to get the coordinate of the clique ; or the keyword `:next` to append the next clique.

About time lag

When the computing time does not supply the buffer-out, a subroutine (according to the buffer-thres) takes over the routine until the complete filling of the buffer-out (according to the buffer-size and taking into account the meter when the synchronisation is effective). This subroutine is either silent (which is the default behavior) or a routine.

Dynamic Markov Chain

See 4.7, 6.5.4 and 8.4 in *Journal of Generative Sonic Art* [Ics, 2014/24]. In short, what I call a 'Dynamic Markov Chain' allows increasing or coercion decreasing the order in a manner that the transition probabilities does not reach 100% for a single event.

See also code instance on the Annex 2 on page 43.

```
N3> (format t "~A~&;~v@{~A~:*~}~2&" *END-PART* 36 #\*)
DOCUMENTATION
;************************************

NIL
N3> (format t "~A" *CHAPTER-A*)
INDEX
NIL
N3>
```

ALL-AREA

→ *list*

...

ALL-SEQUENCING

→ *list*

...

ALL-SOM

→ *list*

...

N3-BACKUP-DIRECTORY

→ *string*

... 26, 32, 34

`*TREE*`

→ *node*

Last node built as root with `dendrogram`. ...

`ACT-ROUTINE`

`self`[sequencing]
→ *thread*

...

`CAH-FANAUX`

`self`[mlt] `tree`[node] `n-class`[int]
&key `trim`[bool]
→ *list*

Divide the tree defined by *self* into *n* classes by trimming. The result is a list of leaves by class. If the key *trim* is set, the result is a list of nodes. ...

`COPY`

`self`[som|area|sequencing]
&key `newname`[symbol] `with`[list] `reset`[bool]
→ *nil*

The optional keywords `:with` and `:reset` concern only *self* as an AREA. The argument of `:with` is a new `soms-list` as a list of SOM new names – which should be the same length of the `soms-list` of *self*. ...

`CREATE-AREA`

`name`[symbol] `soms-list`[list]
→ *area*

Macrocolonne instantiation. ... 19

`CREATE-MLT`

`name`[symbol] `n-input`[int] `n-neurons`[int]
&key `carte`[function] `topology`[int] `field`[num|list] `cover-value`[int] `n-fanaux`[int]
→ *mlt*

Colonne instantiation with *n* input as stimuli, and the number of neurons constituting the carte of the SOM.
`:carte` – Function (if lambda function then arguments are (`a`) the name as MLT and (`b`) the number of neurons).
`:topology` – Number of dimensions.
`:field` – List of maximum value by dimension or maximum value for all

dimensions (by default 10 with `rnd-map` and $\lfloor \sqrt[d]{N} \rfloor$ with `quadrare`).
`:cover-value` – length of the MCT.
`:n-fanaux` – Initiate the fanaux list according to the hierarchical clustering using Ward's method. ... 8, 20

`CREATE-SEQUENCING`

&key `dub`[symbol] `description`[string] `net`[som|mlt|area] `remanence`[bool] `buffer-size`[int] `buffer-thres`[int] `pattern`[list] `odds`[function] `sync`[bool] `meter`[num] `pulse`[num] `tag`[symbol] `osc-in`[list|int] `osc-out`[list|int]
→ *sequencing*

Sequencing instantiation. ...

`DENDROGRAM`

`self`[som|mlt|list] `aggregation`[int]
&key `diss-fun`[function] `newick`[bool] `with-label`[bool] `and-data`[bool|list]
→ *node*

Compute the hierarchical clustering of *self* using one of the following aggregation method: single linkage (1), complete linkage (2) and Ward's method (3).
`:diss-fun` – Function (if lambda function then arguments are (`a`) and (`b`) as two items to compare).
`:newick` – Save *newick* file (set by default).
`:with-label` – Display on the dendrogram the label of all nodes (by default only leaves are labelised).
`:and-data` – Allows to integrate a bijective alist of symbol or string relative to the input data as a list and/or write tree file. In the SOM context, write data file – in addition to the tree file – with for each line the number of classes according to the minimal distance of the parent node and the sum of all intra-class inertia. ... 10, 11, 20, 29

`DIFFERENTIAL-VECTOR`

`xs/l1`[list] `xs/l2`[list]
&key `result`[keyword] `opt`[keyword] `thres`[num] `ended`[keyword] `cluster`[keyword] `tolerance`[keyword]
→ *num|list*

`(documentation 'differential-vector 'function)`
See also Chapter 5 *Motivic Recognition* in *Journal of Generative Sonic Art* [Ics, 2014/24]. ...

`EUCLIDEAN`

`arg1`[neuron|som|list|event] `arg2`[neuron|list|event]
&key `modulo`[num|bool] `position`[bool] `weight`[num|list]

→ *num*

... 9

`EXP-DECAY`

`epoch`[int] `init-val`[num] `learning-rate`[num]
&optional `final-val`[num]
→ *No value*

... 9, 33

`FN-MEX`

`distance`[num] `radius`[num] `learning-rate`[num]
&key `inh`[num]
→ *num*

'Mexican hat'. ... 9

`GAUSS`

`distance`[num] `radius`[num] `learning-rate`[num]
→ *num*

... 9

`GET-LEAVES`

`self`[node]
&key `trim`[num|node]
→ *list*

List all leaves from *self* as a root node, or from a given node or distance set with the key *trim*. ...

`KILL-ROUTINE`

`self`[sequencing]
&key `message`[list]
→ *nil*

...

`LAMBDA*`

lambda-list
→ *function*

Macro as a 'normal' lambda function – but do not require to be call with `#'` – allowing to keep track of the 'source code'. ... 22, 26, 27, 43

`LEARN`

`self`[som|mlt|area]
&key `seq`[list]
→ *No value*

The key *seq* takes as argument a ordered list of sequential input data according to the `soms-list`; the data can be a list of input data or a data file defined by its pathname as such or as a string, or *nil* by default. ...

`LOAD-NEURAL-NETWORK`

`nn`[string]
→ *nil*

Load SOM, MLT or AREA according to its full pathname or according to its name if it is stored in the `*n3-backup-directory*` – see diagram on figure 12 page 37. ... , 37

`LOCATE-CLIQUE`

`self`[area] `nodes`[list]
&key `test`[function]
→ *list*

List of potential *microcolonne*(s) constituting a clique from a partial list of node(s). ... 17

`LOCATE-CYCLE`

`nodes`[list|hash-table|mlt|area]
&optional `kw`[keyword] `order`[int]
→ *list*

List all cycles from a list of edges.
The keyword is effective only on the MLT or the AREA as the second value of the multiple return values.
`:wlist` – list of weight's edges by cycle.
`:wstats` – list of [sum of weight's edges by cycle + mean value of the normalized *wlist* + standard deviation of the normalized *wlist*] by cycle
...

`LOCATE-TOURNOI`

`self`[mlt] `nodes`[list]
&key `remanence`[bool] `test`[function]
→ *list*

List of potential *tournoi*(s) – in the form of a one-dimensional coordinate position list of the *microcolonnes* of the temporally ordered `fanaux-list` of *self* – whose order is defined by the length of the *tournoi* and where each unknown is represented by a wild card type '?'.

When the `remanence` is effective, it takes into account the `cover-value` during learning. ... 15, 17

`MAPPING`

`self`[som] `iteration`[int] `dataset`[list]
&key `init-lr`[num] `end-lr`[num] `init-rad`[num] `end-rad`[num]
`df`[function] `ds`[list]
→ *No value*

`:init-lr` – Initial `learning-rate` (0.1 by default).
`:end-lr` – Final `learning-rate` (0.01 by default).
`:init-rad` – Initial `radius` (the default value is the mean value of the `field` of *self* divided by 2).
`:end-rad` – Final `radius` (0.1 by default).
`:df` – Decreasing function (`exp-decay` by default).
`:ds` – Set `scaling` key(s) as a list of pair(s) keyword+value. ...

`NET-MENU`

Display available network to load. ...

`NEXT-EVENT-PROBABILITY`

`head`[list] `self`[list|mlt|area]
&key `remanence`[bool] `result`[keyword] `compute`[function]
`opt`[keyword|list]
→ ...

Estimate from probabilities – and according to the function defined by the key *compute* – the occurrence of an event after a given subsequence of symbols when *self* is a list of symbols or a list of index(es) *microcolonnes(s)* or clique(s) – when *self* is respectively MLT or AREA– temporally ordered.
The key `:result` takes as argument:
`:eval` by default or,
`:prob` as an ordered list of probabilities or,
`:verbose` to display the probabilities.

The key `:opt` in the context of –
• AREA allows with the keyword `:onset` to evaluate a possible start according to the probability of an event as such and as the first occurrence of the onset of each SOM. [→ should be improved taking into account the weight of the onset]; and with the keyword `:buffer` to evaluate the last event of the buffer according to its tail as remanence.
• MLT allows to evaluate the next event among the list of expected events as argument. ... 16, 18, 26

OSC-LISTEN

`port`[int]

Usage:

```
#+sbcl (defparameter listen-port-7771 (sb-thread:make-thread
  #'(lambda () (osc-listen 7771)) :name "listen-port-7771"))
#+ccl (defparameter listen-port-7771 (ccl:process-run-function
  "listen-port-7771" #'(lambda () (osc-listen 7771)))) ...
```

QUADRARE

`self`[som] `nbre-neurons`[int]
&key `about`[int] `topology`[int]
→ *No value*

Multidimensional mapping according to the relationship $\sqrt[d]{N} \in \mathbb{N}$ with N the number of neurons and d the number of dimensions. ... 9, 30

RND-MAP

`self`[som] `nbre-neurons`[int]
&key `about`[int] `topology`[int]
→ *No value*

Random multidimensional mapping. ... 9, 30

SAVE

`self`[node|som|mlt|area|sequencing]
→ *nil*

Save *self* in `*n3-backup-directory*`.
If *self* is a node, it is saved in the following subdirectories:
`/SOM[name]/aggregation[number]+NODE[name]+SOM[epoch]/`. ...

SCALING

`data`[num|list]
&key `minin`[num] `maxin`[num] `minout`[num] `maxout`[num] `curve`[num] `mlt`[mlt] `norm`[list|num] `dim`[list|num] `bypass`[bool|{lambda*}] `update`[bool]
→ *num|list*

Normalize *data* value(s) occuring between *minin* and *minout* – as minimal and maximal values of the dataset by default – within the interval *minout* and *maxout*, respectively 0 and 1 by default.
`:curve` – Map *data* from a curve-exponential input range (set to 0 – namely linear – by default).
`:mlt` – Allows to retrieve initial data values or to keep track of the data

range of MLT (last setting by default) according to the interpretation of the input data in terms of type defined by the keys *norm* or *dim*. The key *update* allows recording this new setting. ... 20, 22, 33

SEND-UDP

`message`[list] `host`[string] `port`[int]
→ *T*

The message is a list of string. ...

SEQUENCING-MENU

Display available sequencing to load. ...

SET-CORPUS

`self`[sequencing] `corpus`[string|pathname|symbol|list|{lambda*}]
→ *nil*

... 26

SET-PATTERN

`self`[sequencing] `lst`[list]
→ *list*

...

SET-ROUTINE

`self`[sequencing]
&body `funcs`[function(s)]
→ *lambda-list*

... 26

SET-RULE

`self`[sequencing]
&body `funcs`[function(s)]
→ *lambda-list*

... 26

SET-SUBROUTINE

`self`[sequencing] `subroutine`[sequencing]
&key `sync`[bool] `anacrusis`[int]
→ *table*

... 26

SET-SYNC

`follower`[sequencing] `leader`[sequencing]
&key `anacrusis`[int] `save`[bool]
→ *table*

... 26

TREE-MENU

Display available tree to load. ...

UPDATE-COVER-VALUE

`self`[mlt] `val`[int]
&key `compute`[function]
→ *No value*

Allow to update the cover-value – if not nil – of *self* and, consequently, the hash table defined by the slot `trns` and the short-term memory defined by the slot `mct`. ... 14

UPDATE-FANAUX

`self`[mlt] `n-fanaux`[int|list]
→ *No value*

Allow to (re-)initiate the fanaux-list of *self* – updating if needed all connections linked with the old fanaux list by surjection (see note 10 on page 15). ... 14, 15, 20

```
N3> (format t "~A" *CHAPTER-B*)
REFERENCES
NIL
N3>
```

→ The references to go further or deeper for advanced readers are prepended by an asterisk.

* Claude Berrou, Vincent Gripon. *Nearly-optimal associative memories based on distributed constant weight codes.* Information Theory and Applications Workshop, September 2012, pp. 269 to 273.
https://www.researchgate.net/publication/254026422

Claude Berrou, Vincent Gripon. *Petite mathématique du cerveau – Une théorie de l'information mentale.* Odile Jacob, 2012.

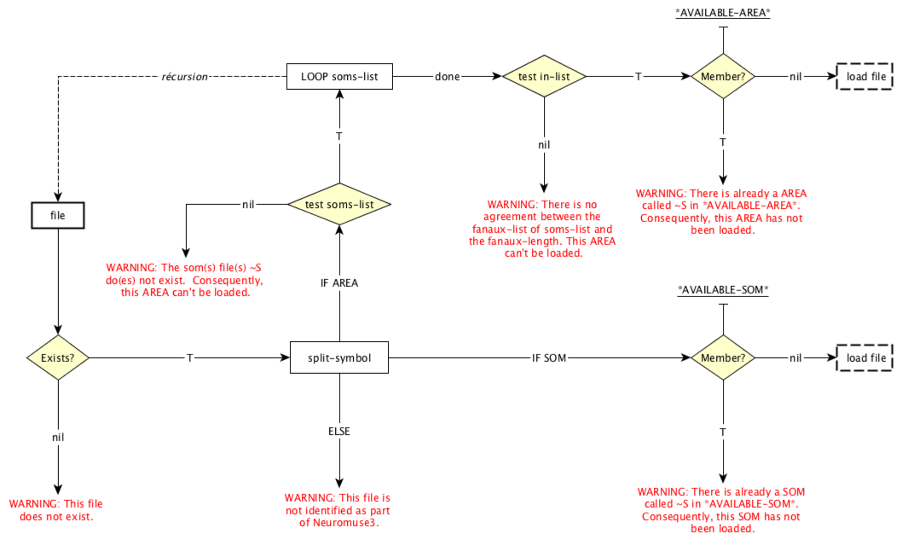

AVAILABLE-AREA
récursion
LOOP soms-list
done
test in-list
T
Member?
nil
load file
T
nil
T
file
nil
test soms-list
WARNING: There is no agreement between the fanaux-list of soms-list and the fanaux-length. This AREA can't be loaded.
WARNING: There is already a AREA called ~S in *AVAILABLE-AREA*. Consequently, this AREA has not been loaded.
WARNING: The som(s) file(s) ~S do(es) not exist. Consequently, this AREA can't be loaded.
IF AREA
AVAILABLE-SOM
Exists?
T
split-symbol
IF SOM
Member?
nil
load file
nil
ELSE
T
WARNING: This file does not exist.
WARNING: This file is not identified as part of Neuromuse3.
WARNING: There is already a SOM called ~S in *AVAILABLE-SOM*. Consequently, this SOM has not been loaded.

FIGURE 12 – Synoptic diagram of `load-neural-network`.

* Claude Berrou, Vincent Gripon. *Sparse Neural Networks With Large Learning Diversity*. IEEE transactions on neural networks, Volume 22, 2011, pp. 1087 to 1096.
https://arxiv.org/pdf/1102.4240

Antonio Rosa Damasio. *L'erreur de Descartes*. Odile Jacob, Paris 2010.

CentraleSupélec. *Cours d'Etienne Klein : les deux théories de la relativité d'Albert Einstein*. YouTube, 23 avril 2020.
https://www.youtube.com/watch?v=1NQ6WHDgtM0

Marie Chavent. *La classification automatique de données quantitatives*. Institut de Mathématiques de Bordeaux UMR 5251, 2013.
http://www.math.u-bordeaux.fr/∼mchave100p/.../2013/10/cours_classif_quanti.pdf

François-Xavier Demoures, Eric Monnet. *Le monde à l'épreuve de l'imagination. Sur « l'expérimentation mentale »*. Tracés, 2005
https://journals.openedition.org/traces/177

Olivier Georgeon, Frank Ritter. *An intrinsically-motivated schema mechanism to model and simulate emergent cognition*. Cognitive Systems Research, 2011.
[doi :10.1016/j.cogsys.2011.07.003]
https://projet.liris.cnrs.fr/ideal/doc/Georgeon02011-emergent-cognition.pdf

* David Goldberg. *What Every Computer Scientist Should Know About Floating-Point Arithmetic*. Numerical Computation Guide, July 2001, pp. 171 to 264.
https://ece.uwaterloo.ca/∼dwharder/NumericalAnalysis/02Numerics/Double/paper.pdf

Werner Herzog. *L'énigme de Kaspar Hauser*. DVD Le Cinéma du Monde, série 7, [*Jeder für sich und Gott gegen alle*] released in November 1974.

Yann Ics. *Journal of Generative Sonic Art*. Web publication, 2014/2024.
https://github.com/yannics/GSA/blob/master/gsa.pdf

* Xiaoran Jiang, Vincent Gripon, Claude Berrou. *Learning long sequences in binary neural networks*. 4th International Conference on Advanced Cognitive Technologies and Applications, July 2012, pp. 165 to 170.
https://www.researchgate.net/profile/Xiaoran_Jiang/publication/265508848

John McCarthy. *From here to human-level AI*. Artificial Intelligence, Volume 171, Issue 18, 2007, pp. 1174 to 1182. [doi :10.1016/j.artint.2007.10.009]
https://www.sciencedirect.com/science/article/pii/S0004370207001476

Marshall McLuhan. *Understanding Media – The Extensions of Man*. First edition 1964, MIT Press edition, 1994.

George Orwell. *1984*. First edition 1949. Trad. Amélie Audiberti, Gallimard, 1991.

Cosma Shalizi. *Distances between Clustering, Hierarchical Clustering*. 36-350, Data Mining, 2009.
http://www.stat.cmu.edu/~cshalizi/350/lectures/08/lecture-08.pdf

Herbert-George Wells. *L'Ile du docteur Moreau*. First edition 1896 [*The Island of Dr. Moreau*], Club français du livre, Paris 1961. Traduction Henry Durand-Davray.

```
N3> (format t "~A" *CHAPTER-C*)
BIBLIOGRAPHY
NIL
N3>
```

→ Here is the list of books, articles and web pages that have been used in some way in the development of this work.

Sous la direction de Daniel Andler. *Introduction aux sciences cognitives*. Gallimard, 2012.

Christophe Assens. *Connexionnisme et théorie des organisations*. Cahiers de Recherche DMSP n°240, Novembre 1995.
https://www.christophe-assens.fr/articles/cahier-de-recherche/

Hugues Bersini. *De l'intelligence humaine à l'intelligence artificielle*. Ellipses, Paris 2006.

Ludwig Von Bertalanfy. *Théorie générale des systèmes.* Dunod, Paris 2012.

Pierre Buser, Claude Debru. *Le temps, instant et durée – de la philosophie aux neurosciences.* Odile Jacob, Paris 2011.

David Michael Cottle. *Computer Music with examples in SuperCollider 3.* Web publication, 2005.
http://rhoadley.net/courses/tech_resources/supercollider/tutorials/cottle/CMSC7105.pdf

Colloque annuel 2008 sous la direction de Stanislas Dehaene et Christine Petit. *Parole et musique.* Odile Jacob, Paris 2009.

Jean-Paul Delahaye. *L'intelligence et le calcul – de Gödel aux ordinateurs quantiques.* Belin, Pour la science 2002.

Jean Fourastié. *Comment mon cerveau s'informe – Journal d'une recherche.* Robert Laffont, 1974.

Peter Hadreas. *Searle versus Derrida ?.* Philosophiques, Volume 23, n°2, 1996, pp. 317-326. Traduction Josette Lanteigne.
https://doi.org/10.7202/027399ar

Arslan Hamza Cherif. *Neurogrid : un circuit intégré pour simuler 1 million de neurones.* Web publication, 2014.
https://www.developpez.com/actu/70629/

Jocelyne Kiss. *Composition musicales et sciences cognitives - Tendances et perspectives.* L'Harmattan, 2004.

Rémy Lestienne. *Dialogues sur l'émergence.* Le Pommier, Paris 2012.

Pierre Mercklé. *Sociologie des réseaux sociaux.* La découverte, Paris 2011.

Alp Mestan. *Introduction aux Réseaux de Neurones Artificiels Feed Forward.* Web publication, 2008.
https://alp.developpez.com/tutoriels/intelligence-artificielle/reseaux-de-neurones/

Israel Rosenfield. *L'invention de la mémoire.* Flammarion, 1994.

Denis Shasha, Cathy Lazere. *Quand la vie remplace le silicium – Aux frontières de la bio-informatique.* Dunod, Paris 2011.

Rupert Sheldrake. *La mémoire de l'univers.* Édition du Rocher, 1989.

Rupert Sheldrake. *Réenchanter la science.* Albin Michel, Paris 2012.

Bob Snyder, Robert Snyder. *Music and Memory : An Introduction.* A Bradford Book Mit Press, 2001.
https://monoskop.org/images/f/f3/Snyder_Bob_Music_and_Memory_An_Introduction.pdf

Richard Solé, Bernat Corominas Murtra, Jordi Fortuny. *La structure en réseaux du langage*. Dossier hors-série Pour la science, n°82 janvier/Mars 2014, pp. 8-15.

Caroline Tourbe. *Nous avons un deuxième cerveau !*. Science & vie, n°1058 novembre 2005, pp. 64-79.

Andrea Valle. *Introduction to SuperCollider*. The MIT Press, 2016.
https://www.scribd.com/document/381922613/intro-to-SuperCollider-pdf

Scott Wilson, David Cottle, Nick Collins. *The SuperCollider Book*. The MIT Press, 2011.
https://www.scribd.com/doc/296526523/

Les Dossiers de Sciences & Univers. *Votre cerveau va vous étonner !*. n°4 novembre 2015/janvier 2016

```
N3> (format t "~A~&;~v@{~A~:*~}~&" *NOTE* 36 #\*)
APOSTIL
;************************************
NIL
N3>
```

This documentation aimed to present the *Neuromuse3* project to this point in time. Also, the project is intended as a reflection about what can be an Artificial Intelligence in a musical context.
Some work needs to be completed, some other achieved.

The *Long-term memory* part has to be experimented more deeply, especially for the (re-)construction of sequences.
The MLT class should be distinct from the SOM class in order to connect any kind of classification to the MLT, focusing thus on the AREA network in terms of nodes and edges.
The sequencing class needs improvements such as a specific syntax to code and to set rules and processes, including the interaction(s) between voices as layers and the environment as incoming OSC messages.

The coding of the *Developmental learning* part is still ongoing. After that, one has to merge IDEAL with AREA, by creating classes able to manage both systems. The scope of this work is to say the least about the philosophical approach it is possible to envisage, and on which terms. Indeed, in music, the 'walls' are not fixed and solid objects. The musical environment is very dependant of what has been learned and how forged by its own ontology. Moreover, the character 'sensitive' – which must be linked to a kind of mood – has to be defined in terms of computing language. The challenge seems daunting, but one must take it up, at least because it can learn a lot about us as a human being.

```
Currently developed with SBCL 2.3.11 and CCL64 1.12.2
```

Thanks for any support or contribution you could bring ...

```
... to be continued ...
```

```
... work in progress ...
```

```
N3> (format t "~A~&;~v@{~A~:*~}~&" *CODE* 36 #\*)
ANNEXES
;************************************
NIL
N3>
```

1 Displaying a learning process of a SOM through SuperCollider via OSC

This code has been used to generate the illustrations of the figure 3 on page 10 and its report is just about to give an idea of how to map SOM in N3 and how to manage OSC protocol from N3 to SuperCollider – mind that it works for limited SOM (100 neurons is fine – that is to say 10×10), and *in fine*, the use of a graphical software like Processing [23] will be more appropriate to display more complex configuration.

```
/*
(create-mlt 'COLOR 3 100 :carte #'quadrare)
;; or :carte #'rnd-map

;; hack to keep track of the initial value
(dolist (e (neurons-list COLOR))
```

23. https://processing.org/

```
   (setf (synapses-list e) (output e)))

;; set distance with modulo
;(setf (distance-in COLOR)
;   (lambda* (a b) (euclidean a b :position t :modulo t)))

(let ((num 3000) ;;  5 minutes with *latency* = 0.1
      (dataset (loop for i in (neurons-list COLOR)
                 collect (synapses-list i)))
      (initial-learning-rate 0.1)
      (initial-radius (/ (funcall #'mean
         (list! (field COLOR))) 2))
      (initial-epoch (epoch COLOR)))
  (setf *latency* 0.1)
  (dotimes (k num)
    (setf (input COLOR)
          (nth (random (length dataset)) dataset)
          (radius COLOR)
          (exp-decay (1+ (- (epoch COLOR) initial-epoch))
   initial-radius num 0.1)
          (learning-rate COLOR)
          (exp-decay (+ 1 (- (epoch COLOR) initial-epoch))
   initial-learning-rate num 0.01))
    (learn COLOR)
    (send-udp (read-from-string
       (format nil
          "(\"\/N3-COLOR\" \"[~{[~{~S~^, ~}]~^, ~}]\")"
          (loop for e in (neurons-list COLOR)
             collect
             (append (xpos (id e)) (output (id e))))))
       "127.0.0.1" 7771)
    (sleep *latency*)))
*/

(
var w, o, data, a = 10, b = 50;
w = Window("SOM activity: COLOR",
    Rect(rrand(100, 200), rrand(100, 200), a*b, a*b));
w.view.background = Color(0, 0, 0, 1);
w.onClose = {o.free};
w.front;
w.alwaysOnTop_(true);
thisProcess.openUDPPort(7771);
o = OSCFunc({ arg msg, time, addr, recvPort;
    data=msg[1];
    data=data.asString.interpret;
    {
    x = UserView(w, Rect(0, 0, a*b, a*b)).canFocus_(false);
    x.drawFunc = {
      Pen.use {
        Pen.width = 0.2;
        Array.fill(data.size,{|i|
          Pen.color = Color.new(
            data[i][2],
            data[i][3],
            data[i][4]);
  //Pen.addOval(Rect(data[i][0]*b, data[i][1]*b, b/2, b/2));
  Pen.addRect(Rect(data[i][0]*b, data[i][1]*b, b, b));
          Pen.perform(\fill);
        });
      };
    };
       x.refresh;
    }.defer;
}, '/N3-COLOR');
)
```

2 Markov chain AREA instance for real time Music processing

2.1 Learning dataset – rhizom-like network

According to the `+corpus+` such as each sequence is defined as follow :

```
((dur_1 deg_1 int_1) (dur_2 deg_2 int_2)
   ... (dur_n deg_n int_n))
```

Let initiate the area network with this corpus thus formatted.

```
(create-mlt 'CS-INT 23 100)
(create-mlt 'CS-DEG 12 100)
(create-mlt 'CS-DUR 8 100)
(create-area 'CORPUS '(CS-DUR CS-DEG CS-INT))
```

Optionally set inner euclidean distance with modulo.

```
(loop for i in (soms-list CORPUS)
   do
   (setf (distance-in (id i))
     (lambda* (a b) (euclidean a b :position t :modulo t))))
```

As seen previously in this writting, the mapping is done for each SOM according to their respective scaling using `lambda*` function.

```
(mapping CS-INT 5000 (loop for i from -11 to 11 collect i)
   :ds '(:bypass (lambda* (x) (>thrifty-code x 23 :mid 1))
   :update t))
```

```
(mapping CS-DEG 5000 (loop for i from 0 to 11 collect i)
   :ds '(:bypass (lambda* (x) (>thrifty-code (mod x 12) 12
      :left 0)) :update t))
(mapping CS-DUR 5000 (loop for i from 1 to 8 collect i)
   :ds '(:bypass (lambda* (x) (>thrifty-code x 8 :right 1))
   :update t))
```

Then initiate the fanaux list according to the discrimination number defined apriori. This can be understood as the perception limits of the AI.

```
(update-fanaux CS-INT (loop for i from -11 to 11
   collect (progn (setf (input CS-INT)
      (>thrifty-code i (nbre-input CS-INT) :mid 1))
      (winner CS-INT))))
(update-fanaux CS-DEG (loop for i from 0 to 11
   collect (progn (setf (input CS-DEG)
      (>thrifty-code i (nbre-input CS-DEG) :left 0))
      (winner CS-DEG))))
(update-fanaux CS-DUR (loop for i from 1 to 8
   collect (progn (setf (input CS-DUR)
      (>thrifty-code i (nbre-input CS-DUR) :right 1))
      (winner CS-DUR))))
```

Make sure the discrimination is effective.

```
(loop for i from -11 to 11 collect
   (progn (setf (input CS-INT)
      (>thrifty-code i (nbre-input CS-INT) :mid 1))
      (winner-take-all (output (id (winner CS-INT))))))
(loop for i from 0 to 11 collect
   (progn (setf (input CS-DEG)
      (>thrifty-code i (nbre-input CS-DEG) :left 0))
      (winner-take-all (output (id (winner CS-DEG))))))
(loop for i from 1 to 8 collect
   (progn (setf (input CS-DUR)
      (>thrifty-code i (nbre-input CS-DUR) :right 1))
      (winner-take-all (output (id (winner CS-DUR))))))
```

The learning process at the area level consists to record all connected nodes from the cliques to the tournois according to the `cover-value` as the short term memory. Knowing that components of the corpus are partial sequences, only the onset state is enable.

```
(loop for seq in (mapcar #'mat-trans +corpus+)
   do
     (set-all-zeros CORPUS :mode :onset)
     (loop
        for i from 0 to (1- (length (car seq)))
        do
        (loop
          for s in (soms-list CORPUS)
          for d in seq do
          (setf (input (id s))
                (scaling (nth i d) :mlt (id s))))
        (learn CORPUS))
     (set-all-zeros CORPUS :mode :fine))
```

2.2 Sequencing instantiation

```
(create-sequencing
 :net CORPUS
 :dub '*voice1*
 :tag "VOICE1"
 :port 7771)
(set-rule *voice1*
  (make-list (length (soms-list (id (net self))))
    :initial-element '?))
(set-routine *voice1*
  (markov-chain self :odds #'rnd-weighted))
;; ... and play
(act-routine *voice1*)
;; ... and stop
(kill-routine *voice1*)
```

3 Study

→ https://github.com/yannics/Neuromuse3/blob/master/study/study.pdf