Virtuality in Neural Dynamical Systems*

Francesco Donnarumma, Roberto Prevete, Giuseppe Trautteur

Dipartimento di Scienze Fisiche,
Università di Napoli Federico II
Via Cintia, 80100 Napoli, Italy
{francesco.donnarumma,prevete,trau}@na.infn.it

Virtuality in computational systems

By virtuality in computational systems we understand the well known capability of interpreting, transforming, and running machines which are present, as it were, only under their code (program) aspect. In particular, distinctive features of the notion of virtuality, that directly take their origin in computability theory, are the capability for modifying machine code and the capability for simulating the behaviour of any member of some given class of computing machines by interpreting the stored code of the machine to simulate. Some motives for regarding virtuality - rather than discreteness and finiteness - as the more characteristic feature of computation are advanced in a recent paper by one of the authors [**?**]. In a physical entity, virtuality gives rise to an "as if" behavioural capability, based on the replacement of actual machinery by its (code) description.

Virtuality in the brain as a dynamical system

Dynamical systems approaches to the Central Nervous System (CNS) are now being vigorously pursued, either independently of, or jointly with, computational-algorithmic approaches. We address the question whether virtuality is present in *neural* dynamical systems, that is, in those special kinds of dynamical systems that are deemed to provide reasonable modelling tools for the CNS. We focus on Continuous Time Recurrent Neural Networks (CTRNN), building on recent work by Paine and Tani [1].

* Appeared in Proceedings of ICMP 2007: International Conference on Morphological Computation, March 26 – 28, Venice, Italy

The work described in the present paper is, in our view, an initial step towards full-fledged virtual behaviour in CTRNN, insofar as it achieves the controlled switching between different functionalities over the same fixed CTRNN structure. The CNS has up to now been modelled, via neural networks, as a special purpose architecture. But is the CNS machinery really "virtuality-free"? Mental arithmetic, introspection, and mind reading are significant examples of mental capabilities which seem to require some kind of virtual behaviour.

It has recently been argued by Dehaene [2] that the vast symbol processing capabilities of the CNS, including recursion involved in doing mental arithmetic, can be accounted for neither by evolutionary adaptation, nor by learning. He proposes an alternative "neuronal recycling" hypothesis, according to which brain architecture allows for the *rapid* development of new specific functionalities ([2], p.147) in brain areas previously devoted to performing other functional tasks.

The phenomenology of introspection includes so-called Higher Order Thoughts (HOT), or thinking about thinking (see [3], p.249 for a telling discussion). HOTs are widely discussed in Cognitive Science literature, but no suggestions concerning their actual brain implementation has been advanced. It seems hard to theorize about these symbolic processes without making appeal to a computational competence which includes full-fledged virtuality.

The so-called Theory of Mind (TOM) approach has been advanced to explain how one understands the intentions of conspecific individuals. TOM explanations involve a simulation of the other individual's behaviour starting from some sort of trigger recognition events.

Thus virtuality seems to be indispensable for such cognitive tasks as imagination, mind reading, planning, games, introspection, experiencing memories.

Historically prominent approaches to explaining this wide-range cognitive phenomenology [4] take their inspiration in the computer metaphor: one *assumes* that CNS activity somehow supports appropriate computational symbol processing capabilities but neglects the problem of explaining how these capabilities are physically realized in the CNS. This latter problem is instead crucial for the structural and functional modelling of the CNS by way of continuous dynamical

systems, in which the above capabilities are unnaturally simulated, if at all. A dynamical system endowed with the virtuality property, which we deem justifying by itself a computational interpretation, would be structurally close to the actual CNS as well as functionally close to its cognitive behaviour. This is, in our view, a powerful motivation to study virtuality in CTRNNs as a preliminary heuristic guide towards the experimental discovery and actual modelling of virtuality in the CNS.

CTRNN dynamical systems

CTRNNs are networks of biologically inspired neurons described by the following general vectorial equation [5,6]:

$$\boldsymbol{\tau}\frac{d\mathbf{y}}{dt} = -\mathbf{y} + \mathbf{W}\boldsymbol{\sigma}\left(\mathbf{y} - \boldsymbol{\theta}\right) + \mathbf{I} \tag{1}$$

where: $\boldsymbol{\tau}$ is the *membrane time constant*, $\mathbf{y}$ is the vector of *membrane potentials* after the deletion of *action potentials*, $\boldsymbol{\theta}$ is the *thresholds* vector, $\boldsymbol{\sigma}\left(\mathbf{x}\right)$ is the standard logistic activation function, $\mathbf{W}$ is the matrix of the *synaptic efficacies* and $\mathbf{I}$ is an independent input current.

We suppose that the input $\mathbf{I}$ can be partitioned into two: $\mathbf{I}^S$ and $\mathbf{I}^F$ deriving from two sets of distinct sources, possibly neurons. We make the assumption that *the firing rate* (*amplitude*) *of* $\mathbf{I}^S$ *changes slowly compared with the rate of* $\mathbf{I}^F$ (S for "slow", F for "fast") .

As is well known, dynamical systems described by equation (1) possess a number of limit sets [6]. We assume that any single computational task performed by the CTRNN consists in its convergence to its stable limit sets. We only consider cases in which the limit sets are *equilibrium points* and introduce *three time scales*:

- the fastest one, T, measures the approach towards a *stable* equilibrium point;
- the second time scale, T^F, is relative to the $\mathbf{I}^F$ input variability;
- the slowest one, T^S, is relative to the $\mathbf{I}^S$ input variability.

On the time scale T the input $\mathbf{I}^S$ and $\mathbf{I}^F$ can be considered constant while the network reaches a stable equilibrium point $\bar{\mathbf{s}}$. The stable

equilibrium points depend on $\mathbf{I}^S$ and $\mathbf{I}^F$. On the time scale T^F the input $\mathbf{I}^S$ can be considered constant. For each input $\mathbf{I}^F$ we take the equilibrium point $\bar{\mathbf{s}}$, reached on the time scale T, to be the network output. In other words the *stability surface* $\bar{\mathbf{s}} = \bar{\mathbf{s}}\left(\mathbf{I}^F; \mathbf{I}^S\right)$ of the CTRNN dynamical system, as a function of the network input $\mathbf{I}^F$ and parameterized by $\mathbf{I}^S$, is the *response function* (*program*):

$$\mathbf{f}_{I^S} \; : \; \mathbf{I}^F \in \mathbb{R}^N \longrightarrow \mathbf{f}_{I^S}\left(\mathbf{I}^F\right) \equiv \bar{\mathbf{s}}\left(\mathbf{I}^F; \mathbf{I}^S\right) \in \mathbb{R}^N$$

computed by the network on the time scale T^F. Of course if there is more than one stable equilibrium point then the response will also depend on initial conditions. The time scale T^S is relative to *the switching among different response functions* of the network, so that, on this time scale, different $\mathbf{I}^S$ values can force the network to compute different response functions on the time scale T^F.

An implemented example

We set up a simulation experiment concerning the switching between different response functions using the following scenario:

- a 2D mobile robot with 10 sonars as sensors and controlled by two parameters: the linear velocity v and the angular velocity ω around the vertical axis;
- a multiple $T-$maze as the robot environment;
- a twofold task of the robot: a) go forward along a corridor while avoiding possible obstacles, and b) turn left (right) at the next $T-$junction if a right (left) turn had been previously chosen. In other words we want the robot to proceed in the multiple $T-$maze through an alternation of turning choices.

Such task can be described by the following simple pseudocode:

```
BEGIN
leftTurn ← TRUE
WHILE (TRUE)
      IF (leftTurn = TRUE)
            leftTurn ← behaviourRight()
      ELSE
            leftTurn ← behaviourLeft()
      ENDIF
END WHILE
END
```

where behaviourRight() is a function (*program*) which controls the robot so as to make it a) go forward avoiding obstacles, b) turn right at a $T-$junction and return FALSE value; behaviourLeft() acts symmetrically.

Equivalently we give a CTRNN architecture, capable of running the two different response functions/programs - behaviourRight() and behaviourLeft() - on a fixed subnet, which controls the robot in order to achieve the above task. Developing ideas by Tani [1], we have *cabled* a two layered network:

- the first layer, L_1, is made up of seven neurons, that are initially fully inter-connected to each other. Five of these neurons directly receive input connections from the 10 sonars (our $\mathbf{I}^F$inputs). Two of these five neurons control the parameters v and ω respectively. The remaining two ones receive the $\mathbf{I}^S$ input. By setting the values on this input $\mathbf{I}^S$, this sub-network is capable of controlling the robot in *two different ways,* selecting the *two different programs*, beviourRight() and behaviourLeft();
- the second layer, L_2, is composed of a single self-connected neuron in such a way as to have two stable equilibrium points p_1 and p_2 (see [6]). The output of L_2 is the $\mathbf{I}^S$ input for the layer L_1. The output of the L_1 neuron controlling ω is given as $\mathbf{I}^F$to the L_2 neuron.

In order to set the $\mathbf{W}$ matrix of the above CTRNN, we have *trained* the layer to implement behaviourLeft() when $\mathbf{I}^S < 0$ and behaviourRight() when $\mathbf{I}^S > 0$. The second layer L_2 has been built so as

to select p_2 when $\omega < \Omega_1$ and p_1 when $\omega > \Omega_2$, where $\Omega_1 < \Omega_2$ are fixed thresholds. In other words the system runs program behaviourLeft() (behaviourRight()), when it "realizes" that the robot has turned right (left). This system was tested in the environment simulator *Player/Stage*. $\mathbf{I}^F$ is updated about every $10^2\,ms$ (the time scale T^F); the layer L_1 converges to a good approximation of the stable equilibrium point in about $10\,ms$ (the time scale T); the time between the switching of the programs is longer than $10^3 - 10^4\,ms$ (the time scale T^S). The entire system obtained by the composition of the two layers L_1 and L_2 succeded in controlling the robot inside various *different size* multiple $T-$maze environments.

Further steps towards actual virtuality

The example above demonstrates the achievement of a controlled switching of two different functionalities over the same physical structure (hardware) consisting of a fixed $\mathbf{W}$ matrix: i.e. the synaptic weights are left unaltered. This is, in our view, a significant step towards full-fledged virtual behaviour, insofar as the $\mathbf{I}^S$ inputs can be considered a *code* for the different behaviours and therefore can be intrerpreted as the equivalent, in a dynamical system environment, of a program in a computational environment. However, notice that L_1 was *trained*, not programmed. In order to obtain general program generation and interpretation capabilities, a structure consisting of a large number of interconnected CTRNN primitives may be envisaged. Within this structure definite subsets of those primitives might be able to output sets of $\mathbf{I}^S$ signals to different parts of the structure and these signals cause remaining parts of the structure to perform specified programs, not *a priori* fixed. The above statement is akin to requiring compositionality for CTRNNs, thereby envisaging the possibility of meeting criticisms of neural network approaches to cognitive modelling based on the principle of compositionality [7]. In this connection it is worth noting that different forms of compositionality are present in the above CTRNN: the aggregate of layers L_1 and L_2 in our example is structurally compositional in nature, while the two programs behaviourRight() and behaviourLeft() within layer L_1 exhibit a *novel kind of compositionality* to be more properly investigated and characterized in future works.

References

1. Rainer W. Paine and Jun Tani. Motor primitive and sequence self-organization in a hierarchical recurrent neural network. *Neural Networks*, 17:1291–1309, 2004.
2. Stanislas Dehaene. *Evolution of Human Cortical Circuits for Reading an Arithmetic: The "Neuronal Recycling" Hypothesis*, chapter 8, pages 133–157. Bradford MIT Press, 2005.
3. Edmund T. Rolls. *The Brain and Emotion*. Oxford U.P. New York, 1999.
4. Zenon W. Pylyshyn. *Computation and Cognition: Towards a Foundation for Cognitive Science*. MIT Press, 1984.
5. John J. Hopfield and David W. Tank. Computing with neural circuits: A model. *Science*, 233:625–633, 1986.
6. Randall D. Beer. On the dynamics of small continuous-time recurrent neural networks. *Adaptive Behavior*, 3(4):469–509, 1995.
7. Jerry A. Fodor and Zenon W. Pylyshyn. Connectionism and cognitive architecture: a critical analysis. *Cognition*, 28(1/2):3–71, 1988.