Elsevier Editorial System(tm) for Neurocomputing
Manuscript Draft

Manuscript Number:

Title: Processing with Cell Assemblies

Article Type: Special Issue: Artificial Brains

Keywords: Cell Assemblies; fLIF Neurons; Structured Program Theory

Corresponding Author: Dr Emma Byrne, Ph.D

Corresponding Author's Institution: Middlesex University

First Author: Emma Byrne, Ph.D

Order of Authors: Emma Byrne, Ph.D; Christian Huyck, PhD

Abstract: Cell assemblies (CAs) were posited by Hebb almost 60 years ago as the unit of representation in the brain. Recent results in the field of neuroscience indicate that CAs are likely to exist, at least in the mammalian brain. The CABot project uses simulations of CAs formed from individual neurons as a basis for learning and behaviour. This paper proves that a network of CAs, as described by Hebb and as implemented in CABot, is complete with respect to structured program theory. It follows that such a network is capable of executing any procedure that can be written as an algorithm.

Processing with Cell Assemblies

Emma Byrne[a,*], Christian Huyck[a]

[a]School of Engineering and Information Sciences, Middlesex University, The Burroughs, London NW4 4BT, United Kingdom

Abstract

Cell assemblies (CAs) were posited by Hebb almost 60 years ago as the unit of representation in the brain. Recent results in the field of neuroscience indicate that CAs are likely to exist, at least in the mammalian brain. The CABot project uses simulations of CAs formed from individual neurons as a basis for learning and behaviour. This paper proves that a network of CAs, as described by Hebb and as implemented in CABot, is complete with respect to structured program theory. It follows that such a network is capable of executing any procedure that can be written as an algorithm.

1. Introduction

The cell assembly (CA) has long been proposed as the basis of memory, or what Hebb called "The simplest instance of a representative process" [1, p 60]. CAs are sets of neurons that may be spatially distributed but that have high mutual interconnectivity. As a result, when a small subset of the neurons in a CA fire, activity tends to propagate to other members of the assembly. Firing in the assembly is sustained over a period of time by these same inter-connections. Recent biological findings indicate that CAs can be found in a number of organisms (e.g [2]).

CABot (the *C*ell *A*ssembly ro*Bot*) is a neurocognitive agent that operate in a virtual environment; the current agents (CABot1 and CABot2) operates in a simple computer game. These agents take commands from a user via natural language, and future agents will interact via dialogue. The neural basis of CABot is a network of model CAs, built from model fatiguing Leaky Integrate and Fire (fLIF) neurons.

CABot is entirely implemented in fLIF neurons (see Section 2 for details), and cell assemblies (CAs) emerge from these (see Section 3). Previous work has shown that the CA architecture can use variable bindings [3], store sequences of stimuli [4], and learn rules [5]. It has also been demonstrated that CAs can implement any finite state automata [6].

Structured program theory [7] demonstrates that any system that can carry out the operations of sequence, selection and iteration can execute any algorithm This paper will demonstrate that networks of model CAs are complete with respect to structured program theory. As a result, any algorithm can be implemented in a network of CAs that is connected as described in this paper.

This paper will: introduce the CA model; define three types of transition between CAs (sequence, selection and iteration); and will prove that CAs can be used to implement any program. Whilst this paper will give specific examples from the CABot agent, the definitions and proofs here are sufficiently general to apply to CAs in any suitably connected network, natural or artificial.

2. fLIF Neurons

The CABot architecture is built on the fatiguing Leaky Integrate and Fire (fLIF) neuron model, which is an idealised model of a biological neuron. The fLIF neuron is a simple, relatively biologically faithful extension of the Integrate and Fire (IF) neuron model [8, 9]. The fLIF model is efficient enough to enable 100,000 neurons to be simulated on a PC in real-time.

The IF neuron is a model of a spiking neuron: at a given timestep, if the activation that reaches the neuron passes a certain threshold, then the neuron fires. Maass and Bishop extended this model to include a leak component [10], based on the fact that some of the activation in a biological neuron 'leaks away' over time if the neuron does not fire. This model is more biologically accurate than the simple IF neuron, and it precludes firing caused by the accumulation of trivial amounts of activation over very long periods of time. The fatigue component [11] models the mechanism by which repeated firings lead to an increase in the threshold level of activation that a neuron must surpass in order to fire.

There are a number of biological features that the fLIF model does not address, such as the opening of ion transfer channels, or synaptic delays. These features are below the level of granularity required for this model of spiking behaviour. The fLIF neuron therefore represents processes that take place in around 10ms of biological time.

A fLIF neuron is described by three sets of equations that define:

1. Firing, in response to the integration of activation levels
2. The leaking of potentiation
3. The fatiguing of neurons due to their firing

The following sections review the IF, LIF and fLIF neuron models.

*Corresponding author
Email addresses: e.byrne@mdx.ac.uk (Emma Byrne), c.huyck@mdx.ac.uk (Christian Huyck)

2.1. Activation and firing

Model IF neurons integrate activation that is propagated from upstream (pre-synaptic) neurons. Let E_b be the level of activation energy for neuron b. If that activation reaches a threshold the neuron fires: it emits activation energy to neurons downstream. For notational convenience we define a firing flag that indicates whether a neuron fires at a given timestep.

Definition 2.1. Let θ be the firing threshold for a neuron. For simplicity we assume a universal firing threshold that is the same for all neurons, but this need not be the case. Neuron b *fires* at time t, and the *firing flag* ϕ_b is set as follows:

$$\begin{array}{lll} \text{if } E_b(t) \geq \theta & \text{then} & \phi_b(t) = 1 \\ & \text{else} & \phi_b(t) = 0 \end{array}$$

In an *IF neuron*, the level of activation in the neuron is a weighted sum of the number of presynaptic neurons that fire:

Definition 2.2. Let w_{ab} be the strength of the connection from neuron a to neuron b (which may have a negative value).

$$E_b(t) = \sum_{a=1}^{n} w_{ab} \times \phi_a(t-1)$$

Implicit in Definition 2.2 is the discrete nature of the model. All of the neurons have a chance to fire, and the activity is passed to other neurons for reintegration in the next cycle. No activity persists in the neuron from cycle to cycle.

2.2. Leak

The Leaky Integrate Fire (LIF) model is more biologically faithful than the IF model neuron. LIF neurons model the accumulation of activation in a neuron over time. The leak component models the observation that accumulated activation "leaks away". The leak component can be modeled such that the LIF neuron does not fire as a result of integrating trivial amounts of activation over a long time. If a neuron does not fire then the activation energy of that neuron at time t in a *LIF neuron* is the sum of activation from pre-synaptic neurons in the previous timestep and the leaky accumulation of activation over all previous timesteps.

Definition 2.3. Let $d > 1$ be a decay constant, which represents the leaking of activation from a neuron over time. Let b be a neuron and let t be a timestep.

$$\begin{aligned} \text{If } \phi_b(t) &= 0 \\ E_b(t+1) &= \frac{1}{d} E_b(t) + \sum_{a=1}^{n} w_{ab} \times \phi_a(t) \end{aligned}$$

However, if neuron b fires at t, all of its activation leaks away.

$$\begin{aligned} \text{If } \phi_b(t) &= 1 \\ E_b(t+1) &= \sum_{a=1}^{n} w_{ab} \times \phi_a(t) \end{aligned}$$

With the activation energy that a LIF neuron receives fixed at some constant value V, the total activation energy of that neuron is bounded:

$$\lim_{t \to \infty} E_a^T(t) = V \times \left(1 + \frac{1}{d-1}\right) \tag{1}$$

2.3. Fatigue

Immediately after firing, a biological neuron undergoes a brief refractory period of 2-3ms. This element of fatigue occurs at $<$ 10ms intervals and so is not represented in the model. After repeated firings, the neuron experiences longer term fatigue in which the response of that neuron diminishes. The higher the fatigue level, the lower the probability that the neuron will fire. In the fLIF model, this feature of the biological cell is modeled with a *fatigue level*, that uses a *fatigue constant* and a *fatigue recovery constant*. Thus, it is possible to model the reduction in the spiking rates of a fatigued neuron.

Definition 2.4. $F_b(t)$ is the *fatigue level* of neuron b at time t. Let F^r be a recovery constant that decreases the fatigue if a neuron does not fire. The overall fatigue level has a lower bound of 0.

$$\begin{aligned} \text{If } \phi_b(t) &= 0 \\ F_b(t) &= \max\{0, F_b(t-1) - F^r\} \end{aligned}$$

Let F^c be a fatigue constant, which increases the fatigue level if a neuron fires.

$$\begin{aligned} \text{If } \phi_b(t) &= 1 \\ F_b(t) &= F_b(t-1) + F^c \end{aligned}$$

Note that F^c and F^r are positive and may take identical values. The ratio between the fatigue constant and the fatigue recovery rate determines the maximum proportion of the neurons in a CA that may be firing on average at any time. For the purposes of the CABot model, all neurons in a network have the same values for F^c and F^r, with the value of $F^b(t)$ entirely dependent on the firing behaviour of neuron b in previous time steps.

Taking fatigue into account in the model, a neuron b fires at time t if and only if:

$$E_b(t) - F_b(t) \geq \theta \tag{2}$$

3. Cell assemblies

A CA is a set of neurons within a network that have high mutual synaptic strength. As a result when (relatively) few of the neurons in the assembly fire, mutually reinforcing activation tends to propagate to the rest of the CA. The CA will then 'reverberate', maintaining the activation pattern for a time, even in the absence of external stimuli. This reverberation serves not only to allow patterns of activity to persist, but also facilitates the strengthening of links between neurons, aiding learning.

Hebb first suggested the CA both as a support to learning and as the basic unit of neural processing thus: "[A] repeated

stimulation of specific receptors will lead slowly to the formation of an 'assembly' of association-area cells which can act briefly as a closed system after stimulation has ceased; this prolongs the time during which the structural changes of learning can occur and constitutes the simplest instance of a representative process (image or idea)." [1, pp 60].

Definition 3.1. A CA is a set of neurons that, through high mutual interconnection, maintain neural firing for a significant period of time when neurons outside the CA are not contributing to its activation.

The relationship between CA activation and neuron firing is not straightforward. Firstly, neurons in a CA may fire without the CA being active: if the set of currently firing neurons in a CA is insufficient to create sustained firing in other neurons in the assembly, then the CA is not active (see Definition 4.5). Secondly, it need not be the case that any particular neuron in a CA fires continuously, or indeed at all. During assembly activation, each cycle may see a different subset of the neurons in the assembly firing.

Each neuron may belong to multiple CAs, and cell assemblies may recruit new neurons via learning. In these proofs, we make the simplifying assumption that each neuron belongs to at least one cell assembly and that membership is fixed. This does not affect the generality of the definitions or proofs.

4. Neuron firings and CA ignitions

This section will define the types of activation that are found at the neuron level and at the CA level. The terms used in the following definitions are given in Table 1.

A, B	CAs (with or without subscripts). For the purposes of the following proofs, CAs can be treated as sets of neurons.
$a \in A, b \in B$	Individual neurons (with or without subscripts).
w_{ab}	Connection weight from neuron $a \in A$ to neuron $b \in B$.
$\phi_a(t)$	Firing flag: 1 if neuron a fires at timestep (t), 0 otherwise.
$\eta_A(t)$	Activation flag: 1 if CA A is active at timestep t and 0 otherwise.
X_A	Ignition threshold: the proportion of neurons in CA A that need to fire in a given cycle to start ignition.
p	Persistence window: the number of timesteps that intra-assembly activation must persist for the assembly to be active.

Table 1: The terms used in this paper.

4.1. Inter- and intra-assembly activation

Where there are weights between neurons in one cell assembly and another, activation spreads between CAs.

Definition 4.1. Let A and B be CAs and let $A \neq B$. There is *inter-assembly activation* from A to B at time t if and only if $\exists a \in A,\ \exists b \in B$ such that:

$$\begin{aligned} w_{ab} &\neq 0 \text{ and} \\ \phi_a(t) &= 1 \end{aligned}$$

Note that w_{ab} may be negative, that is, inter assembly activation includes inhibition as well as excitation.

Activation also spreads within a CA.

Definition 4.2. Let B be a CA. There is *intra-assembly activation* at time t if and only if $\exists b_i, b_j \in B$ such that:

$$\begin{aligned} w_{b_i b_j} &\neq 0 \text{ and} \\ \phi_{b_i}(t) &= 1 \end{aligned}$$

Note that $w_{b_i b_j}$ may be negative: intra-assembly activation includes the effect of inhibitory links.

Biological neurons integrate activation energy over a time window (modeled here as a number of discrete timesteps), from the firing of pre-synaptic neurons. In order to analyse the dynamics of a CA it is necessary to distinguish the activation energy a neuron receives from internal activation alone, from the activation energy a neuron receives from internal and external activation.

Definition 4.3. Let $E_b^I(t)$ be the *internal activation energy* of neuron b at time t. This activation energy comes from intra-assembly activation only.

$$\begin{aligned} E_b^I(t+1) &= \frac{1}{d} E_b^I(t) \\ &+ \sum_{b_i \in B} w_{b_i b} \times \phi_{b_i}(t) \end{aligned}$$

where $w_{b_i b_i} = 0$ for any i.

Definition 4.4. Let $E_b^T(t)$ be the *total activation energy* for neuron b at time step t. This activation comes from both intra- and inter-assembly activation.

$$\begin{aligned} E_b^T(t+1) &= \frac{1}{d} E_b^T(t) \\ &+ \sum_{b_i \in B} w_{b_i b} \times \phi_{b_i}(t) \\ &+ \sum_{n=1}^{N}, \sum_{a \in A_n} w_{ab} \times \phi_a(t) \end{aligned}$$

where $w_{b_i, b_i} = 0$ for any i and $A_n \neq B$.

4.2. Ignition, activation and extinguishing of CAs

Typically a CA ignites, is active for some time and then is extinguished: X_A is an upper bound on the proportion of neurons firing in A which is sufficient to sustain the firing of neurons in A by internal activation. Depending on interconnections, each cell assembly may have different values of X_A.

Definition 4.5. Let A be a CA and let $|A|$ be the number of neurons in A. X_A is the activation threshold for A.

$$X_A \leq \frac{\sum_{a \in A} \phi_a(t)}{|A|}$$

such that $\forall t_j \in [t, t+p]$:

$$\exists a \in A \text{ s.t. } E_a^I(t_j) \geq \theta$$

A CA is *active* whilst the number of neurons active is greater than the threshold X_A. The activation flag $\eta_A(t) = 1$ at time t, if and only if, the proportion of $a \in A$ that fire is greater than X_A at time (t). Otherwise $\eta_A(t) = 0$

A CA *ignites* at the first timestep in which the level of activation exceeds the threshold X_A, after a period of quiescence. These neurons may be receiving activation from outside the CA, but the CA is ignited only when the neurons in the CA are able to sustain activation in the absence of external input.

A CA becomes *extinguished* on the timestep that internal activation is no longer sufficient to fire at least some of its neurons for a time period p, due to lack of internal support, fatigue, inhibition, or a combination of these factors. Note that a CA can ignite and become immediately extinguished without ever becoming active, due to the presence of external inhibition.

5. CA processes

Structured program theory [7] proved that it is possible to decompose any program into three basic operators: sequence, selection and iteration. The following sections will demonstrate that an appropriately connected network of CAs, whether biological or synthetic, can implement these three operations. This result shows that CAs can, in theory, implement any known program. Examples of these operations in the CABot agents are given, but these definitions are general and apply to any suitably connected network of CAs.

For the sake of simplicity, in the definitions and proofs below, the postcondition of an operator is always a single CA. The purpose of this paper is to show that programs can be implemented with CAs. It is therefore sufficient to prove that there exist sequence, selection and iteration operations with single CA postconditions. These definitions and proofs also apply where pre- and postconditions are sets of CAs. Whilst sets of CAs would allow more efficient (fewer neurons) implementation of programs, proof is more complex. We leave these proofs to the interested reader.

The operators described here are restricted to sequence, selection and iteration. There are many other processes that a cell assembly may execute. For example, the ignition in turn of CAs A_1, A_2, A_1, A_4, A_1 is not a sequence, selection or iteration, but it can be implemented in a CA network. However, demonstrating that the sequence, selection and iteration operators can be implemented is sufficient to demonstrate that CA networks are complete with respect to structured programme theory, and so these other processes are not addressed in this paper.

For all definitions and proofs below, assume that only the listed CAs are active. Also assume that, aside from the connections mentioned in the proof, there are no other sources of activation for the postcondition CA. As a result, no other active CAs can lead to spurious activation or inhibition. Recall that (at least) X_A neurons fire for the duration that CA A is active.

5.1. Sequence

A network of CAs performs a *sequence* if the ignition of one CA at time t deterministically leads to the ignition of another, distinct, CA at time $t + \tau$ (Figure 1).

Definition 5.1. Let A and B be CAs. Let $\tau \geq 1$ be a time increment value. $\langle A \rightsquigarrow B \rangle$ is a *sequence step* if and only if $\forall t$:

$$\begin{aligned} \text{if } \eta_A(t) &= 1 \text{ it follows that:} \\ \eta_B(t+\tau) &= 1 \end{aligned}$$

Steps in a sequence can be concatenated to make longer sequences.

Definition 5.2. Let $\langle A_1 \rightsquigarrow A_2 \rangle, \cdots, \langle A_n \rightsquigarrow A_{n+1} \rangle$ be a set of sequence steps.

$$[\langle A_1 \rightsquigarrow A_2 \rangle, \cdots, \langle A_n \rightsquigarrow A_{n+1} \rangle]$$

is a sequence if and only if, for each subsequence of 2 consecutive steps, the postcondition of step n is the precondition of step $n+1$:

$$[\langle A_i \rightsquigarrow A_j \rangle, \langle A_j \rightsquigarrow A_l \rangle]$$

If the second CA in a step inhibits the first, leading to strictly feedforward activity, this is a strict sequence (Figure 1). Strict sequences are built of paired selection steps and suppression steps. A suppression step occurs when a CA deterministically causes another to extinguish.

Definition 5.3. Let A and B be CAs. Let $\tau \geq 1$ be an increment value. $\langle A \vdash B \rangle$ (A is *suppressed* by B) if and only if $\forall t$:

$$\begin{aligned} \text{if } \eta_B(t) &= 1 \text{ it follows that} \\ \eta_A(t+\tau) &= 0 \end{aligned}$$

In a strict sequence, each CA causes the previous CA to be extinguished (see Figure 1).

Definition 5.4. Let $S_A = [\langle A_1 \rightsquigarrow A_2 \rangle, \cdots, \langle A_n \rightsquigarrow A_{n+1} \rangle]$ be a sequence. If, for every sequence step $\langle A_i \rightsquigarrow A_j \rangle$ in this sequence S_A, $\langle A_i \vdash A_j \rangle$ then S_A is a strict sequence.

In the CABot1 system [12], the *Erase* network is a strict sequence. The Erase network is a timing sequence that selectively erases connections by allowing some weights to weaken whilst reinforcing others. There are 18 CAs in the Erase network, each of which fires in sequence, in which activation spreads from the first to the last CA over a number of timesteps. As CA n becomes active, it sends excitation downstream, and inhibition upstream, such that, after several timesteps, CA $n+1$ becomes active and CA n is extinguished.

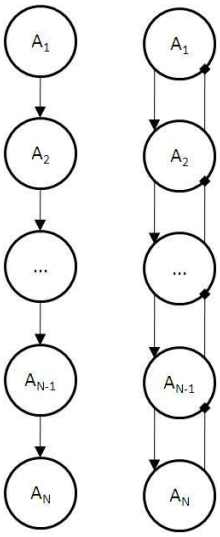

Figure 1: $A_1, \cdots, A_n$ are CAs. Lines with arrowheads denote excitatory connections. Lines with diamond heads denote inhibitory connections. On the left, a standard sequence $[\langle A_1 \rightsquigarrow A_2\rangle, \cdots, \langle A_{n-1} \rightsquigarrow A_n\rangle]$. On the right, a strict sequence $[\langle A_1 \rightsquigarrow A_2\rangle, \langle A_1 \vdash A_2\rangle, \cdots, \langle A_{n-1} \rightsquigarrow A_n\rangle, \langle A_{n-1} \vdash A_n\rangle]$.

Theorem 5.1. Any step in a sequence can be implemented by a network of CAs.

Proof. Assume $[\langle A \rightsquigarrow B\rangle]$ is a one-step sequence. Let the number of neurons in $A = |A|$ and in $B = |B|$. Let A be active at time step t. It follows that A has at least $|A| \times X_A$ neurons that are firing.

For each neuron in $b \in B$ let the fatigue level $F_b(t) \approx 0$. Let there be a connection from each neuron in each A to $|B| \times X_B$ neurons in B with a weight $\theta/(|A| \times X_A)$. At time $t+1$, $X_B \times |B|$ neurons in B will receive θ activation and will fire. As a result, B will ignite. □

Theorem 5.2. With the addition of inhibition, any sequence can be a *strict sequence*, for each step $\langle A \rightsquigarrow B\rangle$ there is a suppression step $\langle A \vdash B\rangle$ in which CA A is extinguished when CA B ignites.

Proof. Let there be a connection from every neuron in B to every neuron in A, such that $w_{b,a} \times -1 \gg \theta$. Whilst $\eta_B(t) = 1$, there are at least $|B| \times X_B$ neurons in B that are firing. Thus every neuron in A will receive $\gg \theta$ in inhibition. As a result, no neuron in A will fire. □

Theorem 5.3. A sequence of arbitrary length can be implemented using a sufficiently large net.

Proof. Let $[\langle A_1 \rightsquigarrow A_2\rangle, \cdots, \langle A_{n-1} \rightsquigarrow A_n\rangle]$ be a sequence of length $n-1$. To make a sequence of length n it is sufficient to concatenate this sequence with sequence step $\langle A_n \rightsquigarrow A_{n+1}\rangle$. □

The proofs for Theorems 5.1, 5.2 and 5.3 assume that no inter-CA connections exist, other than those defined in the *step* and *sequence* operators. Adding new connections between CAs deterministically implements changed state behaviour at the CA level.

5.2. Selection

Deterministic selection between CAs is also possible: the CA that will ignite next (the postcondition) is conditional on which set of CAs is currently active (the precondition) (see Figure 2). When assemblies A_1 and A_2 are active at the same time, assembly B_1 will ignite as a result. Neither A_1 nor A_2 is sufficient to ignite B_1 alone. Likewise, when assemblies A_3 and A_2 are active at the same time, assembly B_2 will ignite as a result. Neither A_3 nor A_2 is sufficient to ignite B_2 alone. In this selection, B_1 and B_2 may be active at the same time.

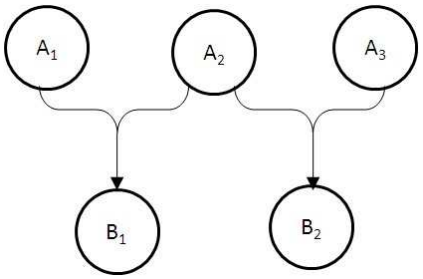

Figure 2: A selection in which CAs $A_1 \wedge A_2$ cause CA B_1 to ignite and CAs $A_2 \wedge A_3$ cause CA B_2 to ignite.

Definition 5.5. Let $\{A_i, \cdots, A_m\} \cap \{A_j, \cdots, A_n\} \neq \emptyset$ and let $\{A_i, \cdots, A_m\} \backslash \{A_j, \cdots, A_n\} \neq \emptyset$. Let $\tau \geq 1$ be an increment value.

$$\langle A_i \wedge \cdots \wedge A_m \rightsquigarrow B_m | A_j \wedge \cdots \wedge A_n \rightsquigarrow B_n\rangle$$

is a *selection* if and only if:

$$\begin{aligned} \forall t, A_x \in \{A_i, \cdots, A_m\}, \text{ if } \eta_{A_x}(t) &= 1 \text{ then} \\ \eta_{B_m}(t+\tau) &= 1 \end{aligned}$$

and

$$\begin{aligned} \forall t, A_x \in \{A_i, \cdots, A_m\}, \text{ if } \eta_{A_x}(t) &= 1 \text{ then} \\ \eta_{B_n}(t+\tau) &= 1 \end{aligned}$$

Theorem 5.4. Any selection can be made by a net with appropriate connection weights.

Proof. Let $\langle A_i \wedge \cdots \wedge A_m \rightsquigarrow B_m | A_j \wedge \cdots \wedge A_n \rightsquigarrow B_n\rangle$ be a selection. Let $m = |\{A_i, \cdots, A_m\}|$ and let $n = |\{A_j, \cdots, A_n\}|$.

Let there be a connection w_{a_x,b_m} from each neuron in each $A_x \in \{A_i, \cdots, A_m\}$ to $|B_m| \times X_{B_m}$ neurons in B_m such that:

$$w_{a_i,b_m} = \frac{\theta}{|A_x| \times X_{A_x}} \times \frac{1}{m}$$

If $\eta_{A_i}(t) = 1$ and ... and $\eta_{A_m}(t) = 1$ then $X_{B_m} \times |B_m|$ neurons in B_m will receive θ activation at time $t+1$ and will fire. As a result, $\eta_{B_m}(t+1) = 1$.

A_j, A_n, B_n may be substituted for A_i, A_m, B_m with the same result. □

With an appropriate ignition threshold X_A and decay constant d, the selection operation is sound, that is: no strict subset of the precondition CAs can cause the post condition CA to ignite due to integration of activation energy over a number of timesteps. Assuming no other sources of activation, simultaneous activation of all precondition cell assemblies is required for the postcondition cell assembly to ignite.

Theorem 5.5. Let $\langle A_i \wedge \cdots \wedge A_m \rightsquigarrow B_m | A_j \wedge \cdots \wedge A_n \rightsquigarrow B_n \rangle$ be a selection. $\forall A_x \in \{A_i, \cdots, A_m\}$ (likewise $\{A_j, \cdots, A_n\}$) let $X_{A_x} > \frac{m-1}{m}$ (likewise $\frac{n-1}{n}$). Let $d \approx \infty$. $\eta_{B_m}(t) = 1$ (likewise $\eta_{B_n}(t) = 1$) if and only if

$$\neg \exists A_x \in \{A_i, \cdots, A_m\}) \text{ s.t. } \eta A_x(t-1) = 0$$

Proof. Let $m = |\{A_i, \cdots, A_m\}|$ Let there be a connection from each neuron in $A_i \wedge \cdots \wedge A_m$ to $|B_m| \times X_{B_m}$ neurons in B_m with weights:

$$\left(\frac{\theta}{|A_i| \times X_{A_i}} \times \frac{1}{m}\right) \cdots \left(\frac{\theta}{|A_m| \times X_{A_m}} \times \frac{1}{m}\right)$$

Let $\{A_k, \cdots, A_l\} \subset \{A_i, \cdots, A_m\}$. The greatest level of activation that can be propagated to the neurons in B_m occurs when:

$$\begin{aligned} |\{A_k, \cdots, A_l\}| &= m-1 \text{ and} \\ \forall A_x \in \{A_k, \cdots, A_l\},\ X_{A_x} &\approx \frac{m-1}{m} \text{ and} \\ \forall a \in \{A_k, \cdots, A_l\},\ \phi_a(t-1) &= 1 \end{aligned}$$

Assuming no activation for any $b \in B_m$ at time $t-1$, $\forall b \in B_m$:

$$E_b^T(t) = \sum_{A_x \in \{A_k, \cdots, A_l\}} \left(\frac{\theta}{|A_x| \times X_{A_x}} \times |A_x| \times \frac{1}{m}\right)$$

Recall that $X_{A_x} > \frac{m-1}{m}$. It follows that:

$$E_b^T(t) < \sum_{A_x \in \{A_k, \cdots, A_l\}} \left(\frac{\theta}{|A_x| \times \frac{m-1}{m}} \times |A_x| \times \frac{1}{m}\right)$$

Also recall that $|\{A_k, \cdots, A_l\}| = m-1$. It follows that:

$$E_b^T(t) < (m-1) \times \left(\frac{m\theta}{m-1} \times \frac{1}{m}\right)$$

Or equivalently:

$$E_b^T(t) < \theta$$

Therefore no neuron in B_m receives sufficient activation to fire at time t. Given $d \approx \infty$, then $\forall b \in B_m$

$$\lim_{t_i \to \infty} E_b^T(t_i) < \theta \times \left(1 + \frac{1}{d}\right)$$

Or, as $limits_{d \to \infty}$

$$\lim_{t_i \to \infty} E_b^T(t_i) < \theta$$

□

If only one of the postcondition CAs can be active for more than one timestep then the selection is a strict selection (see Figure 3).

Definition 5.6. Let A_1, A_2, A_3 be precondition CAs. Let B_1 and B_2 be postcondition CAs. Let $\tau \geq 1$ be an increment value.

$$\langle A_1 \wedge A_2 \rightsquigarrow B_1 | A_2 \wedge A_3 \rightsquigarrow B_2 | B_2 \vdash B_1 \rangle$$

is a *strict selection* if and only if $\forall t$:

$$\begin{aligned} \text{if } \eta_{A_1}(t) = 1 \quad &\text{and} \quad \eta_{A_2}(t) = 1 \text{ it follows that:} \\ \eta_{B_1}(t+\tau) = 1 \quad &\text{and} \quad \eta_{B_2}(t+\tau) = 0 \end{aligned}$$

and

$$\begin{aligned} \text{if } \eta_{A_2}(t) = 1 \quad &\text{and} \quad \eta_{A_3}(t) = 1 \text{ it follows that:} \\ \eta_{B_1}(t+\tau) = 0 \quad &\text{and} \quad \eta_{B_2}(t+\tau) = 1 \end{aligned}$$

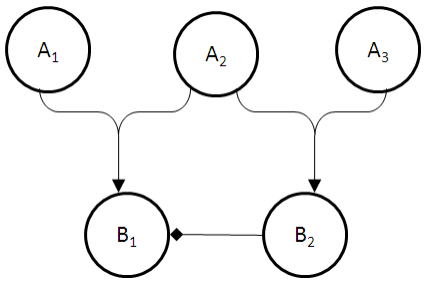

Figure 3: A selection in which CAs $A_1 \wedge A_2$ cause CA B_1 to ignite and CAs $A_2 \wedge A_3$ cause CA B_2 to ignite. In addition, CA B_2 inhibits the activation of CA B_1, preventing both B_1 and B_2 being simultaneusly active for more than one time step.

Theorem 5.6. Inhibition ensures that strict selection can be implemented in a net.

Proof. Let B_1 be a CA. Let $\langle A_1 \wedge A_2 \rightsquigarrow B_1 | A_2 \wedge A_3 \rightsquigarrow B_2 | B_2 \vdash B_1 | \rangle$ be a strict selection. Let there be a suppression step, $\langle B_2 \vdash B_1 \rangle$. Whilst $\eta B_1(t) = 1$, it will remain the case that $\eta B_2(t+1) = 0$. □

In the CABot1 agent there are several selection networks. For example, the stack top network either increments or decrements a counter depending on whether it is receiving activation from a "push" or a "pop" CA.

The "value" of the CA that is currently ignited determines the two values that may be reached next. That is, if the value of the stack top is i it may change to $i+1$ or $i-1$, but no other value, in the next time step.

Consider Figure 2 as an illustration of the stack top process. If the current stack top value is 2, represented by A_2, and the pop and push CAs are A_1 and A_3 respectively, it follows that B_1 is the stack top value of 1 and B_2 is the stack top value of 3. If A_1 and A_2 ("pop" and "stack top = 2") are active simultaneously then B_1 becomes active ("stack top = 1"). Activation in the push CA also sends activation to the $i+1$ CA, whilst activation in the pop CA sends activation to the $i-1$ CA. This is enough to cause one of the two stack top CAs to ignite.

5.3. Iteration

Iteration is the repeated execution of a sequence until a condition holds. As such, any it is possible to implement iteration by a combination of sequence and selection.

Figure 4 shows an iteration in which (a set of) CAs, A, is repeatedly activated whilst the condition $B_1 \wedge B_2$ holds. $B_1 \wedge B_2$ are repeatedly activated whilst A holds. C becomes active when $B_2 \wedge B_3$ hold. A ceases to be activated when condition C holds. The top and bottom sections of Figure 4 show the isolated selection and sequence elements of the iteration respectively.

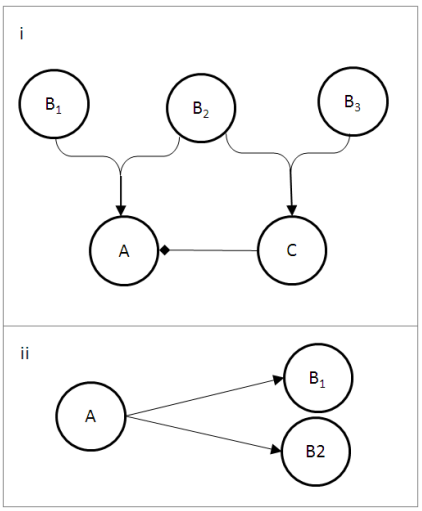

Figure 4: The selection (i) and sequence (ii) elements of an iteration. The sequence $[\langle A \rightsquigarrow B_1 \wedge B_2\rangle, \langle B_1 \wedge B_2 \rightsquigarrow A\rangle \cdots]$ continues until CA B_3 becomes active. Inhibitory connections between C and A shut down A when C becomes active, thus ending the iteration.

Definition 5.7. Let A, B_1, B_2, B_3 be CAs. An *iteration* exists if $[\langle A \rightsquigarrow B_1 \wedge B_2\rangle]$ is a sequence and $[\langle B_1 \wedge B_2 \rightsquigarrow A\rangle | \langle B_2 \wedge B_3 \rightsquigarrow C \; \langle | \langle A \vdash C \rangle]$ is a selection.

Theorem 5.7. A network of CAs can implement any iteration.

Proof. By Definition 5.7, an iteration can be implemented as a combination of sequence and selection operations. By Theorem 5.3, a network of CAs can implement any sequence. By Theorem 5.6, a network of CAs can implement any selection. It follows that a network of CAs can implement any iteration. □

As any sequence, selection, and iteration can be implemented in a large enough network of neurons with CAs, any program can be implemented [7].

6. Related models

6.1. The Hopfield Network

CABot is not the first project to model CAs. One existing model of CAs is the Hopfield network [9] which uses integrate and fire neurons (see Section 2) and a well connected net. Hopfield networks can store patterns using a calculation that is a variant of the Hebbian learning rule, implementing a type of autoassociative memory: if a pattern is presented that is near to a stored pattern, the network will settle into the stored pattern. In order to move to stable states, a Hopfield network is implemented with bidirectional connections. A Hopfield stable state is consistent with the definition of a CA. However, once it has settled into one stable state, the Hopfield model cannot move into another. As a result, Hopfield networks are unable to implement strict sequence, strict selection or iteration.

Furthermore, the Hopfield model lacks some elements of biological plausibility. Neurons can be more or less central to a CA; this is not the case in the Hopfield model. Moreover the brain is not well-connected. In contrast, the CABot model has higher biological fidelity. The biologically plausible connections in the CABot model allow it to carry out processing using the sequence, selection and iteration operators defined in this paper.

6.2. Models of neuronal dynamics

This paper has presented three operations that can arise from the neuro-dynamics of a network of CAs. However, There are other existing models of neuronal dynamics, such as the synfire chain [13, 14, 15, 16] and the "neuronal avalanche" model [17]. Whilst these models describe the tightly synchronised patterns that emerge from the firing of synaptically connected neurons, they do not explain the higher level processes (sequence, selection and iteration) that are defined here.

The neuronal avalanche model is based on observations of activation that spreads between neurons in tightly synchronised repeating pattern. Plenz and Thiagrajan [17] propose the neuronal avalanche as a method of propagating activity *within* CAs and, as such, it provides an explanation for how CAs sustain activity over time. However, the model is not concerned with the processes that emerge as activation spreads from assembly to assembly.

Synfire chains were proposed by Abeles [13] and have been modeled in a number of ways, both mathematically [14, 15] and biologically [16]. Synfire chains are precisely timed sequences of firing activity in pools of neurons, such that each neuron in one pool has excitatory connections to many neurons in the next pool. There are few, if any, lateral connections between neurons in the same pool. As a result, activity at the beginning of the chain either fades away or, if propagated, is propagated as a synchronous wave. As a result stereotypical dynamics arise from stochastic synapses [16].

Synfire chains are essentially feed forward in nature: activity in one pool propagates to the next pool with little support from intra-pool connectivity. CA processes on the other hand arise from both inter- and intra-assembly connectivity. Inter-assembly connectivity propagates activation from assembly to assembly, but in the absence of spontaneous activation, it is intra-assembly connectivity that causes CAs to ignite.

Synfire chains describe the detailed temporal dynamics of a network of neurons, whereas CA processing is concerned only with the *order* in which activity is propagated. CA processes are also able to implement the three basic operations of process flow: sequence, selection and iteration, whereas synfire chains model only the sequence of activity as it moves from pool to pool.

7. Conclusion

Structured program theory, first presented in [7], demonstrated that any program can be written using only three operators: sequence, selection and iteration. We have demonstrated that a biologically plausible model of neural architecture, the CA, is able to carry out those three operations. As a result, any algorithm can be implemented using a suitably connected network of CAs.

This is no surprise. Elsewhere, it has been shown that CAs can implement any finite state automata [6], and that they can implement stacks [18]. Consequently, they are Turing complete. The continuous version of this model is Super-Turing complete [19]. This paper has shown in addition that CAs are complete with respect to structured program theory, and has therefore shown how any structured program can be directly implemented as a network of CAs.

This paper has given proofs on deterministic models. While deterministic models are a subset of stochastic models (where all randomness is removed), it is clear that many models, including the likely actual biological mechanism, are stochastic. Depending on the degree of redundancy and randomness, these systems will vary in their programmatic faithfulness, with operations no longer being certain, but more or less likely. Nevertheless, the central finding still holds: that suitably connected networks of CAs can carry out processes.

Moreover, the strength of neural systems is not in their ability to implement any program, though there is something to be said about parallel implementation. Instead, the strength lies with the ability to learn these processes and basic symbols. The implementation of programs in neurons can be a useful bridge between the way actual biological neural systems are implemented and our current knowledge of program implementation.

Whilst we do not claim that behaviour arises from these three operators, this result demonstrates that the CA is not only what Hebb called a “conscious content” but is also a plausible component in processes that involve moving from state to state.

Acknowledgements: The authors would like to thank Roman Belavkin and David Corney for their constructive feedback during the preparation of this paper. This work was supported by EPSRC grant GR/R13975/01.

References

[1] D. O. Hebb, The Organization of Behaviour, J Wiley and Sons, 1949.
[2] Y. Sakurai, How do cell assemblies encode information in the brain?, Neuroscience & Biobehavioral Reviews 23 (6) (1999) 785–796.
[3] C. Huyck, R. Belavkin, Counting with neurons: Rule application with nets of fatiguing leaky integrate and fire neurons, in: The 7th International Conference on Cognitive Modeling, 2006.
[4] H. Ghalib, C. Huyck, A cell assembly model of sequential memory, in: Neural Networks, 2007. IJCNN 2007. International Joint Conference on, 2007, pp. 625–630.
[5] R. Belavkin, C. Huyck, The emergence of rules in cell assemblies of fLIF neurons, in: Proceedings of the Eighteenth European Conference on Artificial Intelligence, 2008.
[6] Y. Fan, C. Huyck, Implementation of finite state automata using fLIF neurons, in: IEEE Systems, Man and Cybernetics Society, 2008, pp. 74–78.
[7] C. Böhm, G. Jacopini, Flow diagrams, turing machines and languages with only two formation rules, Commun. ACM 9 (5) (1966) 366–371.
[8] W. McCulloch, W. Pitts, A logical calculus of the ideas immanent in nervous activity, Bulletin of Mathematical Biology 5 (4) (1943) 115–133.
[9] J. J. Hopfield, Neural networks and physical systems with emergent collective computational abilities, PNAS 79 (8) (1982) 2554–2558.
[10] W. Maass, C. M. Bishop, Pulsed Neural Networks, MIT Press, 2001.
[11] S. Kaplan, M. Sonntag, E. Chown, Tracing recurrent activity in cognitive elements (TRACE): a model of temporal dynamics in a cell assembly - connection science, Connection Science 3 (2) (1991) 179–206.
[12] C. Huyck, CABot1: a videogame agent implemented in fLIF neurons, in: IEEE Systems, Man and Cybernetics Society, 2008, pp. 115–120.
[13] M. Abeles, Corticonics: Neural Circuits of the Cerebral Cortex, Cambridge University Press, 1991.
[14] T. Wennekers, Dynamics of spatio-temporal patterns in associative networks of spiking neurons, In ICANN99: Ninth International Conference on Artificial Neural Networks. Institute of Electrical Engineers 32 (2000) 597—602.
[15] G. Hayon, M. Abeles, D. Lehmann, A model for representing the dynamics of a system of synfire chains, Journal of Computational Neuroscience 18 (1) (2005) 41–53.
[16] Y. Ikegaya, G. Aaron, R. Cossart, D. Aronov, I. Lampl, D. Ferster, R. Yuste, Synfire chains and cortical songs: Temporal modules of cortical activity, Science 304 (5670) (2004) 559–564.
[17] D. Plenz, T. C.Thiagarajan, The organizing principles of neuronal avalanches: cell assemblies in the cortex?, Trends in Neurosciences 30 (3) (2007) 101–110.
[18] C. Huyck, Y. Fan, Parsing with fLIF neurons, in: Proceedings of Advances in Cybernetic Systems, 2007, Dublin.
[19] H. T. Siegelmann, E. D. Sontag, On the computational power of neural nets, Tech. rep., SYCON (1991).

*** Biography of the author(s)**

Click here to download Biography of the author(s): Bios.txt

Dr Emma Byrne received her PhD from University College London in 2005. She joined Middlesex University in 2008 where she is currently a Research Assistant on the CABot project. From 2006 - 2008 she was employed as a Research Associate at Aberystwyth University on the Robot Scientist project. Her work has focused on logical models for artificial intelligence, and on optimisation of closed loop learning for the Robot Scientist.

Dr Christian Huyck received his PhD from the University of Michigan in 1994. He has been at Middlesex University since 1997, and is currently the Reader in Artificial Intelligence. His work has focused on Natural Language Processing, more broadly Artificial Intelligence, and recently in neural models. Basing his work around Cell Assemblies, he has developed systems for real world categorisation tasks, and for theoretical implementations of a range of behaviours including sequence, variable binding, and rule learning. He is primary investigator on the CABot project, funded by EPSRC, that brings these strands together.

*** Photo of Emma Byrne**

Click here to download high resolution image

*** Photo of Chris Huyck**

Click here to download high resolution image