Economic Attention Networks: Associative Memory and Resource Allocation for General Intelligence

Matthew Ikle', Joel Pitt, Ben Goertzel, George Sellman

Adams State College (ASC), Singularity Institute for AI (SIAI), Novamente LLC and SIAI, ASC
1405 Bernerd Place, Rockville MD 20851, USA
ben@goertzel.org, stephan@bugaj.com

Abstract

A novel method for simultaneously storing memories and allocating resources in AI systems is presented. The method, Economic Attention Networks (ECANs), bears some resemblance to the spread of activation in attractor neural networks, but differs via explicitly differentiating two kinds of "activation" (Short Term Importance, related to processor allocation; and Long Term Importance, related to memory allocation), and in using equations that are based on ideas from economics rather than approximative neural modeling. Here we explain the basic ideas of ECANs, and then investigate the functionality of ECANs as associative memories, via mathematical analysis and the reportage of experimental results obtained from the implementation of ECANs in the OpenCog integrative AGI system.

Introduction

One of the critical challenges confronting any system aimed at advanced general intelligence is the allocation of computational resources. The central nature of this issue is highlighted by Hutter's (2004) mathematical results showing that if one formalizes intelligence as the achievement of complex computable goals, then there are very simple software programs that can achieve arbitrarily high degrees of intelligence, so long as they are allotted huge amounts of computational resources. In this sense, coping with space and time limitations is the crux of the AGI problem.

Not surprisingly, given its central nature, the management of computational resources ties in with a variety of other concrete issues that AGI systems confront, in ways depending on the specific system in question. In the approach we will describe here, resource allocation is carried out by the same structures and dynamics as associative memory, whereas the relationship between resource allocation and other system processes like reasoning and procedure learning involves feedback between distinct software components.

We will describe here a specific approach to resource allocation and associative memory, which we call Economic Attention Networks or ECANs. ECANs have been designed and implemented within an integrative AGI framework called OpenCog (which overlaps with the related Novamente Cognition Engine system; see Goertzel, 2006). However, ECANs also have meaning outside the OpenCog context; they may be considered nonlinear dynamical systems in roughly the same family as attractor neural networks such as Hopfield nets (Amit, 1992). The main focus of this paper is the study of ECANs as associative memories, which involves mathematical and experimental analyses that are independent of the embedding of ECANs in OpenCog or other AGI systems. But we will also discuss the implications of these results for specific interactions between ECANs and other OpenCog components

Economic Attention Networks

First we summarize the essential ideas of ECANs; in later sections two specific variants of ECAN equational formalizations are presented.

An ECAN is a graph, consisting of un-typed nodes and links, and also links that may be typed either HebbianLink or InverseHebbianLink. It is also useful sometimes to consider ECANs that extend the traditional graph formalism and involve links that point to links as well as to nodes. The term Atom will be used to refer to either nodes or links. Each Atom in an ECAN is weighted with two numbers, called STI (short-term importance) and LTI (long-term importance). Each Hebbian or InverseHebbian link is weighted with a probability value.

The equations of an ECAN explain how the STI, LTI and Hebbian probability values get updated over time. The metaphor underlying these equations is the interpretation of STI and LTI values as (separate) artificial currencies. The motivation for this metaphor has been elaborated somewhat in (Goertzel, 2007) and will not be recapitulated here. The fact that STI (for instance) is a currency means that the total amount of STI in the system is conserved (except in unusual instances where the ECAN controller

decides to introduce inflation or deflation and explicitly manipulate the amount of currency in circulation), a fact that makes the dynamics of an ECAN dramatically different than that of, say, an attractor neural network (in which there is no law of conservation of activation).

Conceptually, the STI value of an Atom is interpreted to indicate the immediate urgency of the Atom to the ECAN at a certain point in time; whereas the LTI value of an Atom indicates the amount of value the ECAN perceives in the retention of the Atom in memory (RAM). An ECAN will often be coupled with a "Forgetting" process that removes low-LTI Atoms from memory according to certain heuristics.

STI and LTI values will generally vary continuously, but the ECAN equations we introduce below contain the notion of an AttentionalFocus (AF), consisting of those Atoms in the ECAN with the highest STI values. The AF is given its meaning by the existence of equations that treat Atoms with STI above a certain threshold differently.

Conceptually, the probability value of a HebbianLink from A to B is the odds that if A is in the AF, so is B; and correspondingly, the InverseHebbianLink from A to B is weighted with the odds that if A is in the AF, then B is not. A critical aspect of the ECAN equations is that Atoms periodically spread their STI and LTI to other Atoms that connect to them via Hebbian and InverseHebbianLinks; this is the ECAN analogue of activation spreading in neural networks.

Based on the strong correspondences, one could plausibly label ECANs as "Economic Neural Networks"; however we have chosen not to go that path, as ECANs are not intended as plausible neural models, but rather as nonlinear dynamical systems engineered to fulfill certain functions within non-brain-emulative AGI systems.

Integration into OpenCog and the NCE

The OpenCog AGI framework, within which the current ECAN implementation exists, is a complex framework with a complex underlying theory, and here we will only hint at some of its key aspects. OpenCog is an open-source software framework designed to support the construction of multiple AI systems; and the current main thrust of work within OpenCog is the implementation of a specific AGI design called OpenCogPrime (OCP), which is presented in the online wikibook (Goertzel, 2008). Much of the OpenCog software code, and many of the ideas in the OCP design, have derived from the open-sourcing of aspects of the proprietary Novamente Cognition Engine, which has been described extensively in previous publications.

The first key entity in the OpenCog software architecture is the AtomTable, which is a repository for weighted, labeled hypergraph nodes and hyperedges. In the OpenCog implementation of ECANs, the nodes and links involved in the ECAN are stored here. OpenCog also contains an object called the CogServer, which wraps up an AtomTable as well as (among other objects) a Scheduler that schedules a set of MindAgent objects that each (when allocated processor time by the Scheduler) carry out cognitive operations involving the AtomTable. The essence of the OCP design consists of a specific set of MindAgents designed to work together in a collaborative way in order to create a system that carries out actions oriented toward achieving goals (where goals are represented as specific nodes in the AtomTable, and actions are represented as Procedure objects indexed by Atoms in the AtomTable, and the utility of a procedure for achieving a goal is represented by a certain set of probabilistic logical links in the AtomTable, etc.). OpenCog is still at an experimental stage but has been used for such projects as statistical language analysis, probabilistic inference, and the control of virtual agents in online virtual worlds (see opencog.org).

So, in an OpenCog context, ECAN consists of a set of Atom types, and then a set of MindAgents carrying out ECAN operations such as HebbianLinkUpdating and ImportanceUpdating. OCP also requires many other MindAgents carrying out other cognitive processes such as probabilistic logical inference according to the PLN system (Goertzel et al, 2008) and evolutionary procedure learning according to the MOSES system (Looks, 2006). The interoperation of the ECAN MindAgents with these other MindAgents is a subtle issue that will be briefly discussed in the final section of the paper, but the crux is simple to understand.

The CogServer is understood to maintain a kind of central bank of STI and LTI funds. When a non-EAN MindAgent finds an Atom valuable, it sends that Atom a certain amount of Stimulus, which results in that Atom's STI and LTI values being increased (via equations to be presented below, that transfer STI and LTI funds from the CogServer to the Atoms in question). Then, the ECAN ImportanceUpdating MindAgent carries out multiple operations, including some that transfer STI and LTI funds from some Atoms back to the CogServer.

Definition and Analysis of Variant 1

We now define a specific set of equations in accordance with the ECAN conceptual framework described above. We define $\mathbf{H}_{STI} = [s_1, \cdots, s_n]$ to be the vector of STI values, and $\mathbf{C} = \begin{bmatrix} c_{11}, \cdots, c_{1n} \\ \vdots \quad \ddots \quad \vdots \\ c_{n1}, \cdots, c_{nn} \end{bmatrix}$ to be the connection matrix of Hebbian probability values, where it is assumed that the existence of a HebbianLink or InverseHebbianLink between A and B are mutually exclusive possibilities. We also define $\mathbf{C}_{LTI} = \begin{bmatrix} g_{11}, \cdots, g_{1n} \\ \vdots \quad \ddots \quad \vdots \\ g_{n1}, \cdots, g_{nn} \end{bmatrix}$ to be the matrix of LTI values for each of the corresponding links.

We assume an updating scheme in which, periodically, a number of Atoms are allocated Stimulus amounts, which

causes the corresponding STI values to change according to the equations

$$\forall i: \quad s_i = s_i - \text{rent} + \text{wages},$$

where rent and wages are given by

$$\text{rent} = \begin{cases} \langle \text{Rent} \rangle \cdot \max\left(0, \dfrac{\log\left(\dfrac{20 s_i}{\text{recentMaxSTI}}\right)}{2}\right), & \text{if } s_i > 0 \\ 0, & \text{if } s_i \le 0 \end{cases}$$

and

$$\text{wages} = \begin{cases} \dfrac{\langle \text{Wage} \rangle \langle \text{Stimulus} \rangle}{\sum_{i=1}^{n} p_i}, & \text{if } p_i = 1 \\ \dfrac{\langle \text{Wage} \rangle \langle \text{Stimulus} \rangle}{n - \sum_{i=1}^{n} p_i}, & \text{if } p_i = 0 \end{cases},$$

where $\mathbf{P} = [p_1, \cdots, p_n]$, with $p_i \in \{0,1\}$ is the cue pattern for the pattern that is to be retieved.

All quantities enclosed in angled brackets are system parameters, and LTI updating is accomplished using a completely analogous set of equations.

The changing STI values then cause updating of the connection matrix, according to the "conjunction" equations. First define

$$\text{norm}_i = \begin{cases} \dfrac{s_i}{\text{recentMaxSTI}}, & \text{if } s_i \ge 0 \\ \dfrac{s_i}{\text{recentMinSTI}}, & \text{if } s_i < 0 \end{cases}.$$

Next define

$$\text{conj} = \text{Conjunction}(s_i, s_j) = \text{norm}_i \times \text{norm}_j$$

and

$$c'_{ij} = \langle \text{ConjDecay} \rangle \, \text{conj} + (1 - \text{conj}) c_{ij}.$$

Finally update the matrix elements by setting

$$c_{ij} = \begin{cases} c_{ji} = c'_{ij}, & \text{if } c'_{ij} \ge 0 \\ c'_{ij}, & \text{if } c'_{ij} < 0 \end{cases}.$$

We are currently also experimenting with updating the connection matrix in accordance with the equations suggested by Storkey (1997, 1998, 1999.)

A key property of these equations is that both wages paid to, and rent paid by, each node are positively correlated to their STI values. That is, the more important nodes are paid more for their services, but they also pay more in rent.

A fixed percentage of the links with the lowest LTI values is then forgotten (which corresponds equationally to setting the LTI to 0).

Separately from the above, the process of Hebbian probability updating is carried out via a diffusion process in which some nodes "trade" STI utilizing a diffusion matrix **D**, a version of the connection matrix **C** normalized so that **D** is a left stochastic matrix. **D** acts on a similarly scaled vector **v**, normalized so that **v** is equivalent to a probability vector of STI values.

The decision about which nodes diffuse in each diffusion cycle is carried out via a decision function. We currently are working with two types of decision functions: a standard threshold function, by which nodes diffuse if and only if the nodes are in the AF; and a stochastic decision function in which nodes diffuse with probability $\dfrac{\tanh(\text{shape}(s_i - \text{FocusBoundary})) + 1}{2}$, where shape and FocusBoundary are parameters.

The details of the diffusion process are as follows. First, construct the diffusion matrix from the entries in the connection matrix as follows:

$$\text{If } c_{ij} \ge 0, \text{ then } d_{ij} = c_{ij},$$
$$\text{else, set } d_{ji} = -c_{ij}.$$

Next, we normalize the columns of **D** to make **D** a left stochastic matrix. In so doing, we ensure that each node spreads no more that a $\langle \text{MaxSpread} \rangle$ proportion of its STI, by setting

$$\text{if } \sum_{i=1}^{n} d_{ij} > \langle \text{MaxSpread} \rangle:$$

$$d_{ij} = \begin{cases} d_{ij} \times \dfrac{\langle \text{MaxSpread} \rangle}{\sum_{i=1}^{n} d_{ij}}, & \text{for } i \ne j \\ d_{jj} = 1 - \langle \text{MaxSpread} \rangle \end{cases}$$

else:

$$d_{jj} = 1 - \sum_{\substack{i=1 \\ i \ne j}}^{n} d_{ij}$$

Now we obtain a scaled STI vector **v** by setting

$$\text{minSTI} = \min_{i \in \{1,2,\cdots,n\}} s_i \text{ and } \text{maxSTI} = \max_{i \in \{1,2,\cdots,n\}} s_i$$

$$v_i = \frac{s_i - \text{minSTI}}{\text{maxSTI} - \text{minSTI}}$$

The diffusion matrix is then used to update the node STIs

$$\mathbf{v}' = \mathbf{D}\mathbf{v}$$

and the STI values are rescaled to the interval $[\text{minSTI}, \text{maxSTI}]$.

In both the rent and wage stage and in the diffusion stage, the total STI and LTI funds of the system each separately form a conserved quantity: in the case of diffusion, the vector **v** is simply the total STI times a probability vector. To maintain overall system funds within homeostatic bounds, a mid-cycle tax and rent-adjustment can be triggered if necessary; the equations currently used for this are

- $\langle \text{Rent} \rangle = \frac{\text{recent stimulus awarded before update} \times \langle \text{Wage} \rangle}{\text{recent size of AF}}$;
- $\text{tax} = \frac{x}{n}$, where x is the distance from the current AtomSpace bounds to the center of the homeostatic range for AtomSpace funds;
- $\forall i: \; s_i = s_i - \text{tax}$

Investigation of Convergence Properties

Now we investigate some of the properties that the above ECAN equations display when we use an ECAN defined by them as an associative memory network in the manner of a Hopfield network.

We consider a situation where the ECAN is supplied with memories via a "training" phase in which one imprints it with a series of binary patterns of the form $\mathbf{P} = [p_1, \cdots, p_n]$, with $p_i \in \{0,1\}$. Noisy versions of these patterns are then used as cue patterns during the retrieval process.

We obviously desire that the ECAN retrieve the stored pattern corresponding to a given cue pattern. In order to achieve this goal, the ECAN must converge to the correct fixed point.

Theorem: For a given value of *e* in the STI rent calculation, there is a subset of hyperbolic decision functions for which the ECAN dynamics converge to an attracting fixed point.

Proof: Rent is zero whenever $s_i \le \frac{\text{recentMaxSTI}}{20}$, so we consider this case first. The updating process for the rent and wage stage can then be written as $f(s) = s + \text{constant}$. The next stage is governed by the hyperbolic decision function

$$g(s) = \frac{\tanh(\text{shape}(\text{s} - \text{FocusBoundary})) + 1}{2}.$$

The entire updating sequence is obtained by the composition $(g \circ f)(s)$, whose derivative is then

$$(g \circ f)' = \frac{\text{sech}^2(\text{f(s)}) \cdot \text{shape}}{2} \cdot (1),$$

which has magnitude less than 1 whenever -2 < shape < 2. We next consider the case $s_i > \frac{\text{recentMaxSTI}}{20}$. The function f now takes the form

$$f(s) = s - \frac{\log(20s/\text{recentMaxSTI})}{2} + \text{constant},$$

and we have

$$(g \circ f)' = \frac{\text{sech}^2(\text{f(s)}) \cdot \text{shape}}{2} \cdot \left(1 - \frac{1}{s}\right).$$

which has magnitude less than 1 whenever $|\text{shape}| < \left|\frac{2 \cdot \text{recentMaxSTI}}{\text{recentMaxSTI} - 20}\right|$. Choosing the shape parameter to satisfy $0 < \text{shape} < \min\left(2, \left|\frac{2 \cdot \text{recentMaxSTI}}{\text{recentMaxSTI} - 20}\right|\right)$ then guarantees that $\left|(g \circ f)'\right| < 1$. Finally, g∘f maps the closed interval [recentMinSti, recentMaxSTI] into itself, so applying the Contraction Mapping Theorem completes the proof.

Definition and Analysis of Variant 2

The ECAN variant described above has performed completely acceptably in our experiments so far; however we have also experimented with an alternate variant, with different convergence properties. In Variant 2, the dynamics of the ECAN are specifically designed so that a certain conceptually intuitive function serves as a Liapunov function of the dynamics.

At a given time t, for a given Atom indexed i, we define two quantities: *$OUT_i(t)$ = the total amount that Atom i pays in rent and tax and diffusion during the time-t iteration of ECAN ; $IN_i(t)$ = the total amount that Atom i receives in diffusion, stimulus and welfare during the time-t iteration of ECAN*. Note that welfare is a new concept to be introduced below. We then define $\text{DIFF}_i(t) = |\text{IN}_i(t) - \text{OUT}_i(t)|$; and define AVDIFF(t) as the average of $\text{DIFF}_i(t)$ over all i in the ECAN.

The design goal of Variant 2 of the ECAN equations is to ensure that, if the parameters are tweaked appropriately, AVDIFF can serve as a (deterministic or stochastic, depending on the details) Liapunov function for ECAN dynamics. This implies that with appropriate parameters the ECAN dynamics will converge toward a state where AVDIFF=0, meaning that no Atom is making any profit or incurring any loss. It must be noted that this kind of convergence is not always desirable, and sometimes one might want the parameters set otherwise. But if one wants the STI components of an ECAN to converge to some

specific values, as for instance in a classic associative memory application, Variant 2 can guarantee this easily.

In Variant 2, each ECAN cycle begins with rent collection and welfare distribution, which occurs via collecting rent via the Variant 1 equation, and then performing the following two steps. Step A: calculate X, defined as the positive part of the total amount by which AVDIFF has been increased via the overall rent collection process. Step B: redistribute X to needy Atoms as follows: *For each Atom z, calculate the positive part of (OUT - IN), defined as deficit(z). Distribute (X + e) wealth among all Atoms z, giving each Atom a percentage of X that is proportional to deficit(z), but never so much as to cause OUT < IN for any Atom (the welfare being given counts toward IN).* Here e>0 ensures AVDIFF decrease; e=0 may be appropriate if convergence is not required in a certain situation.

Step B is the welfare step, which guarantees that rent collection will decrease AVDIFF. Step A calculates the amount by which the rich have been made poorer, and uses this to make the poor richer. In the case that the sum of deficit(z) over all nodes z is less than X, a mid-cycle rent adjustment may be triggered, calculated so that step B will decrease AVDIFF. (I.e. we cut rent on the rich, if the poor don't need their money to stay out of deficit.)

Similarly, in each Variant 2 ECAN cycle, there is a wage-paying process, which involves the wage-paying equation from Variant 1 followed by two steps. Step A: calculate Y, defined as the positive part of the total amount by which AVDIFF has been increased via the overall wage payment process. Step B: exert taxation based on the surplus Y as follows: *For each Atom z, calculate the positive part of (IN - OUT), defined as surplus(z). Collect (Y + e1) wealth from all Atom z, collecting from each node a percentage of Y that is proportional to surplus(z), but never so much as to cause IN < OUT for any node (the new STI being collected counts toward OUT).*

In case the total of surplus(z) over all nodes z is less than Y, one may trigger a mid-cycle wage adjustment, calculated so that step B will decrease AVDIFF. I.e. we cut wages since there is not enough surplus to support it.

Finally, in the Variant 2 ECAN cycle, diffusion is done a little differently, via iterating the following process: If AVDIFF has increased during the diffusion round so far, then choose a random node whose diffusion would decrease AVDIFF, and let it diffuse; if AVDIFF has decreased during the diffusion round so far, then choose a random node whose diffusion would increase AVDIFF, and let it diffuse. In carrying out these steps, we avoid letting the same node diffuse twice in the same round. This algorithm does not let all Atoms diffuse in each cycle, but it stochastically lets a lot of diffusion happen in a way that maintains AVDIFF constant. The iteration may be modified to bias toward an average decrease in AVDIFF.

The random element in the diffusion step, together with the logic of the rent/welfare and wage/tax steps, combines to yield the result that for Variant 2 of ECAN dynamics, AVDIFF is a stochastic Lyaponov function. The details of the proof of this will be given elsewhere due to space considerations but the outline of the argument should be clear from the construction of Variant 2. And note that by setting the e and e1 parameter to 0, the convergence requirement can be eliminated, allowing the network to evolve more spontaneously as may be appropriate in some contexts; these parameters allow one to explicitly adjust the convergence rate.

One may also derive results pertaining to the meaningfulness of the attractors, in various special cases. For instance, if we have a memory consisting of a set M of m nodes, and we imprint the memory on the ECAN by stimulating m nodes during an interval of time, then we want to be able to show that the condition where precisely those m nodes are in the AF is a fixed-point attractor. However, this is not difficult, because one must only show that if these m nodes and none others are in the AF, this condition will persist. Rigorous proof of this and related theorems will appear in a follow-up paper.

Associative Memory

We have carried out experiments gauging the performance of Variant 1 of ECAN as an associative memory, using the implementation of ECAN within OpenCog, and using both the conventional and Storkey Hebbian updating formulas. Extensive discussion of these results (along with Variation 2 results) will be deferred to a later publication due to space limitations, but we will make a few relevant comments here.

As with a Hopfield net memory, the memory capacity (defined as the number of memories that can be retrieved from the network with high accuracy) depends on the sparsity of the network, with denser networks leading to greater capacity. In the ECAN case the capacity also depends on a variety of parameters of the ECAN equations, and the precise unraveling of these dependencies is a subject of current research. However, one interesting dependency has already been uncovered in our preliminary experimentation, which has to do with the size of the AF versus the size of the memories being stored.

Define the size of a memory (a pattern being imprinted) as the number of nodes that are stimulated during imprinting of that memory. In a classical Hopfield net experiment, the mean size of a memory is usually around, say, .2-.5 of the number of neurons. In typical OpenCog associative memory situations, we believe the mean size of a memory will be one or two orders of magnitude smaller than that, so that each memory occupies only a relatively small portion of the overall network.

What we have found is that the memory capacity of an ECAN is generally comparable to that of a Hopfield net with the same number of nodes and links, if and only if the ECAN parameters are tuned so that the memories being imprinted can fit into the AF. That is, the AF threshold or (in the hyperbolic case) shape parameter must be tuned so that the size of the memories is not so large that the active nodes in a memory cannot stably fit into the AF. This

tuning may be done adaptively by testing the impact of different threshold/shape values on various memories of the appropriate size; or potentially a theoretical relationship between these quantities could be derived, but this has not been done yet. This is a reasonably satisfying result given the cognitive foundation of ECAN: in loose terms what it means is that ECAN works best for remembering things that fit into its focus of attention during the imprinting process.

Interaction between ECANs and other OpenCog Components

Our analysis above has focused on the associative-memory properties of the networks, however, from the perspective of their utility within OpenCog or other integrative AI systems, this is just one among many critical aspects of ECANs. In this final section we will discuss the broader intended utilization of ECANs in OpenCog in more depth.

First of all, associative-memory functionality is directly important in OpenCogPrime because it is used to drive concept creation. The OCP heuristic called "map formation" creates new Nodes corresponding to prominent attractors in the ECAN, a step that (according to our preliminary results) not only increases the memory capacity of the network beyond what can be achieved with a pure ECAN but also enables attractors to be explicitly manipulated by PLN inference.

Equally important to associative memory is the capability of ECANs to facilitate effective allocation of the attention of other cognitive processes to appropriate knowledge items (Atoms). For example, one key role of ECANs in OCP is to guide the forward and backward chaining processes of PLN (Probabilistic Logic Network) inference. At each step, the PLN inference chainer is faced with a great number of inference steps from which to choose; and a choice is made using a statistical "bandit problem" mechanism that selects each possible inference step with a probability proportional to its expected "desirability." In this context, there is considerable appeal in the heuristic of weighting inference steps using probabilities proportional to the STI values of the Atoms they contain. One thus arrives at a combined PLN/EAN dynamic as follows:

1. An inference step is carried out, involving a choice among multiple possible inference steps, which is made using STI-based weightings (and made among Atoms that LTI weightings have deemed valuable enough to remain in RAM)
2. The Atoms involved in the inference step are rewarded with STI and LTI proportionally to the utility of the inference step (how much it increases the confidence of Atoms in the system's memory)
3. The ECAN operates, and multiple Atom's importance values are updated
4. Return to Step 1 if the inference isn't finished

An analogous interplay may occur between ECANs and the MOSES procedure learning algorithm that also plays a key role in OCP.

It seems intuitively clear that the same attractor-convergence properties highlighted in the present analysis of associative-memory behavior, will also be highly valuable for the application of ECANs to attention allocation. If a collection of Atoms is often collectively useful for some cognitive process (such as PLN), then the associative-memory-type behavior of ECANs means that once a handful of the Atoms in the collection are found useful in a certain inference process, the other Atoms in the collection will get their STI significantly boosted, and will be likely to get chosen in subsequent portions of that same inference process. This is exactly the sort of dynamics one would like to see occur. Systematic experimentation with these interactions between ECAN and other OpenCog processes is one of our research priorities going forwards.

References

Amit, Daniel (1992). *Modeling Brain Function*. Cambridge University Press.

Goertzel, Ben (2006). *The Hidden Pattern*. Brown Walker.

Goertzel, Ben (2007). *Virtual Easter Egg Hunting*. In *Advances in Artificial General Intelligence*, IOS Press.

Goertzel, Ben (2008). *OpenCogPrime: Design for a Thinking Machine*, online at http://www.opencog.org/wiki/OpenCogPrime:WikiBook

Goertzel, Ben, Matthew Ikle', Izabela Goertzel and Ari Heljakka. *Probabilistic Logic Networks*. Springer.

Hutter, Marcus (2004). *Universal AI*. Springer.

Looks, Moshe (2006). *Competent Program Evolution*. PhD thesis in CS department, Washington University at St. Louis.

Storkey A.J. (1997) *Increasing the capacity of the Hopfield network without sacrificing functionality*, ICANN97 p451-456.

Storkey, Amos (1998). *Palimpsest Memories: A New High Capacity Forgetful Learning Rule for Hopfield Networks*.

Storkey A.J. and R. Valabregue (1999) *The basins of attraction of a new Hopfield learning rule*, Neural Networks 12 869-876.