Virtual Easter Egg Hunting: A Thought-Experiment in Embodied Social Learning, Cognitive Process Integration, and the Dynamic Emergence of the Self

Ben GOERTZEL
Novamente LLC

Abstract. The Novamente Cognition Engine (NCE) architecture for Artificial General Intelligence is briefly reviewed, with a focus on exploring how the various cognitive processes involved in the architecture are intended to cooperate in carrying out moderately complex tasks involving controlling an agent embodied in the AGI-Sim 3D simulation world. A handful of previous conference papers have reviewed the overall architecture of the NCE, and discussed some accomplishments of the current, as yet incomplete version of the system; this paper is more speculative and focuses on the intended behaviors of the NCE once the implementation of all its major cognitive processes is complete. The "iterated Easter Egg Hunt" scenario is introduced and used as a running example throughout, due to its combination of perceptual, physical-action, social and self-modeling aspects. To aid in explaining the intended behavior of the NCE, a systematic typology of NCE cognitive processes is introduced. Cognitive processes are typologized as global, operational or focused; and, the focused processes are more specifically categorized as either forward-synthesis or backward-synthesis processes. The typical dynamics of focused cognition is then modeled as an ongoing oscillation between forward and backward synthesis processes, with critical emergent structures such as self and consciousness arising as attractors of this oscillatory dynamic. The emergence of models of self and others from this oscillatory dynamic is reviewed, along with other aspects of cognitive-process integration in the NCE, in the context of the iterated Easter Egg Hunt scenario.

Introduction

The Novamente Cognition Engine (NCE) is an in-development software system, ultimately aimed toward artificial general intelligence at the human level and beyond. It is programmed in C++ and designed for large-scale implementation on a distributed network of Linux machines. The overall architecture of the system is based on principles from cognitive science and complex systems science, and grounded in a systems theory of intelligence; but the specific mechanisms used within the system are drawn from contemporary computer science, utilizing on a number of recent advances in AI theory and practice. Among the key cognitive mechanisms of the Novamente system are a probabilistic reasoning engine based on a novel variant of probabilistic logic called Probabilistic Logic Networks; an evolutionary learning engine that is based on a synthesis of probabilistic modeling and evolutionary programming called MOSES, described in Moshe Looks' 2006 PhD thesis from Washington University (see [1]); and an artificial economics based system for attention allocation and credit assignment. Implementation of the system has been underway for some time but is not yet complete[1]; however, the current incomplete system has nevertheless been used to experiment with various forms of learning and reasoning. Among other applications, the NCE is now being used to control a simulated humanoid agent in 3D simulation world called AGISim (see [2]), similar to a video game world, and in this context is being taught simple behaviors such as playing fetch and tag, finding objects, and recognizing objects by name. These teaching exercises serve to give the system basic world-knowledge which can then be deployed beyond the scope of the simulation world; and they serve as the first stage of a principled process of "AGI developmental psychology" based loosely on the ideas of Jean Piaget ([3]).

The Novamente design has been overviewed in a number of conference papers before[2] ([4,5,6,7]) and our goal here is not to repeat this content, but rather to give a more elegantly conceptually organized perspective on the cognitive processes contained in the system, and to explore the means by which it is hypothesized these processes will be able to act together, in a coordinated way, to enable embodied social learning and to spawn the emergence within a Novamente system's dynamic knowledge base of a structure fairly describable as a "self" (or, to use a more technical term introduced by [8]), a "phenomenal self"). To get the most out of this paper, the reader should first read these prior overviews; but nonetheless the current paper has been written to be minimally self-contained.

Two recent papers ([2,9]) discuss learning and reasoning experiments that have been conducted with the current system; and, two other papers in this volume discuss aspects of the Novamente system as well. Looks' paper discusses the MOSES probabilistic evolutionary learning system created for incorporation into Novamente; and the Ikle' et al paper discusses some of the mathematics underlying Novamente's Probabilistic Logic Networks (PLN), including a simple example of PLN inference applied to embodied learning in the AGISim simulation world. This paper is more speculative, and describes both implemented/tested and as-yet unimplemented/untested aspects of the NCE design, with a focus on some of the integrative cognitive dynamics and emergent structures that are expected to be observed once the implementation, testing and tuning of the system is more complete.

Section 2 gives a brief overview of the Novamente architecture, largely covering ground already reviewed in prior publications. Section 3 presents a typology of cognitive processes using the notions of forward and backward synthesis, and utilizes this to give a unified presentation of the various cognitive processes existing in the NCE, and to compose an hypothesis regarding the origins of the phenomenal self and the attentional focus from complex cognitive dynamics. Section 4 describes the iterated Easter Egg Hunt scenario, and discusses how the NCE's cognitive dynamics are expected to productively cooperate in this context, in a completely implemented NCE.

The system-theoretic focus makes this paper different than the typical paper in the AI literature; and this difference reflects the relatively unusual nature of the system-theoretic methodology underlying the NCE project. In our view, the only kind of approach to AGI likely to meet with success is one in which the systematic coordinated behavior of a holistic AGI system is carefully envisioned prior to the detailed implementation and testing of the system. I.e., I suggest that the parts of an AGI system

[1] The implementation process is time-consuming not only due to the usual difficulties of large-scale software engineering, but also due to the need to carefully define and test a large number of details not fully specified in the overall AI design.

[2]

must be designed, in detail, with a specific view toward causing the whole to behave appropriately. Consistently with this, I assign low success odds to “integrative” designs in which multiple “best of breed” narrow-AI or proto-AGI components created independently by multiple research teams are hooked together within a a flexible overall design. And, I assign a fundamental necessity to in-depth speculative exercises such as the one reported in Section 4 here -- if AGI is to be achieved via computer science methods (rather than, for example, by detailed emulation of the human brain).

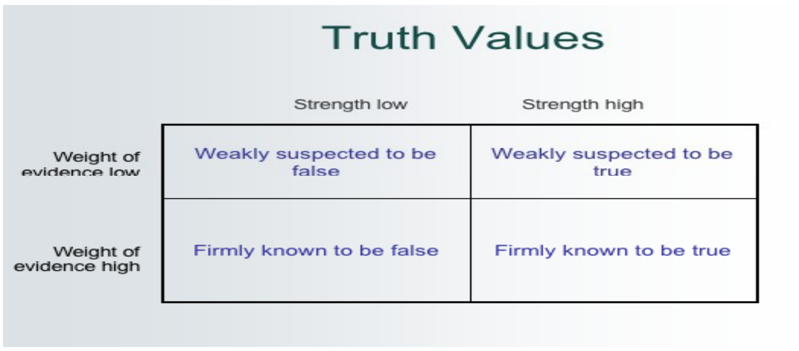
Above it was stated that the ultimate objective of NCE development is to create artificial intelligence at the human level and beyond. More specifically, the long-term objective of NCE development is to create a powerful “artificial scientist” that can ingest a vast variety of quantitative, relational and textual data from the Internet and then interact with human scientists to produce innovative scientific creations and discoveries. It may seem that Easter Egg hunting – which in its more sophisticated social-learning aspects is still beyond the current Novamente implementation – is an extremely long way from this sort of ambitious goal. But according to the developmental-psychology paradigm outlined in ([9]), there is a very natural path from these early-childhood-type learning tasks to the more complex and formal learning tasks required for doing science. The hypothesis motivating our current work on early-childhood-level learning tasks is that doing science does not require any cognitive mechanisms, knowledge representations or architecture components beyond those required for doing Easter Egg hunting (or other similar tasks) in a robust, adaptive, socially and reflectively aware way.

1. Brief Overview of the Novamente Architecture

One may describe the Novamente AGI architecture in terms of four different aspects: knoweldge representation, cognitive architecture, cognitive processes, and emergent structures. This section deals mainly with the former two; and the next section deals mainly with the latter two.

1.1.NCE Knowledge Representation

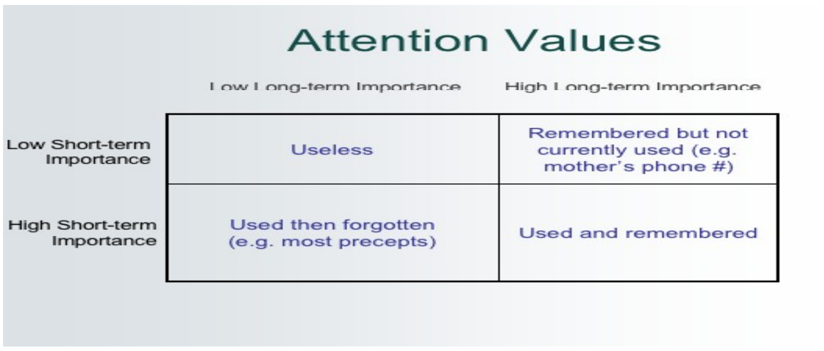
The NCE utilizes a knowledge representation in which declarative knowledge is represented using weighted, labeled hypergraphs; procedural knowledge is represented using programs in a customized functional programming language called Combo; and mechanisms are in place for freely converting between declarative and procedural knowledge. Nodes and links in the declarative-knowledge hypergraph are grouped together into the category of "Atoms." Atoms are quantified with truth values (see Figure 1) that, in their simplest form, have two components, one representing probability ("strength") and the other representing "weight of evidence"; and also with "attention values" (see Figure 2) that have two components, short-term and long-term importance, representing the estimated value of the Atom on immediate and long-term time-scales.



Truth Values

	Strength low	Strength high
Weight of evidence low	Weakly suspected to be false	Weakly suspected to be true
Weight of evidence high	Firmly known to be false	Firmly known to be true

Figure 1. A coarse-grained view of the semantics of the truth values attached to Novamente Atoms.



Attention Values

	Low Long-term Importance	High Long-term Importance
Low Short-term Importance	Useless	Remembered but not currently used (e.g. mother's phone #)
High Short-term Importance	Used then forgotten (e.g. most precepts)	Used and remembered

Figure 2. A coarse-grained view of the semantics of the attention values attached to Novamente Atoms.

The vocabulary of node and link types used to represent knowledge in Novamente has been presented in ([4]). Figures 3 and 4 give some simple examples. An incomplete list of node types is as follows:

- ConceptNodes
    - “tokens” for links to attach to
- PredicateNodes
- ProcedureNodes
- PerceptNodes
    - Visual, acoustic percepts, etc.
- NumberNodes

And an incomplete list of link types is:

- Logical links
    - InheritanceLink
    - SimilarityLink
    - ImplicationLink
    - EquivalenceLink
    - Intensional logical relationships
- HebbianLinks
- Procedure evaluation links

The semantics of these types is discussed in prior publications.

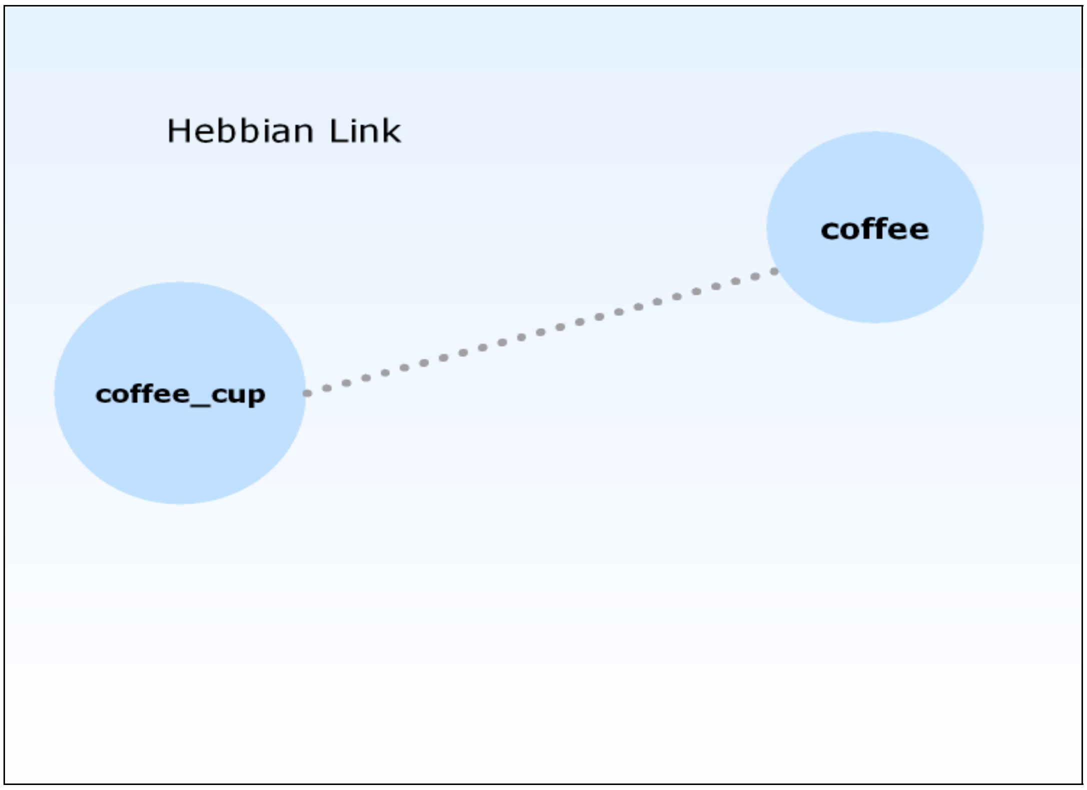


Figure 3. HebbianLinks may denote generic association, calculated based on the degree to which two Atoms have proved simultaneously useful to the system.

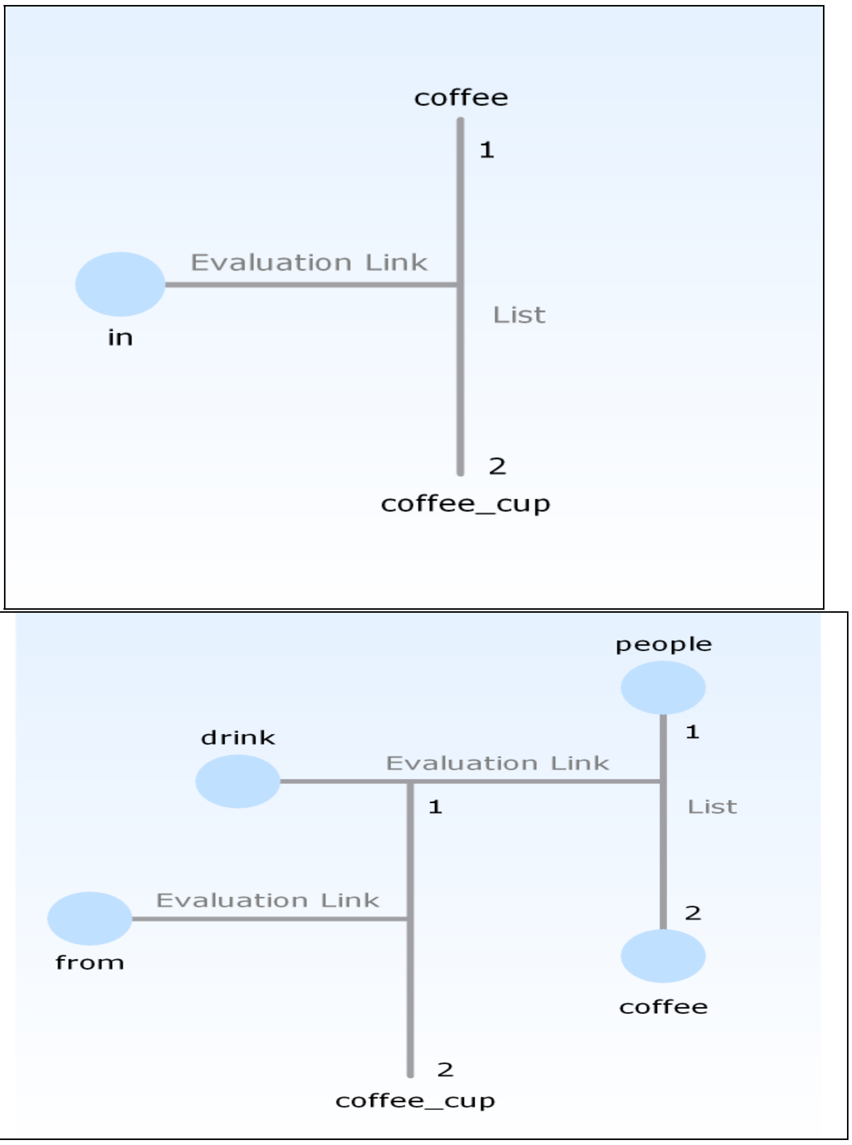
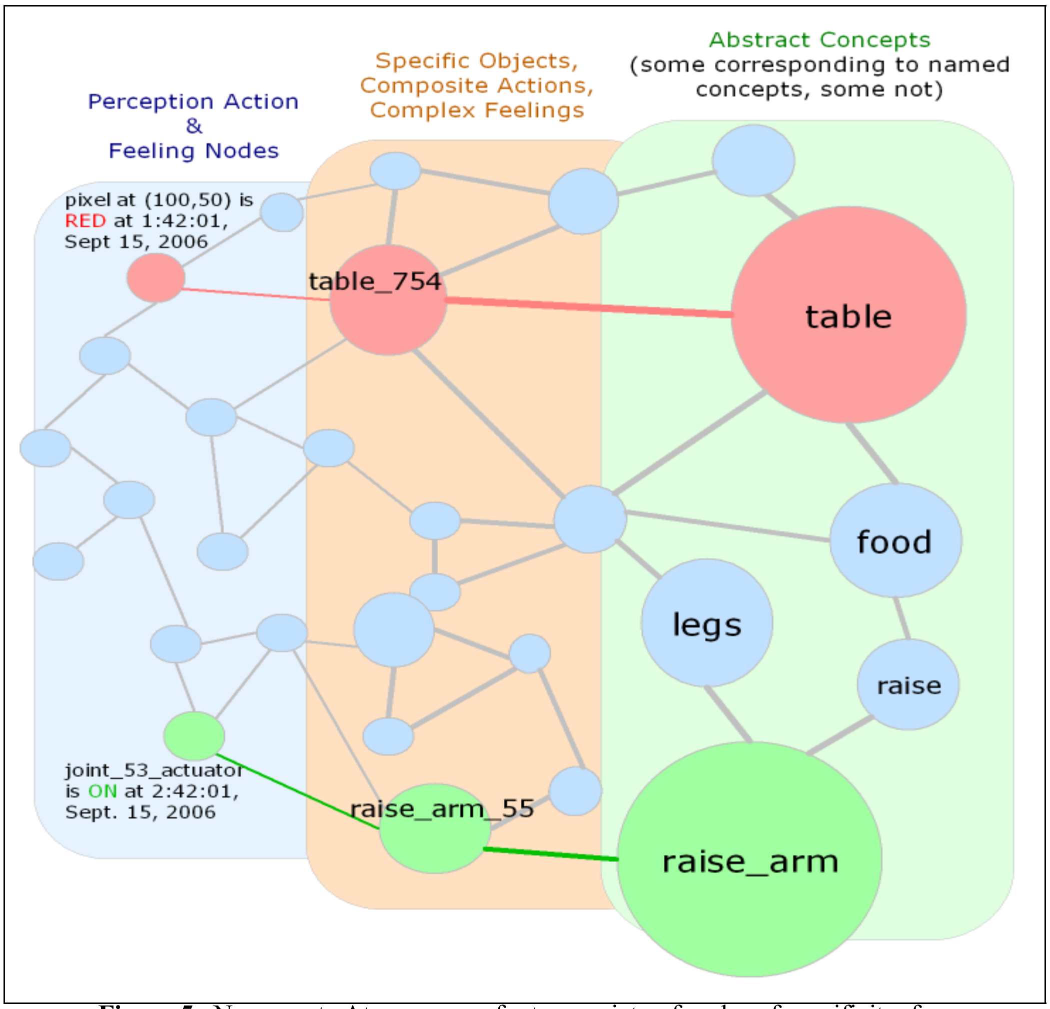


Figure 4. Links may also denote precise logical relationships, quantified with degrees of uncertainty and importance (quantifications not shown in the diagram).



Perception Action & Feeling Nodes
Specific Objects, Composite Actions, Complex Feelings
Abstract Concepts (some corresponding to named concepts, some not)
pixel at (100,50) is RED at 1:42:01, Sept 15, 2006
table_754
table
food
legs
raise
joint_53_actuator is ON at 2:42:01, Sept. 15, 2006
raise_arm_55
raise_arm

Figure 5. Novamente Atoms may refer to a variety of scales of specificity, from specific percepts or action-commands, to specific entities (like one particular table or one particular action of raising the arm), to general concepts. Some nodes in the diagram are labeled and some are not, reflecting the fact that in the actual Novamente Atomspace many nodes do not correspond to any particular English word nor any concept or entity compactly describable in English.

It is important to note that the Novamente declarative knowledge representation is neither a neural net nor a semantic net, though it does have some commonalities with each of these traditional representations. It is not a neural net because it has no activation values, and involves no attempts at low-level brain modeling. However, "attention values" are very loosely analogous to time-averages of neural net activations.On the other hand, it is not a semantic net because of the broad scope of the Atoms in the network (see Figure 5): for example, Atoms may represent percepts, procedures, or parts of concepts. Most Novamente Atoms have no corresponding English label. However, most Novamente Atoms do have probabilistic truth values, allowing logical semantics.

1.2.NCE Cognitive and Software Architecture

The overall NCE software architecture consists of a collection of specialized units, each of which uses the same knowledge representations and cognitive mechanisms to achieve a particular aspect of intelligence, such as perception, language learning, abstract cognition, action selection, procedure learning, etc. The breakdown into units, indicated in Figure 8, is based loosely on ideas from cognitive science, and is fairly similar to that presented in other integrative AGI architectures proposed by [11, 12] and others. However, the specifics of the breakdown have been chosen with care, with a view toward ensuring that the coordinated dynamics of the mechanisms and units will be able to give rise to the emergent structures and dynamics associated with intelligence, including a sophisticated self-model and an appropriately adaptive "moving focus of attention."

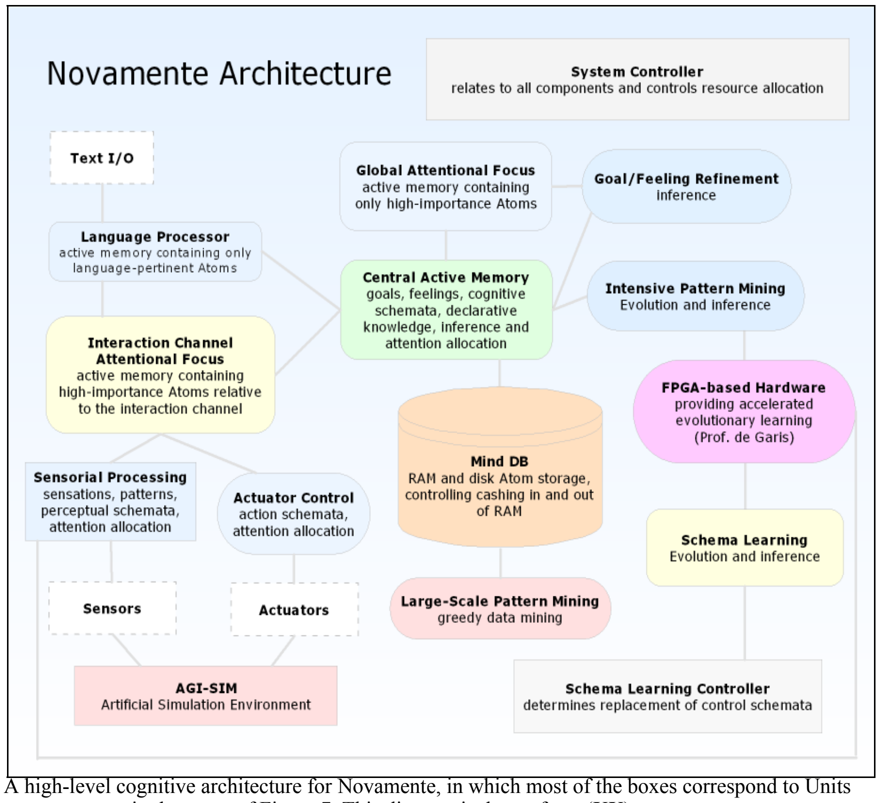

Novamente Architecture
System Controller
relates to all components and controls resource allocation
Text I/O
Global Attentional Focus
active memory containing only high-importance Atoms
Goal/Feeling Refinement
inference
Language Processor
active memory containing only language-pertinent Atoms
Central Active Memory
goals, feelings, cognitive schemata, declarative knowledge, inference and attention allocation
Intensive Pattern Mining
Evolution and inference
Interaction Channel Attentional Focus
active memory containing high-importance Atoms relative to the interaction channel
FPGA-based Hardware
providing accelerated evolutionary learning (Prof. de Garis)
Mind DB
RAM and disk Atom storage, controlling cashing in and out of RAM
Sensorial Processing
sensations, patterns, perceptual schemata, attention allocation
Actuator Control
action schemata, attention allocation
Schema Learning
Evolution and inference
Sensors
Actuators
Large-Scale Pattern Mining
greedy data mining
AGI-SIM
Artificial Simulation Environment
Schema Learning Controller
determines replacement of control schemata

Figure 6. A high-level cognitive architecture for Novamente, in which most of the boxes correspond to Units in the sense of Figure 7. This diagram is drawn from (XX).

As shown in Figure 7, each individual unit within the NCE architecture contains a table of Atoms and Combo trees representing knowledge, and then a collection of processes called MindAgents that collaboratively act on the AtomTable. The Mind Agents embody various cognitive processes, as will be described below. A unit may span several machines or may be localized on a single machine: in the multi-machine case, Nodes on one machine may link to Nodes living in other machines within the same unit. On the other hand, Atoms in one Unit may not directly link to Atoms in another Unit; though different Units may of course transport Atoms amongst each other. This architecture is workable to the extent that Units may be defined corresponding to pragmatically distinct areas of mental function, e.g. a Unit for language processing, a Unit for visual perception, etc.

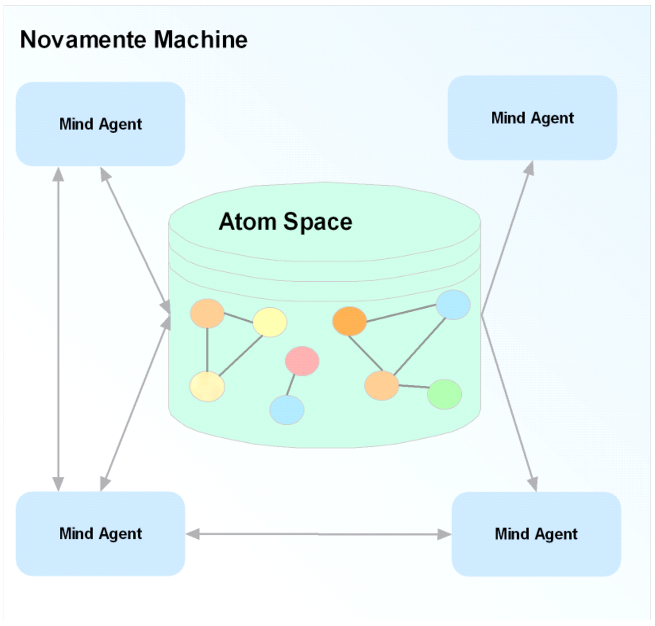


Figure 7. The architecture of a single Novamente "lobe", involving a table of Atoms and then a collection of processes called MindAgents that collaboratively act on the AtomTable. A Lobe may run on one machine or several. A Scheduler object regulates the activity of the MindAgents; and the MindAgents may spawn objects called Tasks that carry out one-time cognitive actions, and are scheduled using a ticketing system. Some MindAgents carry out actions not centered on the AtomTable, such as the SchemaExecution MindAgent which executes procedures in the manner of a programming language interpreter.

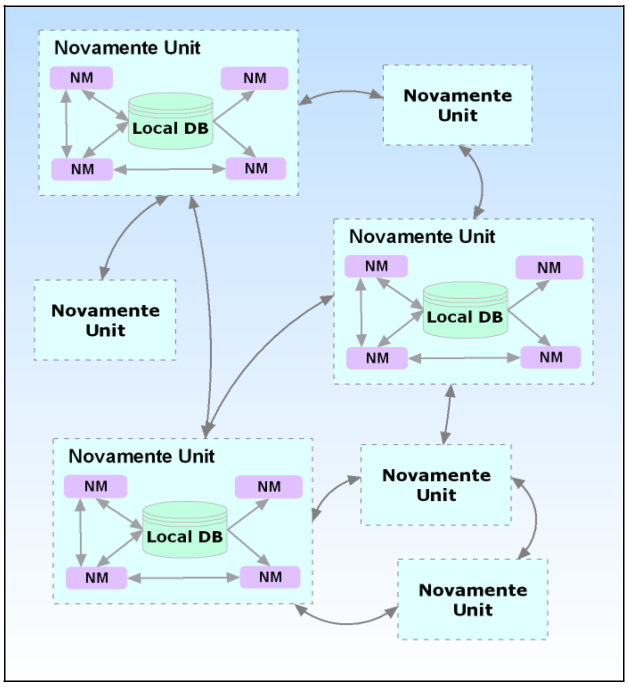


Figure 8. The high-level architecture of Novamente consists of a set of Units, each of which consists of a set of machines embodying the basic one-machine Novamente architecture. All the machines in a Unit share a common Atomspace, i.e. links in the AtomTable of one machine in a Unit may point to Atoms in another machine in the Unit. On the other hand, different Units have separate Atomspaces and Atoms in one Unit may not directly link to Atoms in another Unit; though different Units may of course transport Atoms amongst each other. This architecture is workable to the extent that Units may be defined corresponding to pragmatically distinct areas of mental function, e.g. a Unit for language processing, a Unit for visual perception, etc.

Next, Figure 9 gives a simplified, schematic view of the flow of information through the Novamente system, in the context of its use to control an agent in the AGISim simulation world. This diagram does not cover “background reasoning and learning” processes that serve to create knowledge useful in figuring out how to achieve goals.

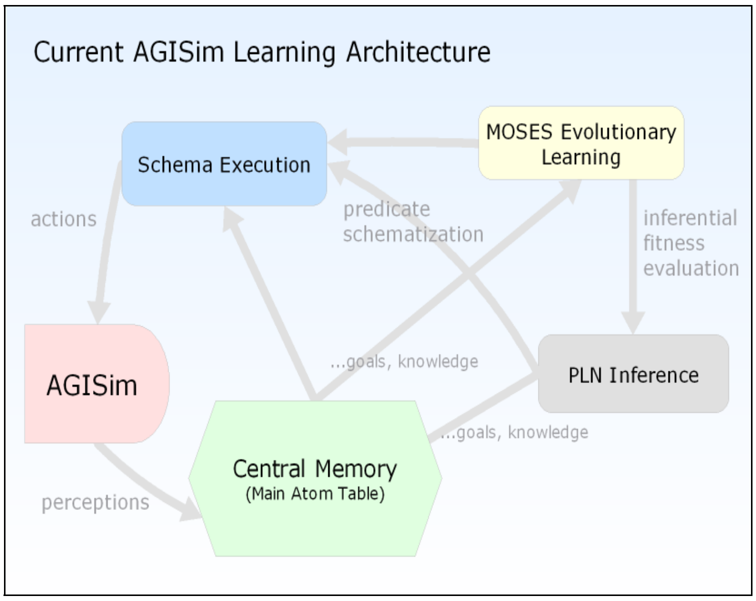


Figure 9. Schematic diagram of the architecture of the current NAIE implementation, which only implements part of the overall design, but successfully carried out various simple learning tasks in the AGISim environment.

The full cognitive architecture has not yet been implemented; the currently implemented subset of the architecture will be briefly reviewed at the start of Section 4.

2. A Typology of Cognitive Processes in Novamente

The most complex aspect of the NCE design is the set of cognitive processes involved. This set of processes may be presented in a number of different ways. One approach is to look at the set of fundamental "AI learning" algorithms that are involved. In this sense, there are four main algorithmic approaches (each one of which involves a combination of a number of specific algorithms):

- Probabilistic Term Logic, a novel approach to inference under uncertainty, which is partially described in ([13]) in this volume.
- Probabilistic Evolutionary Learning, which in the current Novamente version takes the form of the MOSES algorithm, discussed in ([14]) in this volume.
- Stochastic pattern mining, which finds combinations of Atoms that satisfy a specified criterion (such as frequency of occurrence, statistical surprisingness, etc.)
- Artificial economics, used for attention allocation and credit assignment, leading to novel forms of adaptively evolving distributed knowledge

representation.

In previous review papers, the set of cognitive processes involved in the Novamente architecture has been presented in a somewhat unorganized manner, as a simple list of MindAgents. Such a presentation is not inaccurate, but gives the impression of even more conceptual complication than actually exists in the design. In this section I will present a typology of Novamente cognitive processes, which makes clear that the diverse population of MindAgents in Novamente naturally subdivides into a handful of high-level conceptual categories. Furthermore (though this has not been validated yet), I conjecture that this typologizing exercise may have conceptual value beyond the scope of Novamente, by imposing a natural ontology on the diversity of cognitive processes necessarily present in any complex, multifaceted AI system.

First of all, at the highest level, I divide the cognitive processes occurring in Novamente into three categories:

- Global processes
    - MindAgents that periodically iterate through all Atoms and act on them
    - "Things that all Atoms do"
- Control Processes
    - Processes directly involved with carrying out specifically orchestrated sequences of actions in the external world or in the system's own infrastructure
- Focused processes
    - MindAgents that begin by selecting a small set of important or relevant Atoms, and then act on these to generate a few more small sets of Atoms, and iterate.

I suggest that processes in all these categories are critical to Novamente's intelligence (and more hypothetically, to intelligence in general). However, we also suggest that the greatest subtlety lies in the focused cognitive processes. The global and control processes are extremely necessary, however, my conjecture is that these processes don't necessarily need to be a lot more sophisticated in a human-level intelligence than in a significantly sub-human-level intelligence. The distinguishing characteristic of human-level intelligence, I suggest, is a much more highly-developed set of focused cognitive processes. To better understand focused cognitive processes in a general way, I will utilize the ideas of forward and backward synthesis developed in ([15]).

Key examples of Novamente control processes are as follows:

- Schema Execution
    - This involves the "programming language interpreter" used to actually execute schemata created from NM Atoms
- Maintenance of "active schema pool" (SchemaSelection MindAgent)
    - This involves choosing which procedures to place in the pool of "currently executing schemata" (meaning, schemata that are currently ready to be activated if their input conditions are met)
- Maintenance of "active goal pool" (FeasibilityUpdating MindAgent)
    - Determination of the set of predicates that are currently actively considered as system goals, which is done by updating "feasibility" information regarding the achievability of various goals given various committments of resources

These are processes that are necessary in order for the system to carry out actions; including the actual execution of actions, the selection of subgoals to be used to determine which actions to carry out, the choice of which actions to carry out, and various specialized actions such as the updating of system control schemata based on meta-learning. In a sense these are "focused cognitive processes" but they have a different nature than the processes to be analyzed here under that label: they are not concerned with taking a set of knowledge items and focused-ly learning more about it, but are, rather, concerned with actually doing things either in the external world or in the system's own infrastructure.

Next, key examples of global processes are

- Attention Allocation
    - Updates short and long term importance values associated with Atoms
    - Uses a "simulated economy" approach, with separate currencies for short and long term importance
- Stochastic pattern mining of the AtomTable
    - A powerful technique for predicate formation
  - Critical for perceptual pattern recognition as well as cognition
    - Pattern mining of inference histories critical to advanced inference control
- Building of the SystemActivityTable
    - Records which MindAgents acted on which Atoms at which times
    - Table is used for building HebbianLinks, which are used in attention allocation

Finally, focused cognitive processes may be decomposed into forward versus backward synthesis processes. The distinction between forward and backward synthesis processes is drawn in detail in ([15]), and depicted in Figures 10 and 11. Roughly, forward synthesis begins with a small set of entities, and then combines them (with each other and with other entities) to form new entities, and iterates this process. On the other hand, backward synthesis begins with a small set of entities, and tries to figure out how to generate this set via forward synthesis starting from some other set of entities that meets certain criteria (such as, in the case of backward inference, being confidently known).

Novamente's key forward synthesis processes are as follows:

- Forward-Chaining Probabilistic Inference (see Figure 12)
    - Given a set of knowledge items, figure out what (definitely or speculatively) follows from it according to the rules of probabilistic term logic
- Concept/Goal Formation (see Figure 13)
    - "Blend" existing concepts or goals to form new ones
    - [16] have explored the cognitive significance of this sort of blending operation in great detail.
- Map formation (see Figure 14)
    - Create new Atoms out of sets of Atoms that tend to be simultaneously important (or whose importance tends to be coordinated according to some other temporal pattern)
- Language Generation
    - A subcategory of forward-chaining inference, but important enough to be isolated and considered on its own
    - Atoms representing semantic relationships are combined with Atoms representing linguistic mapping rules to produce Atoms representing syntactic relationships, which are then transformed into sentences
- Importance Propagation
    - Atoms pass some of their "attentional currency" to Atoms that they estimate may help them become important again in the future

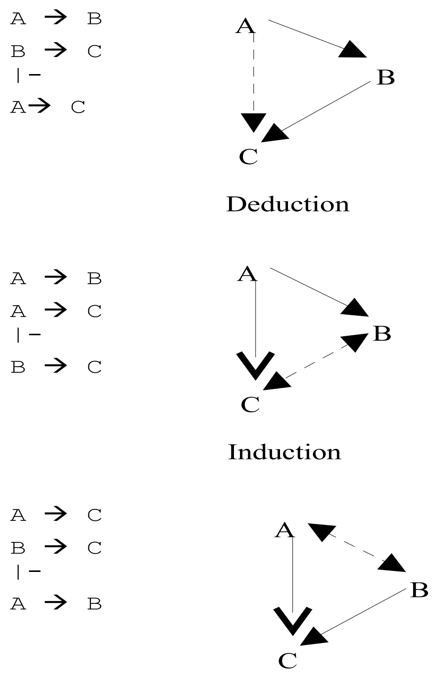


Figure 12. Probabilistic Logic Networks for uncertain inference: an illustration of first-order deduction, induction and abduction acting on extensional InheritanceLinks. This is a forward-synthesis process which chooses pairs of links and combines them to form new links, with truth values determined by probabilistic inference rules.

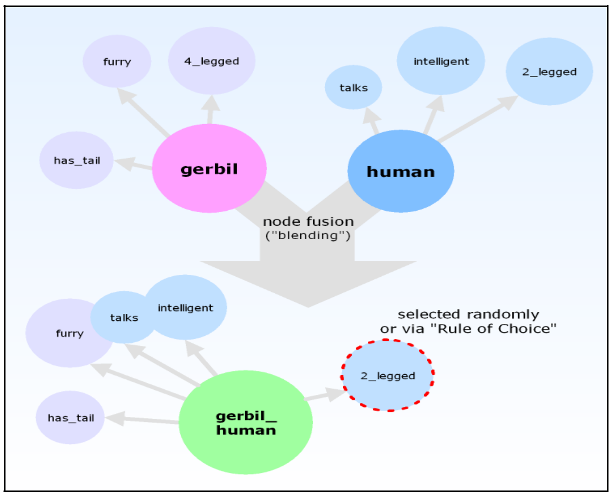


Figure 13. One of Novamente's heuristics for new-concept creation is "blending," in which some links from one concept are combined with some links from another concept.

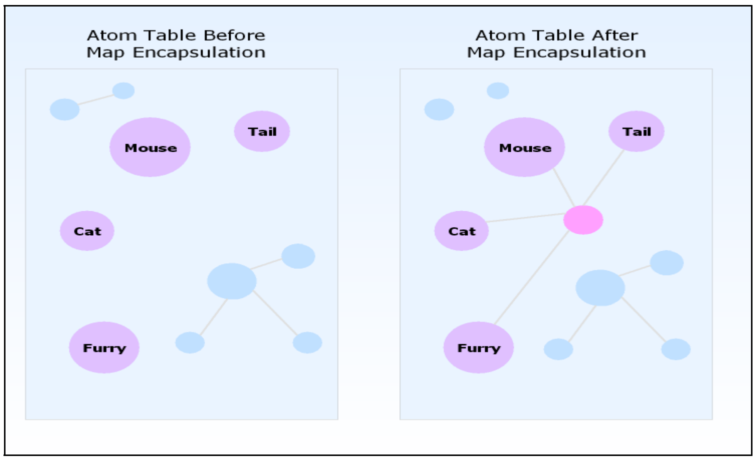

Atom Table Before Map Encapsulation
Mouse
Tail
Cat
Furry
Atom Table After Map Encapsulation
Mouse
Tail
Cat
Furry

Figure 14. Atoms commonly used together may be grouped together via linking them all to a newly created Atom. This process is called "map formation" and is one way that the Novamente system can effectively recognize patterns in itself.

Next, Novamente's key backward synthesis processeses are:

- Backward-chaining probabilistic inference
  - Given a target Atom, find ways to produce and evaluate it logically from other knowledge
- Inference process adaptation
  - Given a set of inferential conclusions, find ways to produce those conclusions more effectively than was done before
- Predicate Schematization (Figure 15)
  - Given a goal, and knowledge about how to achieve the goal, synthesize a procedure for achieving the goal
- Credit Assignment
  - Given a goal, figure out which procedures' execution, and which Atoms' importance, can be expected to lead to the goal's achievement
- Goal Refinement
  - Given a goal, find other (sub)goals that imply that goal
- Model-Based Predicate Generation
  - Given probabilistic knowledge about what patterns characterize predicates or procedures satisfying a certain criterion, generate new predicate/procedures satisfying the criterion
- Criterion-Based Predicate Modeling
  - Building of probabilistic knowledge regarding the patterns characterizing predicates satisfying a certain criterion
- Language Comprehension
  - Syntax parsing: given a sentence, or other utterance, search for assignments of syntactic relationships to words that will make

the sentence grammatical

- Semantic mapping: Search for assignment of semantic meanings to words and syntactic relationships that will make the sentence contextually meaningful

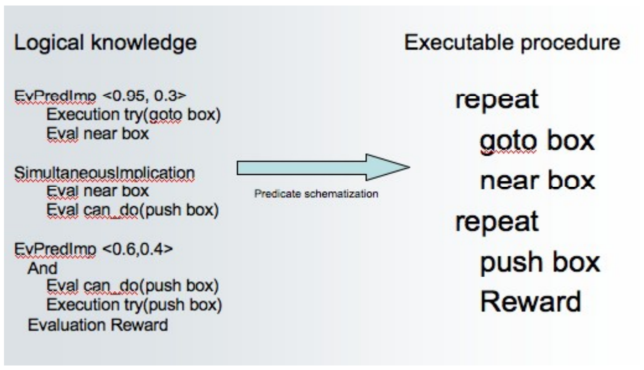


Figure 15. Predicate schematization: the backward synthesis process that maps declarative knowledge about how to achieve goals, into procedural knowledge that may be executed to actually achieve goals.

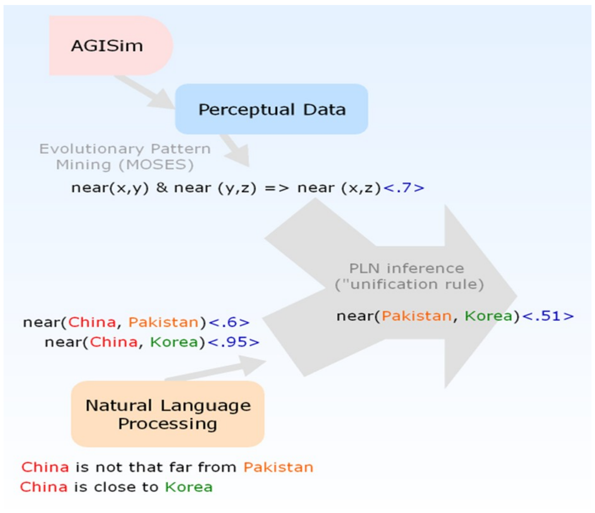

AGISim
Perceptual Data
Evolutionary Pattern Mining (MOSES)
near(x,y) & near (y,z) => near (x,z)<.7>
PLN inference ("unification rule)
near(Pakistan, Korea)<.51>
near(China, Pakistan)<.6>
near(China, Korea)<.95>
Natural Language Processing
China is not that far from Pakistan
China is close to Korea

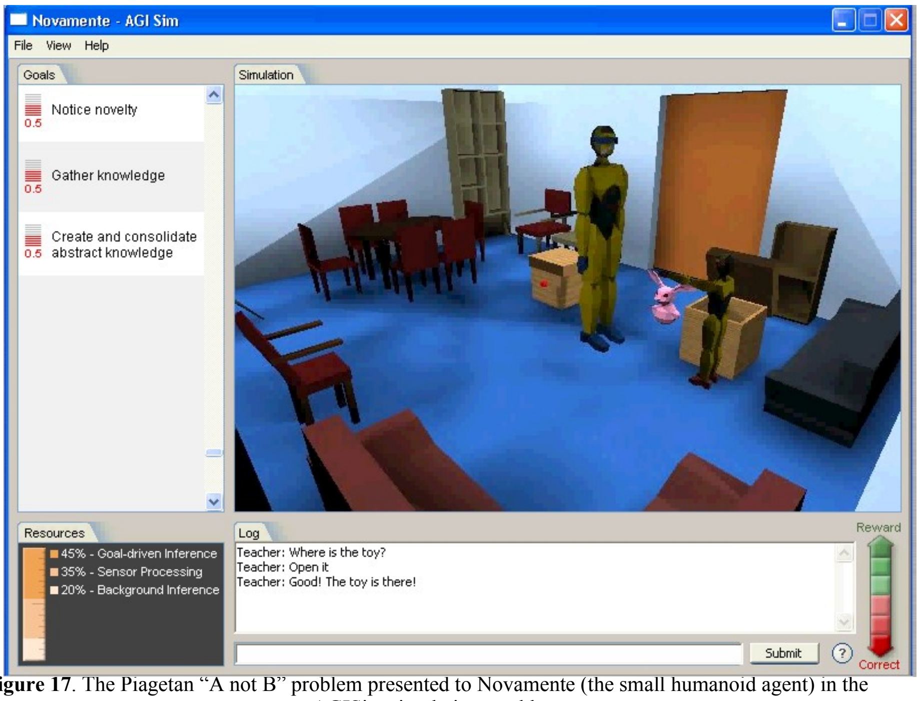
Figure 16. A simple illustration of symbol grounding achieved through integrative intelligence. Evolutionary pattern mining discovers the grounding of the word "near" in perceived (AGISim) relationships, and then discovers the transitivity of nearness through analysis of multiple examples of perceived nearness. Language processing understands that China and Korea are near each other, and that China and Pakistan are somewhat near each other. Inference may then combine these facts learned by language processing with the transitivity of nearness learned via evolutionary learning on percepts gained via embodied experience, and conclude that Pakistan is somewhat near Korea. While simple, this is a nontrivial example of grounding in the sense that involves the grounding of an abstract relationship (transitivity of nearness) in perceived reality and then transferral of this relationship to nonperceived reality.

As a simple example of forward and backward synthesis in action, observe the example of embodied learning of commonsense knowledge shown in Figure 16, reflecting actual experiments conducted with the current version of the NCE. In these experiments, MOSES was used to ground the concept of "nearness" in a simulation world. MOSES was able to learn the rule that nearness is transitive – via "backward synthesis," i.e. the search for rules simplifying and explaining the available data. Then, through use of the RelEx English language parsing front end, simple natural language sentences regarding the nearness of different countries to each other were entered in, and translated into semantic nodes and links in the NCE. (This is not the way a mature NCE AGI would process language, but it is a useful approach for preliminary experimentation while the NCE is not yet at the stage where it can learn English language on its own via its embodied social experience.) The PLN inference system can then apply the learned transitivity of nearness to the knowledge gained via language processing – a simple example of forward-chaining inference, resulting in the conclusion that (in the example in the Figure) Korea is slightly near Pakistan. The general process exemplified in this Figure is an important one. Embodied experience is learned to gain commonsense knowledge, which is expressed in abstract form, and can then be inferentially applied to domains remote from the one in which the knowledge was originally gained.

3. Embodied Social Cognition in the AGISim Environment

Now we briefly move from cognitive processes to another critical topic: methodologies of instruction. There is a variety of methods by which an AGI system may viably gain knowledge, including but not limited to:

- physically embodied experience, via robotic embodiment
- virtually embodied experience, via embodiment within a simulation world
    - example: the AGISim world, shown in Figure 17
- non-embodied experience
    - e.g. via interaction with various online software agents
- conversation with humans
- reading in structured data from databases
    - databases of general knowledge constructed for other purposes
    - relational databases
    - the Mizar mathematics database
    - quantitative scientific, financial, etc. data
    - knowledge DB's constructed specifically for AI's and other software programs
        - everyday knowledge oriented DB's, e.g. Cyc [17]
        - linguistics oriented DB's, e.g WordNet [18], FrameNet [19]
- reading knowledge encoded in language
    - natural language texts, e.g. online texts or textbooks
    - texts written for AI's in constructed languages like Lojban or Lojban++ [20]



Teacher: Where is the toy?
Teacher: Open it
Teacher: Good! The toy is there!

Figure 17. The Piagetan "A not B" problem presented to Novamente (the small humanoid agent) in the AGISim simulation world.

One of the methodological principles underlying the Novamente approach to AGI education is that there is no need to choose between these. Given the right AI design you can have your cake and eat it too. Initially, the NCE is being instructed via a combination of database-ingestion and virtually embodied experience. In the next phase, we plan to augment this with a combination of instruction in English and in Lojban++. This flexibility is enabled because the NCE's knowledge representation permits explicit representation of knowledge (e.g. the creation of nodes corresponding to English-language concepts), but also permits experiential learning and implicit representation of knowledge in terms of learned patterns. The intention is that once the NCE becomes clever enough it will learn mainly via reading knowledge encoded in language, and conversational interaction with humans. Embodiment in physical robots may also be interesting but is not viewed as critically necessary.

Our initial goal in teaching the NCE in the AGISim world is simply to make the system able to learn infant-level behaviors "without cheating" -- i.e. with the only instruction being interactions with a human-controlled agent in the simulation world. Example behaviors desired here are: naming objects, asking for objects, fetching objects, finding hidden objects, playing tag. The system will be tested using a set of tasks derived from human developmental psychology, a process that is already ongoing with tasks such as word-object association and Piaget's A-not-B task ([3], [21]); see Figure 17 for a depiction of this problem as presented to a Novamente-controlled agent in the AGISim simulation world).

The next step beyond this has to do with language understanding. Via instructing the system in language simultaneously with interacting with it in the simulation world, we believe that the system will be taught language in a way that it really understands it pragmatically and personally, unlike the kind of quasi-understanding possessed by current statistical or rule-based natural language systems. Once it is partway through this stage, it will possess the ability to learn from human teachers via linguistic communication utilizing complex recursive phrase structure grammar and grounded semantics. After this point is reached, we anticipate that future progress will accelerate considerably.[3]

Tied up with language understanding, of course, are social interaction and self-understanding, as argued by [22]. If all goes as envisioned, then as NCE improves its communicative ability, it will also improve its self-understanding. This process may be understood in terms of the concepts of forward and backward synthesis introduced above, as will be discussed a little later.

3.1.The Currently Implemented NCE/AGISim Architecture

Currently, as we work toward making a more and more completely functional and robust "artificial baby," we are working with a partial version of the NCE as depicted in Figure 9 above, incorporating the following components:

- **Novamente core system**
  - AtomTable, MindAgents, Scheduler, etc.
  - Now runs on one machine; designed for distributed processing
- **PLN**
  - Relatively crude inference control heuristics
  - Simplistic predicate schematization
  -
  -
- **MOSES**
  - Little experimentation has been done evolving procedures with complex control structures
  - Not yet fully integrated with PLN
- **Schema execution framework**
  - Enacts learned procedures
- **AGISim**
  - And proxy for communication with NM core
- **Perception**
  - Stochastic conjunctive pattern mining for finding repeated patterns in data coming in from AGISim
- **NLP front end**

[3] Also, at this stage, the symbol groundings learned by the system will be valuable for various narrow-AI purposes, such as natural language question answering.

- o External NLP system for "cheating" style knowledge ingestion in the form of logical predicates, without the AGI system itself understanding the syntactic rules

Using this restricted system, we are working with simple tasks such as fetch, tag, word-object association and the Piagetan A-not-B experiment. The current version has proved capable of carrying out these tasks in various cases, and we conjecture that it will be capable of robustly carrying out a variety of similar tasks. However, to move beyond the infantile level we will clearly need to implement more of the NCE architecture, including most critically the economic attention allocation component; and we will need to integrate the MOSES and PLN components more fully than has been done so far.

3.2.The Emergence of the Self via Embodied Social Learning

As noted above, one of the key things we hope to see via teaching the NCE in the AGISim environment is the adaptive emergence within the NCE's knowledge base of an active and effectively evolving "phenomenal self." The process of the emergence of the self may, we hypothesize, be productively modeled in terms of the processes of forward and backward synthesis discussed above. This point is made carefully in [14] and just briefly summarized here.

What is ventured there is that the dynamic pattern of alternating forward and backward synthesis may play a fundamental role in cognition. Put simply, forward synthesis creates new mental forms by combining existing ones. Then, backward synthesis seeks simple explanations for the forms in the mind, including the newly created ones; and, this explanation itself then comprises additional new forms in the mind, to be used as fodder for the next round of forward synthesis. Or, to put it yet more simply:

… Combine … Explain … Combine … Explain … Combine …

This sort of dynamic may be expressed formally, in a Novamente context, as a dynamical iteration on the space of Atoms. One may then speak about attractors of this iteration: fixed points, limit cycles and strange attractors. And one may hypothesize some key emergent cognitive structures are strange attractors of this equation. I.e., the iterative dynamic of combination and explanation leads to the emergence of certain complex structures that are, in essence, maintained when one recombines their parts and then seeks to explain the recombinations. These structures are built in the first place through iterative recombination and explanation, and then survive in the mind because they are conserved by this process. They then ongoingly guide the construction and destruction of various other temporary mental structures that are not so conserved. Specifically, we suggest that both self and attentional focus may be viewed as strange attractors of this iteration. Here we will focus only on self.

The "self" in the present context refers to the "phenomenal self" [8] or "self-model." That is, the self is the model that a system builds internally, reflecting the patterns observed in the (external and internal) world that directly pertain to the system itself. As is well known in everyday human life, self-models need not be completely accurate to be useful; and in the presence of certain psychological factors, a more accurate self-model may not necessarily be advantageous. But a self-model that is too badly inaccurate will lead to a badly-functioning system that is unable to effectively act toward the achievement of its own goals.

The value of a self-model for any intelligent system carrying out embodied agentive cognition is obvious. And beyond this, another primary use of the self is as a foundation for metaphors and analogies in various domains. A self-model can in many cases form a self-fulfilling prophecy (to make an obvious double-entendre'!). Actions are generated based on one's model of what sorts of actions one can and/or should take; and the results of these actions are then incorporated into one's self-model. If a self-model proves a generally bad guide to action selection, this may never be discovered, unless said self-model includes the knowledge that semi-random experimentation is often useful.

In what sense, then, may it be said that self is an attractor of iterated forward-backward synthesis? Backward synthesis infers the self from observations of system behavior. The system asks: What kind of system might we be, in order to give rise to these behaviors that we observe myself carrying out? Based on asking itself this question, it constructs a model of itself, i.e. it constructs a self. Then, this self guides the system's behavior: it builds new logical relationships its self-model and various other entities, in order to guide its future actions oriented toward achieving its goals. Based on the behaviors new induced via this constructive, forward-synthesis activity, the system may then engage in backward synthesis again and ask: What must we be now, in order to have carried out these new actions? And so on.

Our hypothesis is that after repeated iterations of this sort, in infancy, finally during early childhood a kind of self-reinforcing attractor occurs, and we have a self-model that is resilient and doesn't change dramatically when new instances of action- or explanation-generation occur. This is not strictly a mathematical attractor, though, because over a long period of time the self may well shift significantly. But, for a mature self, many hundreds of thousands or millions of forward-backward synthesis cycles may occur before the self-model is dramatically modified. For relatively long periods of time, small changes within the context of the existing self may suffice to allow the system to control itself intelligently.

This sort of system-theoretic speculation is difficult to validate scientifically, at the present stage of development of AI and cognitive science; yet, it is critical in terms of guiding the education of proto-AGI systems like the NCE as they interact with humans in environments like the NCE. The goal of educating the NCE in AGISim is not just to give it practical understanding of the world around it, but above all to give it a practical understanding of its own self, in relation to the world around it. By watching it do things, and explaining to it what it does; and having it watch others do things, and explain these things and react to them – in this way the baby AGI's systems self-model will originate and mature. The AGI's learning processes must be calibrated to allow both forward and backward synthesis operations to occur, specifically pertaining to patterns involving the system's perceptions of its own actions, behaviors and cognitions. In this way (assuming the cognitive operations are powerful enough and the environmental and social interactions are rich enough) the natural interaction of forward and backward cognition will lead to the emergence of an effective and growing self-model…

It is with this in mind that the Iterated Easter Egg Hunt scenario, described in Section 5 below, has been chosen for discussion and future experimentation. The goal has not been merely to choose a challenging learning problem (there are plenty of more challenging ones, of course), but rather to choose a learning problem that focuses specifically on the interaction between perception, cognition, action, and the pragmatic, contextual modeling of self and others. From this sort of interaction emerges the self; and from the actively evolving self-model emerge the seeds of more advanced autonomous general cognition.

4. An Economic Approach to Attention Allocation and Action Selection

This section briefly describes the economics-based approach to attention allocation and action selection contained in the NCE design. Along with PLN and MOSES, this is the third of the original and critical cognitive processes existing in the NCE. There are other cognitive processes, described above, that are critical but not that original in content: for instance, concept creation via blending, which is a very simple heuristic process of "cutting and pasting" Atoms; and map formation, which is essentially just conjunctive pattern mining applied to the Atom table. Although not yet in final form, PLN and MOSES have been fully implemented and experimented with, and are treated in other papers in this volume, as noted above. On the other hand, economic attention allocation and action selection have currently been implemented only in an incomplete prototype form. They will be discussed here only briefly and somewhat cursorily, the goal being to provide sufficient background that their role in the discussion of the iterated Easter Egg Hunt scenario in the following section may be understood. In terms of the above typology of cognitive processes, economic attention allocation underlies a number of processes: credit assignment, importance propagation, attention allocation, schema execution, and maintenance of the active schema and goal pools.

4.1. Economic Attention Allocation

This section outlines an approach to the allocation of processor time and memory to NCE Atoms based on the introduction of a concept of "currency" (money) into the NCE. The general advantages of currency-based attention allocation and credit assignment have been emphasized by Eric Baum in various papers (e.g. [23]; and see also discussion in [24]). The specific approach described here is not directly related to any of Baum's detailed ideas, but is grounded in the same philosophical ideas.

Two separate types of currency are introduced: STICurrency and LTICurrency, corresponding to short and long term importance values.[4] Atoms, MindAgents, and Units (the Unit, in the Novamente architecture, being a possibly distributed set of Novamente Lobes that are considered as sharing a common AtomTable for purposes of attention allocation) are then all considered as financial agents. The Units also have a unique role, as "mints" capable of producing new money. Money is not transferred between Units in this approach; each Unit has its own local, closed economy. It is assumed here, for sake of discussion, each Unit has a certain fixed total amount of currency of each type in it[5]. For starters, we may assume that this total amount is fixed for all eternity.

In the following discussion, the forgetting mechanism is assumed to be as follows: When the need to free up memory occurs, the Atoms with the least amount of LTI (i.e. of Currency of CurrencyType LTI) are removed from RAM. This is a simplistic forgetting mechanism which ignores subtler possibilities such as removing some links from some nodes with low LTI, but then letting them stay around a while in shrunken form.

On the other hand, the main point of STI is to govern the Atom-selection behavior of MindAgents. Many MindAgents choose Atoms to act upon based on their STI (STICurrency level). And, note that, in the economic approach described here, STI can pretty easily go to zero (or even become negative, representing "STI debt") if an Atom is useless for a while (i.e. an Atom can go "STI bankrupt"). If an Atom has negative

[4]

[5]

STI, it won't be selected by MindAgents using STI as a selection criterion, but it may receive STI from other Atoms related to it via the "importance spreading" dynamic, which may increase its STI to the level where it may once again be selected more frequently as an object of cognitive action.

Only the simplest variety of economic attention allocation is described here. More complexity is introduced when, for example, one introduces additional MindAgent-specific currencies, representing the STI of an Atom relative to the purposes of a given MindAgent. But these complexities are not needed for the discussion of goal and action selection and embodied social learning in the following sections, so they will be passed by for now.

4.1.1.Simple Equations for the Economics of Attention

The default equations for updating the amount of currency possessed by an Atom A, a MindAgent MA and a Unit U at a certain moment in time t are as follows.

First, equations for LTI currency:

```
   LTI_Atom(A,t+1) = LTI_Atom(A,t) - LTIAtomRent *
memoryUsage(A,t) + LTI_Atom_fee * (#times A used in cycle
t) + LTI_Atom_rewards

   LTI_MindAgent(MA,t+1) = LTI_MindAgent(MA,t) -
LTI_fees_paid(MA,t) - processor_fees_paid(MA,t) +
rewards_received(MA,t)

   LTI_Unit(U,t+1) = LTI_Unit(U,t) +
LTI_rents_received(U,t) - LTI_rewards_paid(U,t)
```

Note that LTI values may go below zero, which is fine. In this case forgetting may remove the Atoms with the biggest debt; and if there are not enough Atoms in debt, it may remove some Atoms with positive LTI net worth as well.

Next, very similar but not quite identical equations for STI currency (the only difference is the lack of the memoryUsage term in the first equation):

```
   STI_Atom(A,t+1) = STI_Atom(A,t) - STIAtomRent  +
STI_Atom_fee * (#times A used in cycle t) +
STI_Atom_rewards

   STI_MindAgent(MA,t+1) = STI_MindAgent(MA,t) -
STI_fees_paid(MA,t) -

   STI_Unit(U,t+1) = STI_Unit(U,t) +
STI_rents_received(U,t) - STI_rewards_paid(U,t)
```

Unlike with LTI, it seems best for Atoms with STI net worth <=T, where T is a specified threshold, to not be charged STIAtomRent. The argument is that if Atoms have such low net worth, they are not in the short-term memory in any useful sense, so they shouldn't have to pay for being in short-term memory. Letting Atoms accumulate a lot of STI debt would make the attentional focus sluggish to respond to new stimuli, destroying its ability to rapidly and spontaneously change focus. The threshold T is the “attentional focus boundary,” whose existence induces an emergent “short term memory” within the overall AtomTable.

Note that in this approach, unlike in many other cognitive-science-based AI architectures, STM is not a separate system component but rather a systemic-dynamic phenomenon that emerges as an outgrowth of the dynamics of STI. This emergence is not magical but occurs because of specific choices in setting up the dynamics, i.e. the attentional focus boundary and its interaction with the rest of the economic attention allocation dynamics. But it has properties going beyond the existence of the attentional focus boundary: the setting of the boundary encourages the formation of complex strange attractors of attentional flow between entities that habitually surpass the attentional focus boundary at the same time.

Note also that the sets of equations for the LTI and STI currencies are totally separate from each other. This is intentional and represents a reasoned choice. In the currently proposed scheme, both LTI and STI currencies are proper, conserved currencies, but there is no mechanism for conversion between the two of them. What is not desired is for Atoms with high LTI to be able to purchase current attention just by virtue of having high LTI. Current attention must be purchased with the currency of recent utility (STI currency), whereas memory space must be purchased with the currency of long-term utility (LTI currency).

The meaning of the terms in the above equations will now be explained. I will first explain the equations for LTI currency, and then afterwards discuss STI currency (which is similar, but has the added complexity of Hebbian currency exchange).

4.1.2.LTI Economics

The LTIAtomRent is a an amount charged to each Atom each cycle, by the Unit, for the privilege of remaining in the AtomTable. This money is decremented from the Atom’s currency balance and incremented to the Unit’s currency balance. The LTIAtomRent is defined as the rent payable by an Atom per unit of memory usage, so that Atoms that are extremely consumptive of memory (for instance Atoms corresponding to large Combo trees in the ProcedureRepository) may be charged more total rent. In the initial version this dependency on memory usage may be omitted.

The LTIWage is the amount that a Unit pays a MindAgent for being utilized by a MindAgent. This enables an Atom to accumulate more currency if it is utilized more often by MindAgents. It seems optimal to enforce the rule that all Atoms must get paid the same wage. Note that, if Atoms could charge different wages (for example, based on their STI values), and wages were decremented from the MA’s individual wealth stores directly, then MA’s would be in the position of sometimes hiring inferior Atoms just to save money, and this would lead to suboptimal intelligence on the MA’s part. It’s true that this would serve to force competition between MA’s, and that competition between MA’s will be useful in future system versions where the system is evolving its own MA’s. But even in these future systems, I think we can use better methods of enforcing competition among MA’s, which do not involve artificially impairing the MA’s intelligences.

Note that, in spite of charging the same fee, some Atoms will accumulate more LTI currency than others, because they will be selected by MA’s more often than others. And the selection, by MA’s, is based in large part on the ShortTermImportance (STI) quantity associated with Atoms. STI is not the only criterion for selection that MA’s may use: in any particular instance, a MA may select Atoms based on any method it wants, which will often mean the use of criteria specific to the particular problem and context it has at hand. But as a default, once a MA has applied any other relevant selection filters, STI is the criterion it should use to select among the remaining Atoms available. This means that there is an influence relationship between STI currency and LTI currency, but it is an indirect relationship, not based on currency transfer. Of course, there is also an influence relationship in the reverse direction: LTI values affect STI values because if an Atom's LTI gets too low, it gets forgotten and therefore cannot get utilized and cannot accumulate any STI currency. So, just because there is no direct mechanism for transforming the two currency types into each other, doesn't mean the two are unrelated; it just means the relationship is not directly "financial."

Now let us look at economics from the MindAgent's perspective. When a MindAgent utilizes processor time, it must pay the Unit some of its currency in recompense for this time (and, symmetrically, the more time it uses, the more Atoms it will stimulate, therefore the more the Unit will pay out to Atoms on account of the MA's activity.) On the other hand, the MindAgents are paid by the Unit for contributing to system goals. This means that MindAgents that are generally more useful will be able to carry out operations involving more Atoms, and more processor time.

From the Unit's point of view, finally, revenue comes from the rents paid by Atoms, and funds are disbursed to Atoms in the form of rewards for utilization by MindAgents.

4.1.3.STI Economics

STI economics is basically the same as LTI economics, but with different parameter values. STI rents are higher, so that Atoms much more easily become STI-bankrupt; and the higher rents of necessity mean the other quantities in the economy must be different (Atoms must charge higher fees, so Units must give bigger rewards). And recall that STI rents are only charged to Atoms with positive STI net worth.

4.1.4.Hebbian Rewards

Next we introduce the notion of Hebbian rewards.

The basic idea is that Atoms pay other Atoms whose utilization is expected to pave the way for their own future utilization. That is, if there is a link

```
CausalHebbianLink A B
```

(denoting the fact that utilization of A seems to cause utilization of B) then it may be worthwhile for B to give some of its wealth to A – so as to (in the case of STI currency) increase the odds that A will be chosen by MA's, or (in the case of LTI currency) increase the odds that A will be retained in memory.

CausalHebbianLinks may come with time-interval stamps. STI Hebbian rewards should be given based on CHLinks with relatively brief time-interval stamps, whereas LTI Hebbian rewards should be given based on CHLinks with any time-interval stamp.

The total amount of wealth that an Atom should be willing to give to other Atoms at any point in time is capped (no sense to an Atom bankrupting itself to support others), and depends on the truth value of the CausalHebbianLinks that actually exist pointing to the Atom.

4.2.Economics of Goal and Action Selection

Now we will describe how these economic mechanisms are intended to interact with the processes of subgoal selection and action selection, in the NCE design. The main actors (apart from the usual ones like the AtomTable, economic attention allocation, etc.) in the tale to be told here are as follows:

- Structures:
    - Supergoal Pool
    - Active Schema Pool
- MindAgents:
    - GoalBasedSchemaSelection
    - GoalBasedSchemaLearning
    - GoalAttentionAllocation
    - FeasibilityUpdating
    - SchemaActivation

4.2.1.Supergoal Pool

The Supergoal Pool contains the Atoms that the system considers as top-level goals. These goals must be treated specially by attention allocation: they must be given funding by the Lobe so that they can use it to pay for getting themselves achieved. The weighting among different top-level goals is achieved via giving them differential amounts of currency. STICurrency is the key kind here, but of course top-level supergoals must also get some LTICurrency so they won't be forgotten. (Inadvertently deleting your top-level supergoals from memory is considered to be a bad thing!)

4.2.2.Promissory transfer of STI funds between goals

Transfer of "attentional funds" from goals to subgoals, and schema modules to other schema modules in the same schema, takea place via a mechanism of *promises of funding* (or "requests for service," to be called "RFS's" from here on). This mechanism relies upon and interacts with ordinary economic attention allocation but also has special properties.

The logic of these RFS's is as follows. If agent A issues a RFS of value x to agent B, then

1. When B judges it appropriate, B may redeem the note and ask A to transfer currency of value x to B.
2. A may withdraw the note from B at any time.

(There is also a little more complexity here, in that we will shortly introduce the notion of RFS's whose value is defined by a set of constraints. But this complexity does not contradict the two above points.) The total value of the of RFS's possessed by an Atom may be referred to as its "promise."

Now we explain how RFS's may be passed between goals. Given two predicates A and B, if A is being considered as a goal, then B may be considered as a subgoal of A (and A the supergoal of B) if there exists a relationship of the form

```
PredictiveImplication B A
```

I.e., achieving B may help to achieve A. Of course, the strength of this link and the temporal characteristics of this link are important in terms of quantifying how strongly and how usefully B is a subgoal of A.

Supergoals (not only top-level ones) allocate RFS's to subgoals as follows. Supergoal A may issue a RFS to subgoal B if it is judged that achievement (i.e., predicate satisfaction) of B implies achievement of A. This may proceed recursively: subgoals may allocate RFS's to subsubgoals according to the same justification.

Unlike actual currency, RFS's are not conserved. However, the actual payment of real currency upon redemption of RFS's obeys the conservation of real currency. This means that agents need to be responsible in issuing and withdrawing RFS's. In practice this may be ensured by having agents follow a couple simple rules in this regard.

3. 1. If B and C are two alternatives for achieving A, and A has x units of currency, then A may promise both B and C x units of currency. Whomever asks for a redemption of the promise first, will get the money, and then the promise will be rescinded from the other one.
4. 2. On the other hand, if the achievement of A requires both B and C to be achieved, then B and C may be granted RFS's that are defined by constraints. If A has x units of currency, then B and C receive an RFS tagged with the constraint (B+C<10). This means that in order to redeem the note, either one of B or C must confer with the other one, so that they can simultaneously request constraint-consistent amounts of money from A.

As an example of the role of constraints, suppose that the goal is to play fetch successfully (a subgoal of "get reward")… Then suppose it is learned that

```
ImplicationLink
   AND
         get_ball
         deliver_ball
   play_fetch
```

Then, if play_fetch has $10 in STICurrency, it may know it has $10 to spend on a combination of get_ball and deliver_ball. In this case both get_ball and deliver_ball would be given RFS's labeled with the contraint

```
RFS.get_ball + RFS.deliver_ball <= 10
```

The issuance of RFS's embodying constraints is different from (and generally carried out prior to) the evaluation of whether the constraints can be fulfilled.

A supergoal may rescind offers of reward for service at any time. And, generally, if a subgoal gets achieved and has not spent all the money it needed, the supergoal will not offer any more funding to the subgoal (until/unless it needs that subgoal achieved again).

As there are no ultimate sources of RFS in Novamente besides top-level supergoals, promise may be considered as a measure of "goal-related importance."

Transfer of RFS's among Atoms is carried out by the GoalAttentionAllocation MindAgent.

4.2.3.Feasibility Structures

Next, there is a numerical data structure associated with goal Atoms, which is called the feasibility structure. The feasibility structure of an Atom G indicates the feasibility of achieving G as a goal using various amounts of effort. It contains triples of the form (t, p, E) indicating the truth value t of achieving goal G to degree p using effort E. Feasibility structures must be updated periodically, via scanning the links coming into an Atom G; this may be done by a FeasibilityUpdating MindAgent. Feasibility may be calculated for any Atom G for which there are links of the form

```
Implication
   Execution S
   G
```

for some S. Once a schema has actually been executed on various inputs, its cost of execution on other inputs may be empirically estimated. But this is not the only case in which feasibility may be estimated. For example, if goal G inherits from goal G1,and most children of G1 are achievable with a certain feasibility, then probably G is achievable with that same feasibility as well. This allows feasibility estimation even in cases where no plan for achieving G yet exists, e.g. if the plan can be produced via predicate schematization, but such schematization has not yet been carried out.

Feasibility then connects with importance as follows. Important goals will get more STICurrency to spend, thus will be able to spawn more costly schemata. So, the GoalBasedSchemaSelection MindAgent, when choosing which schemata to push into the ActiveSchemaPool, will be able to choose more costly schemata corresponding to goals with more STICurrency to spend.

4.2.4.Goal Based Schema Selection

Next, the GoalBasedSchemaSelection selects schemata to be placed into the ActiveSchemaPool. It does this by choosing goals G, and then choosing schemata that are alleged to be useful for achieving these goals. It chooses goals via a fitness function that combines promise and feasibility. This involves solving an optimization problem: figuring out how to maximize the odds of getting a lot of goal-important stuff done within the available amount of (memory and space) effort. Potentially this optimization problem can get quite subtle, but initially some simple heuristics are satisfactory. (One subtlety involves handling dependencies between goals, as represented by constraint-bearing RFS's.).

Given a goal, the GBSS MindAgent chooses a schema to achieve that goal via the heuristic of selecting the one that maximizes a fitness function balancing the estimated effort required to achieve the goal via executing the schema, with the estimated probability that executing the schema will cause the goal to be achieved.

When searching for schemata to achieve G, and estimating their effort, one factor to be taken into account is the set of schemata already in the ActiveSchemaPool. Some schemata S may simultaneously achieve two goals; or two schemata achieving different goals may have significant overlap of modules. In this case G may be able to get achieved using very little or no effort (no additional effort, if there is already a schema S in the ActiveSchemaPool that is going to cause G to be achieved). But if G decides it can be achieved via a schema S already in the ActiveSchemaPool, then it should still notify the ActiveSchemaPool of this, so that G can be added to S's index (see below). If the other goal G1 that placed S in the ActiveSchemaPool decides to withdraw S, then S may need to hit up G1 for money, in order to keep itself in the ActiveSchemaPool with enough funds to actually execute.

4.2.5.SchemaActivation

Next, what happens with schemata that are actually in the ActiveSchemaPool? Let us assume that each of these schema is a collection of modules, connected via ActivationLinks, which have semantics: (ActivationLink A B) means that if the schema that placed module A in the schema pool is to be completed, then after A is activated, B should be activated.

When a goal places a schema in the ActiveSchemaPool, it grants that schema an RFS equal in value to the (some fraction of) the (promissory+real) currency it has in its possession. The heuristics for determining how much currency to grant may become sophisticated; but initially we may just have a goal give a schema all its promissory currency; or in the case of a top-level supergoal, all its actual currency.

When a module within a schema actually executes, then it must redeem some of its promissory currency to turn it into actual currency, because executing costs money (paid to the Lobe). Once a schema is done executing, if it hasn't redeemed all its promissory currency, it gives the remainder back to the goal that placed it in the ActiveSchemaPool.

When a module finishes executing, it passes promissory currency to the other modules to which it points with ActivationLinks.

The network of modules in the ActiveSchemaPool is a digraph (whose links are ActivationLinks), because some modules may be shared within different overall schemata. Each module must be indexed via which schemata contain it, and each schema must be indexed via which goal(s) want it in the ActiveSchemaPool.

4.2.6.GoalBasedSchemaLearning

This, finally, refers to the process of trying to figure out how to achieve goals, i.e. trying to learn links between ExecutionLinks and goals G. This process should be focused on goals that have a high importance but for which feasible achievement-methodologies are not yet known. Predicate schematization is one way of achieving this; another is MOSES procedure evolution.

5. Embodied Social Learning in the Iterated Easter Egg Hunt Scenario

The goal of this section is to discuss how all the different aspects of the NCE design are intended cooperate to allow the system to carry out a moderately complex early-childhood-level task (iterated Easter Egg Hunt, or IEEH) in the AGISim world. The skeptical reader may be justified in viewing this section as a kind of highly technical science fiction, as the NCE has not yet been applied to this task, and will not be until a bit more development has been done. However, as argued above, I believe it is necessary to richly and deeply conceptualize the holistic behavior of an AGI system prior to designing its parts in detail (let alone implementing and testing its parts).

Note, it is not claimed that the approach described here is the optimal approach to solving the IEEH problem, either within the NCE or outside of it. Rather, IEEH is being used to exemplify the interaction of various cognitive mechanisms. An optimal IEEH agent would likely learn much less from IEEH than either the NCE or a young human child.

An expected consequence of solving IEEH using a sophisticated cognitive approach (rather than, say, an operations-research or machine-learning approach) is superior generalization capability. For instance, suppose an NCE instance has learned how to play IEEH effectively using the general approach described here. Then it should have a much easier time learning to play hide-and-seek than an NM instance that has a similar background except that it has not learned how to play IEEH effectively. This kind of "transfer learning" is a key method of assessing the extent to which a task (like IEEH) has been learned in a way supporting general intelligence versus a narrow-AI/machine-learning sort of way (the latter tending to involve overfitting to the particular task). What one expects to see is that after a task T is learned, the learning of other tasks S becomes easier, with the degree of increased easiness being proportional to the similarity between S and T. While it is not clear what similarity measure is best used here, if we are aiming for roughly human-like intelligence then qualitative similarity according to human judgment is adequate.

5.1.Definition of Iterated Easter Egg Hunt

Firstly, "Easter Egg Hunt" is defined as a game in which one agent hides a bunch of eggs, and a group of other agents try to find them.

The main goal of each finder agent is to find as many eggs as possible. There may be other goals too, such as allowing each other agent to find at least one egg; or, finding more eggs than any other agent.

And, the main goal of the hider agent is to cause the finder agents to need to take a long time to find the eggs. Again, other goals may also be used in parallel, such as making it likely that each finder will get to find some eggs, rather than one seeker finding all the eggs.

Next, "Iterated Easter Egg Hunt" is defined as repetition of Easter Egg Hunt N times within a group of K agents, with different agents being the hider each time (the most interesting case is where N>K so everyone gets to be hider more than once).

5.2.Examples of Learning in IEEH

Useful patterns that may be recognized by an intelligent agent operating in an IEEH scenario include:

- Short agents cannot either hide or find eggs in very high places
- Short agents are more likely than tall ones to find eggs hidden in very low places (e.g. under a couch)
- Some agents may repetitively hide eggs in the same places each time they're serving as hider
- Some agents may try to hide eggs in different places each time they're serving as hider
- Once an agent A has found an egg in a certain place P, or seen another agent finding an egg in that place, then A is more likely to look in P again in the future

These patterns may be helpful to guide both hiding and finding behavior.

5.3.Pattern Mining, MOSES and Inference in IEEH

Many of the patterns discussed above may be found by pattern mining, and then validated by inference. In this subsection specific examples of this are described.

First of all, we may suppose that the system collects information such as

```
egg_134 found under couch_4 by agent_3
```

because it knows that "find" is a relevant predicate to the IEEH situation, so it tells the perception MindAgent to identify and record observed instances of "find"-ing.

If the system also has a general inclination to think about the effort levels being expended by agents, it may also collect information such as

```
egg_134 found under couch_4 by agent_3 with apparent relative effort level LOW
```

(which might be expressed explicitly in Novamente Atoms, e.g., by

```
Evaluation [1]
   find
         agent_3
         egg_134
         SatisfyingSet
               Evaluation
                     under
                           $1
                           couch

Inheritance [2] Easter_Egg_Hunt
Inheritance [2] indoor

Context
   [2]
   Evaluation
         Effort
               [1]
               Low
)
```

Mining a collection of relationships of this form, using simple conjunctive pattern mining, may lead to patterns such as

```
Finding eggs hidden in drawers tends to be hard
```

The system may also record knowledge such as

```
agent_3 has height around 1 meter
```

as well, if it has the habit of recording physical properties of other agents.

Based on all this data, MOSES-driven pattern mining may easily discover the pattern

```
   Short agents have much higher odds of finding eggs
hidden under the couch than tall agents do
```

It may also perhaps discover patterns such as

```
   Short agents have much higher odds of finding eggs
hidden in the bottom drawer of a cabinet than tall agents
do
```

(For this, MOSES would be given a fitness function defined by a measure of "statistical interestingness.")

Now, where does PLN come into the process? Well, the system may also have a concept of "low" (in terms of height) and then be able to learn from the above knowledge, via simple PLN inference, that

```
   Short agents have much higher odds of finding eggs
hidden in low places than tall agents do
```

Next, supposing this statement has been learned via a combination of conjunctive pattern mining, MOSES and PLN, as described above. Then, it may be given a high importance value, because of its relevance to the current goals, and its surprisingly high truth value. Given this high importance, more attention will be focused on it.

Among other things that may happen because of this attention, inference will focus on the Atom, and resulting from this, generalization may occur. A good generalization would be

```
   Agents have relatively high odds of finding eggs that
they can relatively easily see
```

Further generalization may teach the system that

```
RetroactiveImplication
   Evaluation find ($1, $2, *)
   Evaluation see ($1, $2)
```

(i.e., usually when an agent finds something, the agent has recently seen that thing.)

This may make the system assign the "see" predicate a high importance, which among other things may cause MOSES to focus on this predicate. The system has many examples of things it has seen and things it has not seen, and may mine this database of knowledge to learn patterns regarding seeing. It may note, for instance, that on several occasions it could not see a certain egg, and then when another agent moved some object that was in front of that egg, afterwards it could see that egg. If the system has a general category "agent" that abstracts both itself and the other agents, then it may learn from this the predicate

```
    If I move an object, I may see an egg that I did not
see before
```

This predicate may then, via the process of predicate schematization, be used to generate a schema that finds and moves objects, hoping to find eggs behind them.

5.4.Schema Execution in IEEH

Next, as an example of the role of multiple goals and schema execution in the Easter Egg Hunt scenario, let us consider the goals of:

- G1: Find as many eggs as possible
- G2: Find more eggs than anybody else

Suppose the NCE-controlled agent sees one egg (Egg_1) across the room, under the couch, with no one else evidently pursuing it; and sees two other eggs (Egg_2 and Egg_3) on a shelf across the room, and is not sure whether Agent_2 is pursuing Egg_2 or some other egg. Then

- G1 will spawn a schema S1 oriented toward retrieving Egg_2 and Egg_3
- G2 will spawn a schema S2 oriented toward retrieving Egg_1

Now there are a couple possibilities:

- Both schemata (S1 and S2) may be put into the ActiveSchemaPool at the same time, and given an amount of STICurrency commensurate with the importance of the parent goal.
- Predicate schematization may be asked to find a single schema serving the combined goal (G1 OR G2); and this single schema is then put in the ActiveSchemaPool

As an example of subgoaling, consider G2 above: find more eggs than anybody else. It may be difficult for the system to continually monitor how many eggs each other agent is finding. Thus, a good strategy would be for the system to learn implications such as

```
Implication (G3 AND G4) G2 <.9>
```

where

- G3 = find more eggs than Bill
- G4 = find more eggs than Bob

(which would be the case e.g. if Bill and Bob were generally the fastest egg-finders in the bunch.) Once this implication has been learned, the FeasibilityUpdating MindAgent has to notice it (which it will do, since it looks for potentially useful implications implying currently important goals), and then make feasibility evaluations regarding G3 and G4. A little inference will tell it that G3 and G4 are probably both more feasible (lower-cost) to achieve than G2, information that may then be recorded in the feasibility structures attached to these Atoms. Then, the GoalAttentionAllocation MindAgent will cause G2 to issue RFS's to G3 and G4; and the GoalBasedSchemaSelection MindAgent will quite likely select G3 and G4 to generate schemata to be placed in the ActiveSchemaPool.

5.5.Map formation in the Understanding of Self and others

Recall that one of the above inferences assumes the system has learned a notion of Agent that encompasses both itself and the other agents in the game. It's fair to assume that this abstraction has been learned prior to the system being able to grapple with a game as socially complex as IEEH. But still it's worth discussing how this learning may occur. This topic does not have to do with IEEH in particular, so it may be considered a kind of digression or appendix to the overall theme of IEEH.

Among other possible routes, this general notion of Agent may potentially be learned via pure unsupervised pattern mining.

First of all, the collection of body parts associated with a particular agent is a good example of a learned map: the body parts associated with some particular agent are all going to be associated with each other habitually, according to many different associations, and so a map should form for each one. (This is just an example of the role of map formation in "object recognition".) Mechanistically, these maps are initiall formed via the implicit activity of attention allocation and HebbianLink formation, which causes links to form between Atoms that are often utilized together in cognitive processes. Then the MapEncapsulation MindAgent will cause Atoms to form explicitly representing these maps, which means that the maps may be explicitly utilized within processes such as inference, pattern mining and MOSES.

So, suppose the agent has a body-map Atom for each of a number of other agents and also a body-map Atom for itself, then it has the opportunity to observe that all these body-maps are somewhat similar, and hence to cluster them together (via the Clustering MindAgent). Now there is a BodyMapCluster node, which may be studied analytically via inference and MOSES, leading to the emergence of explicit general knowledge regarding what constitutes a body, and the relations between aspects of bodies. For instance, it may be noted that when a body moves, this is generally associated with certain classes of leg and foot movements.

The system may then note that when its own body-map displays movement-associated leg and foot movements, this is associated with the execution of certain motoric schemata, internally. This merely requires conjunctive pattern mining, applied to a set of predicates involving both abstract predicates observed in the perceptual stream (the abstract predicate of movement-associated leg and foot movements), and also predicates involving motoric commands. What the system learns here is "When I carry out these particular actions, the result is that I carry out the action that I have identified as 'walking'." In other words, it has learned that certain leg and foot movements cause it to move in the same way that it observes other agents moving.

A host of different associations similar to this one may be mined, based on studying internal actions (mostly motoric, at this early stage) and their relationship to observed events involving the body-map. And, these various associations are often usefully considered together in reasoning about how to achieve various goals via coordinating actions. Because of these, these various associations will be Hebbianly associated – and ultimately the MapEncapsulation MindAgent may form a number of maps involving them. This is the root of the "self-model," as it exists in infantile embodied agents.

The use of this kind of self-model and other-model in the IEEH context should be fairly obvious. For instance, the system may observe that Agent_4 and Agent_7 are more similar to it body-map-wise than any of the other agents. If it has learned that agents with similar body-maps often carry out similar behaviors, then it may figure that it is better off trying to imitate what Agent_4 and Agent_7 do, egg-hunting-wise, rather than trying to imitate other agents. It may then focus its imitative efforts on imitating the best egg-finders, and also the egg-finders most similar to it.

6. Conclusion

Achieving artificial general intelligence at the human level and ultimately beyond is a large, ambitious task. Above, building on prior review papers, I have summarized some general aspects of the NCE design, and explained roughly how we intend them to work together in addressing a moderately complex early-childhood-level task, the iterated Easter Egg Hunt. / And I have professed the opinion that the cognitive processes required for this scenario are the same ones required for more complex, adult-level learning and reasoning.

More generally, why do I believe it is plausible to assert that the NCE design may actually be capable of achieving highly advanced general intelligence? The simplest answer consists of two very general points:

- The NCE is based on a *well-reasoned, truly comprehensive theory of cognition*, covering both the concretely-implemented and emergent aspects
- The specific algorithms and data structures chosen to implement this theory of mind are efficient, robust and scalable. And, so is the software implementation

A more nuanced answer refers to the system-theoretic ideas introduced in ([14, 25, 26]) and elaborated above: In the NCE design, forward and backward synthesis are implemented in a powerful and general enough way adequate to give rise to self and focused consciousness as strange attractors. This, is the crux of why, in my view, the NCE will very likely be able to give rise to powerful general intelligence. To yield powerful AGI, the “mechanics” of a system has to be right – procedures have to get executed, basic probabilistic conclusions have to be drawn, useless knowledge has to be forgotten, etc. But all this won’t give rise to powerful intelligence unless the overall system is properly configured so as to give rise to the right emergent structures, key among which are the phenomenal self and the moving bubble of consciousness. NCE has been designed with this sort of emergence specifically in mind. Assuming the project continues as planned, the next years of work will tell us whether this methodology of emergence-oriented design, as instantiated in the NCE, is really as powerful as expected.

Endnotes

[2] A thorough technical review of the Novamente design has not yet been published; and whether and when such publication will occur is a subject of uncertainty. The main issue is not that the Novamente codebase is proprietary, but rather the issue of “AI safety” ([10, 27, 28, 29]). Given the premise that the Novamente design may actually be capable of being used to create a human-level software intelligence, it then becomes ethically debatable whether the design should be placed in the public domain, due to the

potential that someone may use the design to create a dangerous human-level software intelligence. Of course, going from a design to a working and educated implementation would be a large task for anyone, but the possibility is there, and has given us pause regarding the publication of too many details of the Novamente approach. On the other hand, we are still interested in sharing ideas with the research community and getting their feedback, and thus for the time being we have chosen the path of discussing the high-level aspects of the system design in papers such as this one, but not sharing technical details.

[4] An alternate approach involving only one currency was also considered. In this approach, STICurrency was taken as the basic currency, and the role of LTICurrency was played by the notion of a "credit rating." While interesting and workable, the added complexity of this approach was judged not worthwhile, in spite of the conceptual elegance of having only a single currency type.

[5] Later on, in a more advanced version of the NCE, this amount may eventually be allowed to change slowly over time – if it is found, for example, that periodically inserting more currency into the economy to cause a small rate of inflation encourages intelligence. This would not be the case in the current system because none of the financial agents are initially carrying out economic actions with any real flexibility or intelligence (though their simple and mechanistic economic actions are enabling the system as a whole to carry out actions with some level of intelligence).

References

[1] Looks, Moshe (2006). Program Evolution for General Intelligence. Proceeding of AGI Workshop 2006, Bethesda MD, IOS Press

[2] Goertzel, Ben, Ari Heljakka, Stephan Vladimir Bugaj, Cassio Pennachin, Moshe Looks *(2006). Exploring Android Developmental Psychology in a Simulation World.* Proceedings of ICCS-2006, Vancouver

[3] Inhelder B. and J. Piaget. *The Growth of Logical Thinking from Childhood to Adolescence*. New York: Basic Books, 1958.

[4] Goertzel, Ben and Cassio Pennachin (2006). The Novamente Design for Artificial General Intelligence. In *Artificial General Intelligence*, Springer-Verlag.

[5] Goertzel, Ben (2006). *Patterns, Hypergraphs and General Intelligence*. Proceedings of International Joint Conference on Neural Networks, IJCNN 2006, Vancouver CA, to appear

[6] Goertzel, Ben, C. Pennachin, A. Senna, T. Maia, G. Lamacie. (2003) "Novamente: an integrative architecture for Artificial General Intelligence." *Proceedings of IJCAI 2003 Workshop on Cognitive Modeling of Agents.* Acapulco, Mexico, 2003

[7] Goertzel, Ben, C. Pennachin, A. Senna,, M. Looks. (2004) "The Novamente Artificial General Intelligence Architecture." *Proceedings of AAAI Symposium on Achieving Intelligence Through Integrated Systems And Research.* Washington DC, 2004

[8] Metzinger, Thomas (2004). *Being No One*. MIT Press

[9] Goertzel, Izabela, Ben Goertzel, Ari Heljakka, Hugo Pinto and Cassio Pennachin (2006). Automated Biological Hypothesis Discovery via Probabilistic Inference on Dependency Grammar Parses of PubMed Abstracts, Proceeding of AGI Workshop 2006, Bethesda MD, IOS Press

[10] Goertzel, Ben and Stephan Vladimir Bugaj (2006). Stages of Development in Uncertain-Logic-Based AI Systems, Proceeding of AGI Workshop 2006, Bethesda MD, IOS Press

[11] Franklin, Stan (2006). A Foundational Architecture for Artificial General Intelligence. Proceeding of AGI Workshop 2006, Bethesda MD, IOS Press

[12] Sloman, A. 1999. What Sort of Architecture is Required for a Human-like Agent? In *Foundations of Rational Agency*, ed. M. Wooldridge, and A. S. Rao. Dordrecht, Netherlands: Kluwer Academic Publishers.

[13] Ikle', Matthew, Ben Goertzel and Izabela Goertzel (2006). Indefinite Probabilities for General Intelligence, Proceeding of AGI Workshop 2006, Bethesda MD, IOS Press

[14] Looks, Moshe (2006). Program Evolution for General Intelligence. Proceeding of AGI Workshop 2006, Bethesda MD, IOS Press

[15] Goertzel, Ben (2006a). A System-Theoretic Analysis of Focused Cognition, and its Implications for the Emergence of Self and Attention. Dynamical Psychology.

[16] Fauconnier, Gilles and Turner, Mark (2002). *The Way We Think: Conceptual Blending and the Mind's Hidden Complexities*. Basic Books.

[17] Lenat, D. and R. V. Guha. (1990). *Building Large Knowledge-Based Systems: Representation and Inference in the Cyc Project*. Addison-Wesley

[18] Feldbaum, Christiane (1998) "WordNet: an electronic lexical database", Cambridge, The MIT press.

[19] Fillmore, Charles J., Collin F. Baker, and Hiroaki Sato. (2002). The framenet database and software tools. In Proceedings of the Third International Conference on Languag Resources and Evaluation, volume IV, Las Palmas. LREC.

[20] Goertzel, Ben (2006b). Lojban++: An Efficient, Minimally Ambiguous, User-Friendly Natural-Like Language for Human-Computer, Computer-Computer and Human-Human Communication, online at http://www.goertzel.org/papers/lojbanplusplus.pdf

[21] Thelen, E. and L. Smith. (1994). *A Dynamic Systems Approach to the Development of Cognition and Action*. Cambridge, MA: MIT Press.

[22] Tomasello, Michael (2005). Constructing a Language. Harvard University Press.

[23] Baum, Eric (1998). "Manifesto for an Evolutionary Economics of Intelligence" in "Neural Networks and Machine Learning" Editor C. M. Bishop, Springer-Verlag (1998), pp 285-344.

[24] Baum, Eric (2004). What Is Thought? MIT Press

[25] Goertzel, Ben (1994). *Chaotic Logic*. Plenum, New York.

[26] Goertzel, Ben (1997). *From Complexity Creativity*. Plenum, New York.

[27] Bostrom, Nick (2002). Existential Risks: Analyzing Human Extinction Scenarios and Related Hazards. Journal of Evolution and Technology, vol. 9

[28] Yudkowsky, Eliezer (2007). Cognitive Biases Potentially Affecting Judgment of Global Risks, in Global Catastrophic Risks, Ed. by Nick Bostrom and Milan Cirkovic, Oxford University Press

[29] Yudkowsky, Eliezer (2007). Artificial Intelligence and Global Risk, in Global Catastrophic Risks, Ed. by Nick Bostrom and Milan Cirkovic, Oxford University Press