An Integrative Methodology for Teaching Embodied Non-Linguistic Agents, Applied to Virtual Animals in Second Life

Ben GOERTZEL, Cassio PENNACHIN, Nil GEISSWEILLER, Moshe LOOKS, Andre SENNA, Welter SILVA, Ari HELJAKKA, Carlos LOPES

Novamente LLC, Washington DC

Abstract. A teaching methodology called Imitative-Reinforcement-Corrective (IRC) learning is described, and proposed as a general approach for teaching embodied non-linguistic AGI systems. IRC may be used with a variety of different learning algorithms, but it is particularly easily described in EC lingo. In these terms, it is a framework for automatically learning a procedure that generates a desired type of behavior, in which: a set of exemplars of the target behavior-type are utilized for fitness estimation; reinforcement signals from a human teacher are used for fitness evaluation; and the execution of candidate procedures may be modified by the teacher via corrections delivered in real-time. An example application of IRC to teach behaviors to AI-controlled artificial animals embodied in the Second Life virtual world is described in detail, including a review of the overall virtual-animal-control software architecture and how the integrative teaching/learning methodology fits into it. In this example application architecture, the learning algorithm may be toggled between hillclimbing and probabilistic evolutionary learning. Envisioned future applications are also discussed, including an application to embodied language learning applicable to agents in Second Life and other virtual worlds.

Keywords. Reinforcement learning, imitative learning, corrective learning, evolutionary programming, hill-climbing, MOSES, intelligent virtual agents

Introduction

Supposing one intelligent agent (the "teacher") has knowledge of how to carry out a certain behavior, and wants to transfer this knowledge to another intelligent agent (the "student"). But, suppose the student agent lacks the power of language (which might be, for example, because language is the thing being taught!). How may the knowledge be transferred? At least three methodologies are possible:

- **Imitative learning**: The teacher acts out the behavior, showing the student by example

- **Reinforcement learning**: The student tries to do the behavior himself, and the teacher gives him feedback on how well he did
- **Corrective learning**: As the student attempts the behavior, the teacher actively corrects (i.e. changes) the student's actions, guiding him toward correct performance

Obviously, these three forms of instruction are not exclusive. What we describe here, and call IRC learning, is a pragmatic methodology for instructing AGI systems that combines these three forms of instruction. We believe this combination is a potent one, and is certainly implicit in the way human beings typically teach young children and animals.

We present IRC learning here primarily in the context of virtually embodied AGI systems – i.e., AGI systems that that control virtual agents living in virtual worlds. There is an obvious extension to physical robots living in the real world and capable of flexible interaction with humans. In principle, IRC learning is applicable more broadly as well, and could be explored in various non-embodied context such as (for instance) automated theorem-proving. In general, the term "IRC learning" may be used to describe any teacher/student interaction that involves a combination of reinforcement, imitation and correction. While we have focused in our practical work so far on the use of IRC to teach simple "animal-like" behaviors, the application that interests us more in the medium term is language instruction, and we will enlarge upon this a bit in the Conclusion.

In collaboration with The Electric Sheep Company, our software firm Novamente LLC is currently in the midst of creating a large-scale commercial implementation of IRC learning, as a methodology for teaching virtual animals in Second Life and other online virtual worlds. The virtual animals we are now experimenting with are nonlinguistic animals that can carry out spontaneous behaviors while seeking to achieve their own goals. and can also specifically be trained by human beings to carry out novel tricks and other behaviors (which were not programmed into them, but rather must be learned by the AI on the fly based on interaction with an avatar controlled by a human teacher). This current experimental work will be used as the basis of a commercial product to be launched sometime in 2008.

In Section 2 we will give a brief overview of our virtual-animal software architecture, and explain how the IRC methodology fits in, utilizing either hillclimbing or evolutionary learning, allied with simple inference, as the underlying learning engine (aspect 3 above). This software is work-in-progress and we don't yet have anywhere near a full understanding of what the strengths and limitations of the IRC methodology will be in this context, but it has already proved capable of learning some simple behaviors and we are confident it will prove considerably extensible. After describing this Second Life virtual animal application in detail, we then briefly review our plans for future R&D, which include a subsequent application to embodied language learning in Second Life and other virtual worlds.

1.1. IRC Learning in the Context of the Quest for Powerful AGI

One may decompose the overall task of creating a powerful AGI system into four

aspects (which of course are not entirely distinct, but still are usefully distinguished):

1. **Cognitive architecture** (the overall design of an AGI system: what parts does it have, how do they connect to each other)
2. **Knowledge representation** (how does the system internally store declarative, procedural and episodic knowledge; and now does it create its own representation for knowledge of these sorts in new domains it encounters)
3. **Learning** (how does it learn new knowledge of the types mentioned above; and how does it learn how to learn, and so on)
4. **Teaching methodology** (how is it coupled with other systems so as to enable it to gain new knowledge about itself, the world and others)

This article focuses on the fourth of these aspects, presenting some ideas about AGI teaching methodology that we believe to have quite general significance, although they were developed in the context of a specific AGI system (the Novamente Cognition Engine) that is founded on specific commitments regarding the other three aspects. For the first author's views on the other three aspects of AGI, in the context of the NCE and more generally, the reader is directed to [1-3] and other prior publications. The focus on the fourth aspect in this paper should not be intended as a slight to the other three aspects: this is a brief paper and focuses narrowly on one part of the problem, but we mustn't forget that it's only one part of the problem. We need to build our AGI systems well, but we also need to teach them well, and it's important to understand fully and deeply exactly what that means. The research presented here has been conducted under the working hypothesis that, via constructing appropriately-architected AGI systems and then instructing them appropriately in virtual worlds, it may be possible to move from the current relatively primitive state of technology to an advanced level of AGI much more rapidly than the bulk of AI researchers believe.

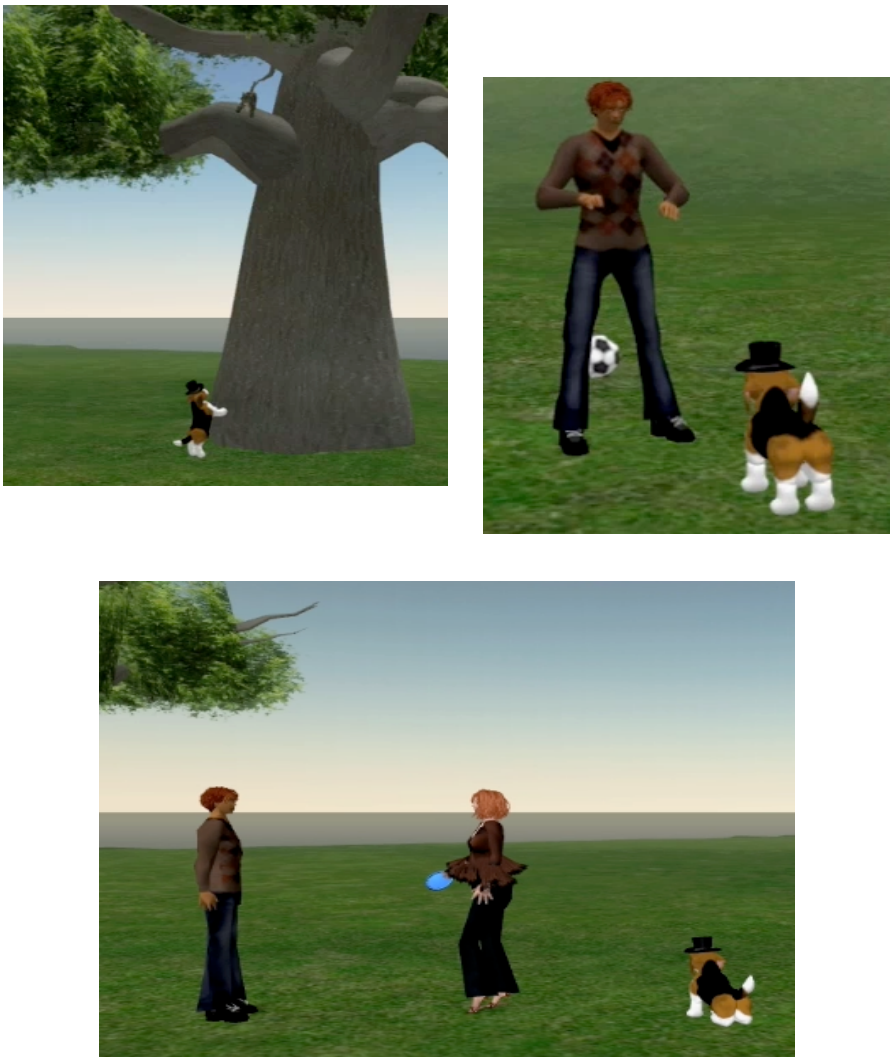

Figure 1. Screenshots from Second Life, illustrating various behaviors of a Novamente-AI-controlled virtual dog. The dog chasing a cat illustrates spontaneous behavior driven by the dog's internal goals; the figure on the upper right illustrates single-avatar teaching (of soccer skills); the bottom figure illustrates two-avatar teaching (of frisbee skills).

1.2. The Power of Virtual Worlds for Accelerating Progress toward Powerful AGI

From an AI theory perspective, virtual worlds may be viewed as one possible way of providing AI systems with *embodiment*. The issue of the necessity for embodiment in

AI is an old one, with great AI minds falling on both sides of the debate. The classic GOFAI systems (see [4] for a high-level review) are embodied only in a very limited sense; whereas [5] and others have argued for real-world robotic embodiment as the golden path to AGI. Our own view is somewhere in the middle: as outlined in [3] we suspect embodiment is very useful though probably not strictly necessary for AGI, and we also suspect that at the present time, it is probably more generally worthwhile for AI researchers to spend their time working with virtual embodiments in digital simulation worlds, rather than physical robots.

The notion of virtually embodied AI is nowhere near a new one, and can be traced back at least to Winograd's (1972) classic SHRDLU system. However, technology has advanced a long way since SHRDLU's day, and the power of virtual embodiment to assist AI is far greater in these days of Second Life, Word of Warcraft, HiPiHi, Creatures, Club Penguin and the like. In 2004 and 2005 we experimented with virtual embodiment in simple game-engine type domains, customized for AGI development (see Figure 3), which has advantages in terms of the controllability of the environment; but we are now leaning toward the conclusion that the greater advantage is to be had by making use of the masses of potential AGI teachers present in commercial virtual worlds.

Virtually embodied AGI in virtual worlds may take many different forms, for instance (to name just a handful of examples):

- ambient wildlife
- virtual pets
- virtual babies for virtual-world residents to take care of, love, and teach
- virtual shopkeepers
- virtual job recruiters (note that dozens of real-world companies are now using human-controlled Second Life avatars to do job recruiting)
- digital twins, imitating users' avatars

To concretely understand the potential power of virtual embodiment for AGI, in this essay we'll focus mostly on just one possibility – the virtual-animal product mentioned above. Possible follow-up products may include virtual talking parrots, and virtual humanoid babies – topics to be briefly discussed in the Conclusion.

2. Creating Virtual Animals for Second Life

Next we discuss some of the practical steps we are currently taking, aimed at gradually realizing the above-described vision. We briefly describe the software architecture we have developed for controlling virtual animals in Second Life – which for the sake of this discussion we will call the Virtual Animal Brain (VAB). This architecture contains some nontrivial AI within it, related to action selection and the overall relationship between goals, procedures and contexts. However, we describe it here primarily to provide context for the discussion in the following two sections, which deal with the combined use of imitative and reinforcement learning in the context of learning embodied behaviors via evolutionary program learning, hillclimbing, and associative memory.

The capabilities of our virtual animals, in their current form, include

- Spontaneous exploration of the environment
- Automated enactment of a set of simple predefined behaviors
- Efficient learning of another set of predefined behaviors
- Flexible trainability: i.e., (less efficient) learning of flexible behaviors invented by pet-owners on the fly
- Communication with the animals, for training of new behaviors and a few additional purposes, occurs in a special subset of English here called ACL (Animal Command Language)
- Individuality: each animal has its own distinct personality
- Spontaneous learning of new behaviors, without need for explicit training

Our main focus here will be on the "flexible trainability" aspect, but we will also touch on other aspects as appropriate. And, though we won't stress this point, the same ideas discussed here in the context of teacher-focused learning, may also be used in a slightly modified form to enable spontaneous learning based on embodied experience gathered during self-directed world-exploration.

Beyond the above, some capabilities intended to be added in relatively-near-future VAB versions include

- Recognition of novel categories of objects, and integration of object recognition into learning
- Generalization based on prior learning, so as to be able to transfer old tricks to new contexts
- Use of computational linguistics (integrated into the Novamente Cognition Engine, which as described below underlies the VAB) to achieve a more flexible conversational facility.

These will also be briefly discussed below.

The VAB architecture described here is not particular to Second Life, but has been guided somewhat by the particular limitations of Second Life. In particular, Second Life does not conveniently lend itself to highly detailed perceptual and motoric interaction, so we have not dealt with issues related to these in the architecture to be described here[1]. At the moment we can run our virtual animal software either in Second Life or in a simple Tcl/Tk testing world we have created; but with modest effort the VAB system could be ported to operate in essentially any virtual world or game world.

[1] Although, we have dealt with some of these issues in a prior version of the architecture, which was connected to the AGISim framework, a wrapper for the open-source game engine CrystalSpace

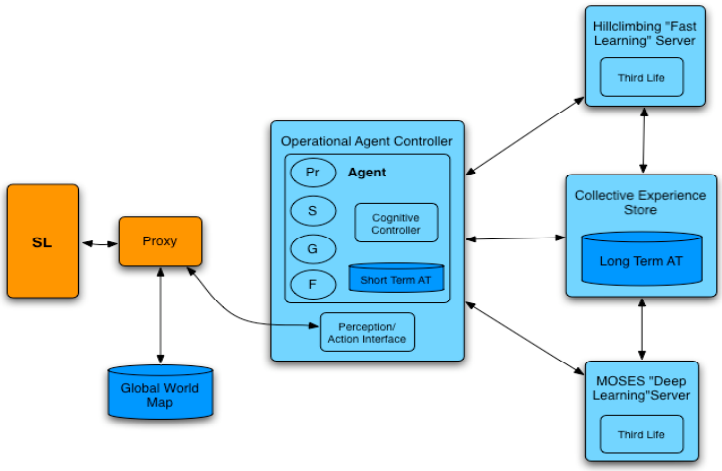

Figure 2. Initial Virtual Animal Brain software architecture

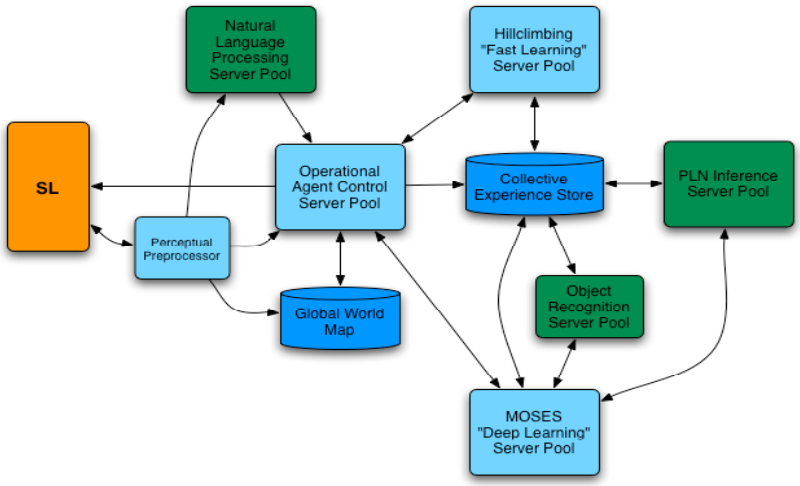

Figure 3. Next-Phase Virtual Animal Brain software architecture (tentatively intended for implementation in 2008)

2.1. Software Architecture for Virtual Animal Control

Figure 2 shows the initial version of the VAB architecture, which has currently been implemented. Figure 3 shows the next version of the architectre, which has been designed in detail and if all goes well will be implemented during 2008.

All components of VAB but the Perceptual Pre-processor and the Global World Map will be specialized instances of the Novamente Cognition Engine, an existing C++ software system described in [1-3] which contains numerous functional components and has been used within two commercial applications (the Biomind ArrayGenius product for gene expression data analysis [7]; and the INLINK product for interactive natural language knowledge entry, founded on the RelEx semantic analysis engine [8]) and also within an R&D system that controls a humanoid agent learning simple behaviors in a 3D simulation world [9].

The learning servers will be described in the following section, as learning is the main focus of this paper. In the remainder of this section we will describe the other aspects of the architecture, which form the context in which the learning occurs.

The Global World Map is in its initial incarnation basically a 2D navigation mesh for Second Life, where each point in space is labeled with information about the agents that reside there. Some markup is included in the map to indicate 3-dimensional features of the environment. Future versions will involve extension into a full 3D navigation mesh.

2.2. Perception and Action

The perceptual pre-processor is essentially a proxy that stands between SL and the rest of the architecture, translating the output of SL into an XML format that the VAB can understand.

The current VAB system receives perceptions at a fairly high level – for instance, it observes a list of nearby objects, with metadata attached to them, and information about the spatial coordinates they occupy. Regarding avatars, it receives information regarding the animations the avatars are running at a given time (a more useful list for some avatars than others, as in some cases avatars may run animations for which our AI agent doesn't know the semantics). Examples of the perceptions emanating from the Perceptual Pre-processor are things like (using a notation in which $ precedes variables):

- I am at world-coordinates $W
- Object with metadata $M is at world-coordinates $W
- Part of object with metadata $M is at world-coordinates $W
- Avatar with metadata $M is at world-coordinates $W
- Avatar with metadata $M is carrying out animation $A
- Statements in Petaverse, from the pet owner

There is also proxy code that translates the actions and action-sequences generated by the VAB into instructions SL can understand (such as "launch thus-and-thus

animation"). Due to the particularities of Second Life[2], the current VAB system carries out actions via executing pre-programmed high-level procedures, such as "move forward one step", "bend over forward" and so forth. Example action commands are:

- Move ($d , $s) :$d is a distance, $s is a speed
- Turn ($a, $S) : $a is an angle , $s is a speed
- Pitch ($a, $S) : turn vertically up/down... [for birds only]
- Jump ($d, $h, $s) : $h is a maximum height, at the center of the jump
- Say ($T), $T is text : for agents with linguistic capability, which is not enabled in the current version
- pick up($O) : $O is an object
- put down($O)

2.3. The Agent Controller

Next, in the agent control software component, each virtual animal is associated with its own "animal brain" software object, including among other things:

- An in-built (weighted) set of goals, and a set of "personality parameters" that guide various aspects of its behavior
- A package of basic information
    - Location, owner, current physical form, etc.
- A declarative memory
    - Contains e.g. associations between names of behaviors, and schemata (procedures) carrying out those behaviors
- A "procedural memory" containing procedures that may be useful to enact
- An "episodic memory" containing specific episodes the animal has been involved in, especially those that have led to reward from a teacher, or to achievement of one of the animal's goals
- An attention allocation process, determining which schemata the pet carries out at each point in time
    - This process will vary its behavior based on the pet's "personality" as specified by user-supplied parameters
- An "active procedural pool" containing procedures that are currently in the midst of being acted out

The declarative memory stores a reasonably long record of what all the pets have seen and done. Of course, when memory gets full, sufficiently old memories are saved to disk (or deleted, but it would be nice to have samples on disk for later mining)

One of the aspects of the system under current active development is the Collective Experience Store, which contains a process that scans the experience base and extracts interesting "unsupervised learning problems." E.g. "Fred likes to feed pets. Let's

[2] The current Second Life API does not provide direct access to the skeletons underlying the characters executed in the world.

figure out what causes him to feed one pet but not another, and place that knowledge in the minds of those pets whose experiences are most useful in doing this figuring out."

2.4. Architecture for Real-Time Action Selection

As Stan Franklin [10] has pointed out, ultimately intelligence has got to come down to action selection. An intelligent agent has got to decide what actions to take at what points in time – this is the way it goes about achieving its goals. In the VAB, action selection is controlled for each animal within the object allocated to that animal inside the agent control component.

As noted above, each animal has a set of goals, which are provided in advance. These are fairly basic things such as: don't be too hungry or too full, don't be too thirsty or too hydrated, seek social interaction, seek novelty, seek praise. Each of these goals has a certain numerical weight attached to it, and the relative weighting of the different goals in an individual animal constitutes an important part of that animal's "personality."

Each animal's brain also contains a declarative knowledge base, containing information the animal has learned during its lifetime – who is its owner, if it has an owner; where it has found various items at various points of time in the past; who has been friendly versus evil to it; and so forth. The declarative knowledge base is an AtomTable, to use NCE lingo; it contains weighted, typed nodes and links in the manner typically used within the Novamente Cognition Engine.

And each animal's brain also contains a procedural knowledge base, consisting of the set of behavior-generating procedures the animal has found useful in various contexts in the past. The declarative knowledge base contains relationships, in NCE node-and-link format, expressing the content

Procedure P, in context C has led to the achievement of goal G

and associated with appropriate weightings including probabilistic truth values. For example, in the mind of an animal serving as a pet, there might be a link

```
Implication <[.8,.95],.95>

   AND
```

```
        Inheritance current_smell food

        Evaluation current_location my_home

        Execution goto my_food_bowl

        Evaluation am_hungry


maintain_appropriate_fullness
```

expressing the fact that if the animal is hungry, and smells food, and is at home, then going to its food bowl may be a way of achieving its goal of maintaining appropriate fullness (not being too full or too hungry). In this example the actual procedure involved is a very simple one, simply the "goto" procedure that heads toward an object. (The <[.8,.95],.95> is an uncertain truth value, representing in the NCE's "indefinite probability" format, see [11]). In general, the calculation of such truth values is not a difficult matter; the hard part is identifying what are the right contexts to use in conjunction with each procedure and each goal. In most cases, though, it turns out that the right contexts for virtual animals are relatively simple ones; this will obviously be more of a problem when one turns to applying a similar approach to virtual humanoid agents or other agents with more complex lives.

How does action selection work, then? Each procedure in the animal's mind is assigned an importance level, at each point in time, based on the degree to which it's estimated to imply the currently important goals given the currently relevant context. And note that the context here includes the other procedures that are executing at that point in time. Some procedures may take a while to execute – so the animal's brain must also maintain a list of the procedures that are currently in the middle of running. Then, a procedure is selected for execution, with the probability of its selection being proportional to its urgency.

2.4.1. Future Developments in Action Selection

Eventually, to make highly intelligent virtual agents, it will be necessary to do advanced probabilistic logical inference (as carried out e.g. by the NCE's PLN inference engine) on the fly in the course of action selection, in order to calculate importances based on the current context in a sufficiently flexible way. From the point of view of a highly intelligent agent, each context is unique, and its relevant relationships to prior contexts must be assessed via a unique chain of reasoning. But from the point of view of controlling virtual animals, it seems acceptable to ignore these subtleties and make use of a fixed set of primitive contexts, allowing the system to look at combinations of contexts in this fixed set but not to interpret procedures more broadly in the context of more general, creatively-created contexts. The primitive contexts the current virtual animals can use include:

- physiological sensations such as am_hungry, am_tired, am_thirsty, etc.
- familiar locations such as am_home, am_outside, am_near_water, am_in_structure_with_ID_n
- social situations like alone_with_owner, near_friends, near_enemies, etc.
- indications of which current procedures are active

Combinations of these yield a variety of different contexts in which the utility of various procedures at leading to various goals can be assessed.

Another aspect of action selection not yet taken into account in our virtual animals is subgoaling. Right now they try to learn how to achieve their built-in goals, which may be weighted differently for
different animals based on personality. A more highly intelligent version of the system would use inference to create new goals serving as subgoals to the built-in goals, and then strive to achieve these subgoals as well. This leads to the necessity for a more complex system combining attention allocation and goals, as contained in the NCE design (and described in moderate detail in [1]).

2.5. Aspects of the VAB Architecture Scheduled for Future Implementation

Figure 3 shows some additional components intended for addition to the VAB, but not yet in place in the operational software.

One functionality we would like to add in future is object recognition – by which we refer to the capability of a pet to look at an object in the SL world and determine what kind of object it is (a vehicle, a car, a truck, a hat, a shirt, a vest, etc.). This may be achieved via deployment of the MOSES algorithm as a supervised categorization engine. This is actually a simpler use of MOSES than the "behavior learning" use that lies at the center of the current VAB version, but has deferred for logistic reasons, and to to reduce the complexity of the initial VAB. For the first version of the VAB, object recognition occurs only insofar as objects are labeled with appropriate metadata indicating their type.

Next, we would like to incorporate the full power of the NCE's Probabilistic Logic Networks reasoning engine [11] into the VAB, so as to enable more effective generalization within the Collective Experience Store. The current VAB utilizes some PLN inference rules within a simple control scheme, but this is not nearly as powerful as enabling general PLN backward and forward chaining inference on the animals' collective memory store. Ultimately, in fact, we would like to incorporate PLN into the real-time agent controller component of the sysetm as well, so as to enable more intelligent and contextually appropriate action selection; but this is a bigger job than incorporating PLN into the CES, because it requires real-time control of advanced PLN inference.

Finally, even though in general real non-human animals don't understand very much natural language, nevertheless it will be of considerable value to integrate a more robust NLP capability into the VUB in a later version. This is because the existing version of the Animal Command Language, while quite powerful in its capabilities, is still very brittle compared to natural languages. Users will need to phrase things in the exact right way or the animals won't understand them. This brittleness can be mitigated somewhat by coding synonyms and alternate syntactic patterns into the

system, but ultimately the only way to make the ACL easily usable by naive end-users is to incorporate some basic NLP parsing and semantic analysis. Of course, this also constitutes a step toward creating more linguistically ambitious animals as mentioned above, such as parrots that talk; and also toward creating humanoid avatars that communicate in English with a level of grounded understanding, rather than in the manner of chat-bots.

3. IRC Learning in the Virtual Animal Brain

Perhaps the best way to introduce the essential nature of the IRC teaching protocol is to give a brief snippet from a script that was created to guide the actual training of our Second Life virtual animals. This snippet involves only I and R; the C will be discussed afterwards.

This snippet demonstrates a teaching methodology that involves two avatars: Bob who is being the teacher, and Jill who is being an "imitation animal," showing the animal what to do by example.

1. *Bob wants to teach the dog Fido a trick. He calls his friend Jill over. "Jill, can you help me teach Fido a trick?"*
2. *Jill comes over. "How much will you pay me for it?*
3. *Bob gives her a kiss.*
4. *"All right," says Jill, "what do you want to teach him?"*
5. *"Let's start with fetching stuff," replies Bob.*
6. *So Bob and Jill start teaching Fido to fetch using the Pet language....*
7. *Bob says: "Fido, I'm going to teach you to play fetch with Jill."*
8. *Fido sits attentively, looking at Bob.*
9. *Bob says: "OK, I'm playing fetch now."*
10. *Bob picks up a stick from the ground and throws it. Jill runs to get the stick and brings it back to Bob.*
11. *Bob says: "I'm done fetching.*
12. *Bob says, "You try it."*
13. *Bob throws a stick. Fido runs to the stick, gets it, and brings it back.*
14. *Bob says "Good dog!"*
15. *Fido looks happy.*
16. *Bob says: "Ok, we're done with that game of fetch.*
17. *Bob says, "Now, let's try playing fetch again."*
18. *This time, Bob throws a stick in a different direction, where there's already a stick lying on the ground (call the other stick Stick 2).*
19. *Fido runs and retrieves Stick 2. As soon as he picks it up, Bob says "No." But Fido keeps on running and brings the stick back to Bob.*
20. *Bob says "No, that was wrong. That was the wrong stick. Stop trying!"*
21. *Jill says, "Furry little moron!"*
22. *Bob says to Jill, "Have some patience, will you? Let's try again."*
23. *Fido is slowly wandering around, sniffing the ground.*
24. *Bob says "Fido, stay." Fido returns near Bob and sits.*
25. *Bob throws Stick 2. Fido starts to get up and Bob repeats "Fido, stay."*
26. *Bob goes and picks up Stick 1, and walks back to his original position.*
27. *Bob says "Fido, I'm playing fetch with Jill again."*

28. *Bob throws the first stick in the direction of stick 2.*
29. *Jill goes and gets stick 1 and brings it back to Bob.*
30. *Bob says "I'm done playing fetch with Jill."*
31. *Bob says "Try playing fetch with me now." He throws stick 1 in another direction, where stick 3 and stick 4 are lying on the ground, along with some other junk.*
32. *Fido runs and gets stick 1 and brings it back.*
33. *Bob and Jill both jump up and down smiling and say "Good dog! Good dog, Fido!! Good dog!!"*
34. *Fido smiles and jumps up and licks Jill on the face.*
35. *Bob says, "Fido, we're done practicing fetch."*

The text directed by Bob to Fido is in a limited dialect of English we call the ACL or Animal Command Language (which takes several different forms with varying levels of linguistic sophistication, but this point can be bypassed here, as the focus of this paper is not computational linguistics). Line 7 initiates a formal training session, and Line 33 terminates this session. The training session is broken into "exemplar" intervals during which exemplars are being given, and "trial" intervals during which the animal is trying to imitate the exemplars, following which is receives reinforcement on its success or otherwise. For instance line 9 initiates the presentation of an exemplar interval, and line 11 indicates the termination of this interval. Line 12 indicates the beginning of a trial interval, and line 16 indicates the termination of this interval.

The above example of combined imitative/reinforcement learning involves two teachers, but, this is of course not the only way things can be done. Jill could be eliminated from the above teaching example. The result of this would be that, in figuring out how to imitate the exemplars, Fido would have to figure out which of Bob's actions were "teacher" actions and which were "simulated student" actions. This is not a particularly hard problem, but it's harder than the case where Jill carries out all the simulated-student actions. So in the case of teaching fetch with only one teacher avatar, on average, more reinforcement trials will be required.

3.1. Corrective Learning

Another interesting twist on the imitative/reinforcement teaching methodology described above is the use of explicit correctional instructions from the teacher to the animal. This is not shown in the above example but represents an important addition to the methodology show there. One good example of the use of corrections would be the problem of teaching would be teaching an animal to sit and wait until the teacher says "Get Up," using only a single teacher. Obviously, using two teachers, this is a much easier problem. Using only one teacher, it's still easy, but involves a little more subtlety, and becomes much more tractable when corrections are allowed.

One way that human dog owners teach their dogs this sort of behavior is as follows:

- Tell the dog "sit"
- tell the dog "stay"

- Whenever the dog tries to get up, tell him "no" or "sit", and then he sits down again
- eventually, tell the dog to "get up"

The real dog understands, in its own way, that the "no" and "sit" commands said after the "stay" command are meta-commands rather than part of the "stay" behavior.

In our virtual-pet case, this would be more like

- tell the dog "I'm teaching you to stay"
- Tell the dog "sit"
- Whenever the dog tries to get up, tell him "no" or "sit", and then he sits down again
- eventually, tell the dog to "get up"
- tell the dog "I'm done teaching you to stay"

The easy way to do this is to give the Animal Command Language an explicit META flag. In this case, the teaching would look like

- tell the dog "I'm teaching you to stay"
- Tell the dog "META: sit"
- Whenever the dog tries to get up, tell him "META: no" or "META:sit", and then he sits down again
- eventually, tell the dog to "get up"
- tell the dog "I'm done teaching you to stay"--

Even without the META tag, this behavior is learnable via our learning algorithms within a modest number of reinforcement trials. But this well illustrates the give-and-take relationship between the sophistication of the teaching methodology and the number of reinforcement trials required. In many cases, the best way to reduce the number of reinforcement trials required to learn a behavior is not to increase the sophistication of the learning algorithm, but rather to increase the information provided during the instruction process. No matter how advanced the learning algorithm, if the teaching methodology only gives a small amount of information, it's going to take a bunch of reinforcement trials to go through the search space and find one of the right procedures satisfying the teacher's desires. One of the differences between the real-world learning that an animal or human child (or adult) experiences, and the learning "experienced" by standard machine-learning algorithms, is the richness and diversity of information that the real world teaching environment provides, beyond simple reinforcement signals. Virtual worlds provide a natural venue in which to experiment with providing this sort of richer feedback to AI learning systems, which is one among the many reasons why we feel that virtual worlds are an excellent venue for experimentation with and education of early-stage AGI systems.

4. The Cognitive Infrastructure Supporting IRC Learning in the Virtual Animal Brain

We now turn to the two pools of "learning servers" described in the above architecture diagram. These architectural components exist to carry out supervised or unsupervised learning, in order to learn new procedures for governing agent behavior, which may then be placed in the Agent Control Server Pool and associated there with the proper animal or set of animals. They constitute a specific cognitive infrastructure implementing the IRC learning methodology in the virtual-animal context, with some extensibility beyond this context as well.

In the VAB, we have chosen to deploy two different learning algorithms, with different strengths and weaknesses. We have implemented a variety of hillclimbing, which is a fast learning algorithm but may fail on harder problems (in the sense of requiring an unacceptably large number of reinforcement trials). And we are currently in the midst of integrating MOSES, a sophisticated probabilistic evolutionary learning algorithm [12], as an alternative. Compared to hillclimbing, MOSES is much smarter but slower, and may take a few minutes to solve a problem. The two algorithms (as implemented for the VAB) share the same knowledge representation (a certain kind of C++ "program tree" used for representing procedures) and some other software components (e.g. normalization rules for placing procedures in an appropriate hierarchical normal form, as described in [12]).

The big challenge involved in designing the VAB system, AI-wise, is that these learning algorithms, used in a straightforward way with feedback from a human-controlled avatar as the fitness function, would need an excessive number of reinforcement trials to learn relatively simple behaviors. This would bore the human beings involved with teaching the animals. This is not a flaw of the particular learning algorithms being proposed, but is a generic problem that would exist with any AI algorithms. To choose an appropriate behavior out of the space of all possible behaviors satisfying reasonable constraints, requires more bits of information that is contained in a handful of reinforcement trials.

Most "animal training" games (e.g. Nintendogs may be considered as a reference case) work around this "hard problem" by not allowing teaching of novel behaviors. Instead, a behavior list is made up front by the game designers. The animals have preprogrammed procedures for carrying out the behaviors on the list. As training proceeds they make fewer errors, till after enough training they converge "miraculously" on the pre-programmed plan

This approach only works, however, if all the behaviors the animals will ever learn have been planned and scripted in advance.

The first key to making learning of non-pre-programmed behaviors work, without an excessive number of reinforcement trials, is in "fitness estimation" -- code that guesses the fitness of a candidate procedure at fulfilling the teacher's definition of a certain behavior, without actually having to try out the procedure and see how it works. This is where the I part of IRC learning comes in.

At an early stage in designing the VAB application, we realized it would be best if the animals were instructed via a methodology where the same behaviors are defined by the teacher both by demonstration *and* by reinforcement signals. The ACL language

described above is designed to encourage this. Learning based on reinforcement signals only can also be handled, but learning will be slower.

In evolutionary programming lingo, we have

- Procedures = genotypes
- Demonstrated exemplars, and behaviors generated via procedures = phenotypes
- Reinforcement signals from pet owner = fitness

One method of imitation-based fitness estimation used in the VAB involves an internal simulation world called Third Life (TL). TL can be visualized using a simple testing UI, but in the normal course of operations it doesn't require a user interface; it is an internal simulation world, which allows the VAB to experiment and see what a certain procedure would be likely to do if enacted in the SL virtual world. Of course, the accuracy of this kind of simulation depends on the nature of the procedure. For procedures that solely involve moving around and interacting with inanimate objects, it can be very effective. For procedures involving interaction with human-controlled avatars, other animals, or other complex objects, it may be unreliable – and making it even moderately reliable would require
significant work that has not yet been done, in terms of endowing TL with realistic simulations of other agents and their internal motivational structures and so forth. Ultimately, for TL to work well would require an agent with a sophisticated Theory of Mind in the developmental-psychology sense (see [13] for a treatment of Piagetan developmental psychology in an AGI context). But short of this, TL has nonetheless proved useful for estimating the fitness of simple behavioral procedures.

When a procedure is enacted in TL, this produces an object called a "behavior description" (BD), which is represented in the NCE's generic Atomspace (weighted labeled hypergraph) knowledge representation format. The BD generated by the procedure is then compared with the BD's corresponding to the "exemplar" behaviors that the teacher has generated, and that the student is trying to emulate. Similarities are calculated, which is a fairly subtle matter that involves some heuristic inferences. An estimate of the likelihood that the procedure, if executed in SL, will generate a behavior adequately similar to the exemplar behaviors.

Furthermore, this process of estimation may be extended to make use of the animal's long-term memory as collected in the CES component. Suppose a procedure P is being evaluated in the context of exemplar-set E. Then

- The experience base is mined for pairs (P', E') that are similar to (P,E)
- The fitness of these pairs (P', E') is gathered from the experience base
- An estimate of the fitness of (P,E) is then formed

Of course, if a behavior description corresponding to P has been generated via TL, this may also be used in the stimilarity matching against long-term memory. The tricky part here, of course, is the similarity measurement itself, which can be handled via simple heuristics, but if taken sufficiently seriously becomes a complex problem of uncertain inference.

One thing to note here is that, although learning is done by each animal individually, this learning is subtly guided by collective knowledge within the fitness estimation process. Internally, we have a "borg mind" with multiple animal bodies,

and an architecture designed to ensure the maintenance of unique personalities on the part of the individual animals in spite of the collective knowledge and learning underneath.

At time of writing, we have just begun to experiment with the learning system as described above, and are using it to learn simple behaviors such as playing fetch, basic soccer skills, doing specific dances as demonstrated by the teacher, and so forth. We have not yet done enough experimentation to get a solid feel for the limitations of the methodology as currently implemented.

Note also that, going forwards, there is a possibility to use NCE's PLN inference component to allow generalization of learned behaviors. For instance, with inference deployed appropriately, a pet that had learned how to play tag would afterwards have a relatively easy time learning to play "freeze tag." A pet that had learned how to hunt for Easter eggs would have a relatively easy time learning to play hide-and-seek. Now, even the initial VAB will have some level of generalization ability in place, due to the use of the Collective Experience Store for fitness estimation. However, explicit use of inference, as is intended for later versions, will allow much more rapid and far-reaching inference capabilities.

4.1. Introducing Corrective Learning

Finally, how may corrections be utilized in the learning process we have described? Obviously, the corrected behavior description gets added into the knowledge base as an additional exemplar. And, the fact of the correction acts as a partial reinforcement (up until the time of the correction, what the animal was doing was correct). But beyond this, what's necessary is to propagate the correction backward from the BD level to the procedure level. For instance, if the animal is supposed to be staying in one place, and it starts to get up but is corrected by the teacher (who says "sit" or physically pushes the animal back down), then the part of the behavior-generating procedure that directly generated the "sit" command needs to be "punished." How difficult this is to do, depends on how complex the procedure is. It may be as simple as providing a negative reinforcement to a specific "program tree node" within the procedure, thus disincentivizing future procedures generated by the procedure learning algorithm from containing this node. Or it may be more complex, requiring the solution of an inference problem of the form "Find a procedure P' that is as similar as possible to procedure P, but that does not generate
the corrected behavior, but rather generates the behavior that the teacher wanted instead." This sort of "working backwards from the behavior description to the procedure" is never going to be perfect except in extremely simple cases, but it is an important part of learning. We have not yet experimented with this extensively in our virtual animals, but plan to do so as the project proceeds.

There is also an interesting variant of correction in which the agent's own memory serves implicitly as the teacher. That is, if a procedure generates a behavior that seems wrong based on the history of successful behavior descriptions for similar exemplars, then the system may suppress that particular behavior or replace it with another one that seems more appropriate – inference based on history thus serving the role of a correcting teacher.

4.2. Applying A Similar IRC Methodology to Spontaneous Learning

We have described the IRC teaching/learning methodology in the context of learning from a teacher – but in fact a similar approach can be utilized for purely unsupervised learning. In that case, the animal's intrinsic goal system acts implicitly as a teacher. Experimentation with this sort of learning is on our list of virtual-animal R&D goals for 2008.

For instance, suppose the animal wants to learn how to better get itself fed. In this case,

- Exemplars are provided by instances in the animal's history when it has successfully gotten itself fed
- Reinforcement is provided by, when it is executing a certain procedure, whether or not it actually gets itself fed or not
- Correction as such doesn't apply, but implicit correction may be used via deploying history-based inference. If a procedure generates a behavior that seems wrong based on the history of successful behavior descriptions for the goal of getting fed, then the system may suppress that particular behavior.

The only real added complexity here lies in identifying the exemplars. In surveying its own history, the animal must look at each previous instance in which it got fed (or sme sample thereof), and for each one recollect the series of N actions that it carried out prior to getting fed. It then must figure out how to set N – i.e. which of the actions prior to getting fed were part of the behavior that led up to getting fed, and which were just other things the animal happened to be doing a while before getting fed. To the extent that this exemplar mining problem can be solved adequately, innate-goal-directed spontaneous learning becomes closely analogous to teacher-driven learning as we've described it. Or in other words: Experience, as is well known, can serve as a very effective teacher.

5. Conclusion and Next Steps

We have described some recent work involving the use of the IRC teaching/learning methodology to instruct virtual animals in Second Life. This constitutes a significant step beyond what is commonly done in virtual worlds and games regarding virtual-animal instruction; and a significant conceptual step beyond the pure reinforcement learning methodology that is commonly studied in the AI field. We have conducted some simple experiments using the IRC methodology in our Virtual Animal Brain already, but we still have a lot to learn about the best ways to pragmatically combine reinforcement, imitative and corrective learning in a virtual-world context. Along these lines, we have formulated a detailed roadmap for further research and development in the domain of virtual animal instruction. This includes a number of items mentioned above: object recognition, extension of the integrative methodology described above to spontaneous learning, and further integration of PLN inference to allow more sophisticated history-based fitness estimation and context-based action selection. However, in this Conclusion, rather than reviewing this short-to-medium-term roadmap in more depth, we will take the opportunity to step back a bit and review the

connections between this virtual-animal work and the larger AGI project of which it forms a component, and describe some of our medium-to-long-term plans for using IRC to enable a transition beyond nonlinguistic virtual animals.

At the start of the paper we noted four critical aspects to AGI: knowledge representation, learning, cognitive architecture and teaching methodology. Our main theme here has been teaching methodology, but we have ventured into the other three categories considerably as well, as necessary to describe the implementation and significance of the teaching methodology presented. The architecture, representations, and learning algorithms described here constitute a fairly small subset of the ones currently embodied in the overall NCE codebase, and the teaching methodology presented here is correspondingly somewhat limited. But the ideas given here have been presented, not only for their innate interest, but as initial concretizations of a larger vision which as yet exists mainly in the form of software designs, theoretical analyses and limited-scale software prototypes, but which we are doing our best to move toward practical realization.

In the remainder of this Conclusion, we will outline some of the steps by which we feel the (admittedly substantial) gap between simple virtual animals and human-level AGI may be filled: a path that leads from nonlinguistic virtual-animal learning system, to parrots that talk, to virtual babies that mature, and ultimately to adult virtual humanoid agents that communicate gesturally, pragmatically and linguistically in a manner grounded in their virtual-world social, perceptual, motoric and cognitive experiences. Obviously, there are many challenges to be faced along this path, but we believe that it is a considerably more viable pragmatic route to powerful AGI than any other that is currently available.

The main theme we wish to focus on in discussing the path forward is *language learning*. We have argued above that the combination of reinforcement, imitation and correction is a complete teaching methodology for cases where there is no linguistic communication between teacher and agent. It is obvious that the potential for high-bandwidth instruction is far greater if one introduces the possibility for linguistic communication. However, the current state of the art in computational linguistics does not support robust, intelligent automated communication. Rather, the best current system for automated dialogue are depressingly similar in character to ELIZA and other simplistic chat bots from decades ago. There has been tremendous progress in information retrieval, semantic relationship extraction and other areas [14], but this has not translated very effectively into progress in automated dialogue. The reason for this, we suggest, is that realistic dialogue is highly centered on experiential grounding. Carrying out dialogue requires a system to understand the contextual meanings of the linguistic terms it is using. Perhaps the hand-coded and statistical rules used in current NLP systems will turn out to be useful for an AGI system; but if so, it will be via embedding them in a framework that allows them to be adaptively deployed based on context. And this requires experiential language learning. We suggest that the IRC framework described here, as deployed in virtual worlds like Second Life, provides an ideal platform for experiential language learning.

Along these lines, part of our tentative plan for future R&D is to integrate, into our current virtual animal infrastructure, an improved version of the language engine called RelEx briefly described in [8]. Of course a virtual animal with a language engine could be concretized many different ways but – inspired in part by Irene Pepperberg's [15] groundbreaking work teaching an actual parrot complex things like grammar and arithmetic -- the specific scenario we've considered most seriously is a virtual talking

parrot. Potentially, this may provide a means to facilitate robust language learning on the part of virtually embodied agents, and lead to an experientially-trained AGI language facility that can then be used to power other sorts of agents such as virtual babies, and ultimately virtual adult-human avatars that can communicate with experientially-grounded savvy rather than in the manner of chat-bots.

Imagine millions of talking parrots spread across different online virtual worlds — all communicating in simple English. Each parrot has its own local memories, its own individual knowledge and habits and likes and dislikes — but there's also a common knowledge-base underlying all the parrots, which includes a common knowledge of English.

Next, suppose that an adaptive language learning algorithm is set up (using for instance the MOSES and PLN algorithms embedded in the Novamente Cognition Engine), so that the parrot-collective may continually improve its language understanding based on interactions with users according to the IRC methodology. If things go well, then the parrots will get smarter and smarter at using language, as time goes on – and will eventually, of course, graduate to a combination of IRC and linguistic learning.

Along related lines, Michael Tomasello [16], in his excellent book *Constructing a Language*, has given a very clear summary of the value of social interaction and embodiment for language learning in human children. And while he doesn't phrase it in these terms, the picture he portrays includes central roles for reinforcement, imitative and corrective learning. Imitative learning is obvious: so much of embodied language learning has to do with the learner copying what it has heard other say in similar contexts. Corrective learning occurs every time a parent rephrases something for a child. And for a virtual parrot, the test of whether it has used English correctly, in a given instance, will come down to whether its human friends have rewarded it, and whether it has gotten what it wanted. If a parrot asks for food incoherently, it's less likely to get food — and since the virtual parrots
will be programmed to want food, they will have motivation to learn to speak correctly. If a parrot interprets a human-controlled avatar's request "Fetch my hat please" incorrectly, then it won't get positive feedback from the avatar — and it will be programmed to want positive feedback.

The intersection between linguistic experience and embodied perceptual/active experience is one thing that makes the notion of a virtual talking parrot very fundamentally different from the "chatbots" on the Internet today. The other major difference, of course, is the presence of learning – chatbots as they currently exist rely almost entirely on hard-coded lists of expert rules. But the interest of many humans in interacting with chatbots suggests that virtual talking parrots or similar devices would be likely to meet with a large and enthusiastic audience.

Yes, humans interacting with parrots in virtual worlds can be expected to try to teach the parrots ridiculous things, obscene things, and so forth. But still, when it comes down to it, even pranksters and jokesters will have more fun with a parrot that can communicate better, and will prefer a parrot whose statements are comprehensible.

And of course parrots are not the end of the story. Once the collective wisdom of throngs of human teachers has induced powerful language understanding in the collective bird-brain, this language understanding (and the commonsense understanding coming along with it) will be useful for many, many other purposes as well. Humanoid avatars — both human-baby avatars that may serve as more rewarding virtual companions than parrots or other virtual animals; and language-savvy human-adult

avatars serving various useful and entertaining functions in online virtual worlds and games. Once AI's have learned enough that they can flexibly and adaptively explore online virtual worlds and gather information from human-controlled avatars according to their own goals using their linguistic facilities, it's easy to envision dramatic acceleration in their growth and understanding.

A baby AI has numerous disadvantages compared to a baby human being: it lacks the intricate set of inductive biases built into the human brain, and it also lacks a set of teachers with a similar form and psyche to it … and for that matter, it lacks a really rich body and world. However, the presence of thousands to millions of teachers constitutes a large advantage for the AI over human babies. And a flexible AGI framework will be able to effectively exploit this advantage. If nonlinguistic learning mechanisms like the ones we've described here, utilized in a virtually-embodied context, can go beyond enabling interestingly trainable virtual animals and catalyze the process of language learning – then, within a few years time, we may find ourselves significantly further along the path to AGI than most observers of the field currently expect.

References

[1] Goertzel, Ben (2007). Virtual Easter Egg Hunting: A Thought-Experiment in Embodied Social Learning, Cognitive Process Integration, and the Dynamic Emergence of the Self. In Advances in artificial general intelligence, Ed. by Ben Goertzel and Pei Wang:36-54. Amsterdam: IOS Press.

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

[3] Goertzel, Ben (2006). The Hidden Pattern. BrownWalker Press

[4] Crevier, Daniel (1993), AI: The Tumultuous Search for Artificial Intelligence, New York, NY: Basic Books

[5] Brooks, Rodney (1999). Cambrian Intelligence. MIT Press.

[6] Winograd, Terry (1972) . Understanding Natural Language. San Diego: Academic.

[7] Goertzel, Ben, Cassio Pennachin, Lucio Coelho, Leonardo Shikida, Murilo Queiroz (2007). Biomind ArrayGenius and GeneGenius: Web Services Offering Microarray and SNP Data Analysis via Novel Machine Learning Methods In Proceedings of IAAI 2007, Vancover CA, July 2007

[8] Goertzel, Ben, Hugo Pinto, Ari Heljakka, Michael Ross, Izabela Goertzel, Cassio Pennachin. Using Dependency Parsing and Probabilistic Inference to Extract Gene/Protein Interactions Implicit in the Combination of Multiple Biomedical Research Abstracts, Proceedings of BioNLP-2006 Workshop at ACL-2006, New York

[9] Heljakka, Ari, Ben Goertzel, Welter Silva, Izabela Goertzel and Cassio Pennachin (2006). Reinforcement Learning of Simple Behaviors in a Simulation World Using Probabilistic Logic, in Advances in Artificial General Intelligence, IOS Press.

[10] Franklin, Stan (1995). Artificial Minds. MIT Press.

[11] Ikle', Matt, Ben Goertzel, Izabela Goertzel and Ari Heljakka (2007). Indefinite Probabilities for General Intelligence, in Advances in Artificial General Intelligence, IOS Press.

[12] Looks, Moshe (2006). Competent Program Evolution. PhD Thesis, Department of Computer Science, Washington University, St. Louis

[13] Goertzel, Ben and Stephan Bugaj (2006). Stages of Cognitive Development in Uncertain-Logic-Based AI Systems. In Advances in artificial general intelligence, Ed. by Ben Goertzel and Pei Wang:36-54. Amsterdam: IOS Press.

[14] Manning, Christopher and Heinrich Scheutze (1999). Foundations of Statistical Natural Language Processing. MIT Press.

[15] Pepperberg, Irene (2000). The Alex Studies. Harvard University Press.

[16] Tomasello, Michael (2003). Constructing a A Language. Harvard University Press.