Cognitive Synergy between Procedural and Declarative Learning in the Control of Animated and Robotic Agents Using the OpenCogPrime AGI Architecture

B. Goertzel[1,3], **J. Pitt**[2], **J. Wigmore**[2], **N. Geisweiller**[1], **Z. Cai**[2,3], **R. Lian**[3], **D. Huang**[2,3], **G. Yu**[2]
[1] Novamente LLC, Rockville MD, USA, [2] M-Lab, School of Design, Hong Kong Polytechnic University,
[3] Fujian Key Lab for Brain-Like Intelligent Systems, Cog. Sci. Dept., Xiamen University, China

Abstract

The hypothesis is presented that "cognitive synergy" – proactive and mutually-assistive feedback between different cognitive processes associated with different types of memory – may serve as a foundation for advanced artificial general intelligence. A specific AI architecture founded on this idea, OpenCogPrime, is described, in the context of its application to control virtual agents and robots. The manifestations of cognitive synergy in OpenCogPrime's procedural and declarative learning algorithms are discussed in some detail.

Supposing one agrees that a broadly "integrative" approach is an appropriate way to achieve advanced AI functionality. One must then specify what "integrative" really means. At one extreme, it could mean merely connecting together different software components that solve problems via highly independent internal processes, in such a way that they occasionally pass each other problems to solve and receive each others' answers. At the other extreme, it could mean binding various components together in tight feedback loops so that the internal dynamics of each component can be effectively understood only in the internal context of the others. While many approaches to integration may be workable, here we are specifically concerned with AI of the "very tightly integrated" type.

Specifically, we describe some theoretical elaborations and practical applications of the concept of **cognitive synergy** introduced in (Goertzel 2009a), defined roughly as: the fitting-together of different intelligent components into an appropriate cognitive architecture, in such a way that the components proactively and mutually assist each other's internal operations, regularly coming to each others rescue "mid thought process" in situations where ineffective cognition is observed or anticipated. We describe some aspects of a research program designed to explore (and leverage the potential truth of) the hypothesis that **the cognitive synergy ensuing from integrating multiple symbolic and subsymbolic learning and memory components in an appropriate cognitive architecture and environment, can ultimately produce artificial general intelligence (AGI) at the human level or beyond**. For discussion of what is meant by the term AGI, see (Goertzel and Pennachin 2005); in brief, it is intended to refer to the broadly human-like capability to solve problems in a variety of domains without domain-specific training, including potentially domains unknown to the system or its programmers at the time of system creation, and the ability to generalize knowledge from a set of domains to dramatically different domains.

This approach fits neatly with what is known about human neurobiology. The human brain is an integration of an assemblage of diverse structures and dynamics, built using common components and arranged according to a sensible cognitive architecture. Its algorithms and structures have been honed by evolution to work closely together – they are very tightly inter-adapted, in the same way that the different organs of the body are adapted to work together. Due their close interoperation they give rise to the systemic behaviors that characterize human-like general intelligence.

At the broadest level, there are four primary challenges in constructing a cognitive synergy based AGI system:

1. choosing an **overall cognitive architecture** that possesses adequate richness and flexibility for the task of achieving advanced cognition
2. choosing **appropriate AI algorithms and data structures** to fulfill each of the functions identified in the cognitive architecture (e.g. visual perception, audition, episodic memory, language generation, analogy,...)
3. ensuring that these algorithms and structures, within the chosen cognitive architecture, are able to cooperate in such a way as to provide appropriate **coordinated, synergetic intelligent behavior** (critical since advanced cognition is an integrated functional response to the world, not a loosely coupled collection of capabilities)
4. embedding one's system in an environment that provides **sufficiently rich stimuli and interactions** to enable the system to use this cooperation to ongoingly create an intelligent internal world-model and self-model

We describe here an AGI-oriented architecture called **OpenCogPrime** (OCP), based on the open-source OpenCog project `http://opencog.org`, which is fundamentally reliant on the cognitive synergy concept, and which is currently being used (in research projects) to control animated agents in video game worlds, and (via the OpenCogBot

(OCB) extension (Goertzel 2010)) humanoid (Nao) robots in a robot lab. The various subsystems of OpenCogBot involve sophisticated methods for visual and auditory pattern inference, language comprehension and generation, action planning, commonsense uncertain reasoning, and concept creation; and most of these capabilities have been evaluated in prior applications of narrower scope.

The medium-term goal of the OCP/OCB project is to create systems that can function broadly comparably to young human children in virtual and robotic preschool contexts (Goertzel and Bugaj 2009). Longer-term, the project is explicitly aimed at the grand old goals of the AI field – full human-adult-level intelligence, and ultimately beyond. The current functionality of the system is not particularly close to these ambitious goals, however, the architecture has been worked out with these goals in mind, both in the broad outline and in the algorithmic and implementation details.

OpenCogBot is a large and complex system whose careful description occupies a lengthy volume (Goertzel, Pennachin, and Geisweiller 2010). Here, after describing the basic concepts of the architecture, we focus on describing the role cognitive synergy plays in two aspects of the architecture in particular, the declarative (probabilistic logic based) learning component and the procedural (probabilistic program evolution based) learning component. While only a fraction of the whole story, this does give some flavor of how cognitive synergy manifests itself in the concrete AGI design. We emphasize the implications of this cognitive synergy for the system's performance in virtual and robotic agent control.

OpenCogPrime and OpenCogBot

Conceptually founded on the "patternist" systems theory of intelligence outlined in (Goertzel 2006), OpenCogPrime combines multiple AI paradigms such as uncertain logic, computational linguistics, evolutionary program learning and connectionist attention allocation in a unified architecture. Cognitive processes embodying these different paradigms interoperate together on a common neural-symbolic knowledge store called the Atomspace. The interaction of these processes is designed to encourage the self-organizing emergence of high-level network structures in the Atomspace, including superposed hierarchical and heterarchical knowledge networks, and a self-model network enabling meta-knowledge and meta-learning.

The high-level architecture of OCP – as shown in the broader OpenCogBot architecture diagram in Figure 1 – involves the use of multiple cognitive processes associated with multiple types of memory to enable an intelligent agent to execute the procedures that it believes have the best probability of working toward its goals in its current context. In a preschool context, for example, the top-level goals are simple things such as pleasing the teacher, learning new information and skills, and protecting the agent's body.

What OCP does not handle is low-level perception and action; OpenCogBot surmounts this problem via integrating OCP with a hierarchical temporal memory system, DeSTIN (Arel, Rose, and Coop 2009).

Memory Types in OpenOCP OCP's memory types are the declarative, procedural, sensory, and episodic memory types that are widely discussed in cognitive neuroscience (Tulving and Craik 2005), plus attentional memory for allocating system resources generically, and intentional memory for allocating system resources in a goal-directed way. Table 1 overviews these memory types, giving key references and indicating the corresponding cognitive processes, and which of the generic patternist cognitive dynamics each cognitive process corresponds to (pattern creation, association, etc.).

The essence of the OCP design lies in the way the structures and processes associated with each type of memory are designed to work together in a closely coupled way, the operative hypothesis being that this will yield cooperative intelligence going beyond what could be achieved by an architecture merely containing the same structures and processes in separate "black boxes."

The inter-cognitive-process interactions in OpenCog are designed so that

- conversion between different types of memory is possible, though sometimes computationally costly (e.g. an item of declarative knowledge may with some effort be interpreted procedurally or episodically, etc.)
- when a learning process concerned centrally with one type of memory encounters a situation where it learns very slowly, it can often resolve the issue by converting some of the relevant knowledge into a different type of memory: i.e. **cognitive synergy**

Goal-Oriented Dynamics in OpenOCP OCP's dynamics has both goal-oriented and "spontaneous" aspects; here for simplicity's sake we will focus on the goal-oriented ones. The basic goal-oriented dynamic of the OCP system, within which the various types of memory are utilized, is driven by "cognitive schematics", which take the form

$$Context \wedge Procedure \rightarrow Goal < p >$$

(summarized $C \wedge P \rightarrow G$). Semi-formally, this implication may interpreted to mean: "If the context C appears to hold currently, then if I enact the procedure P, I can expect to achieve the goal G with certainty p." Cognitive synergy means that the learning processes corresponding to the different types of memory actively cooperate in figuring out what procedures will achieve the system's goals in the relevant contexts within its environment.

OCP's cognitive schematic is significantly similar to production rules in classical architectures like SOAR and ACT-R; however, there are significant differences which are important to OCP's functionality. Unlike with classical production rules systems, uncertainty is core to OCP's knowledge representation, and each OCP cognitive schematic is labeled with an uncertain truth value, which is critical to its utilization by OCP's cognitive processes. Also, in OCP, cognitive schematics may be incomplete, missing one or two of the terms, which may then be filled in by various cognitive processes. A stronger similarity is to MicroPsi's triplets.

Finally, the biggest difference between OCPs cognitive schematics and production rules or other similar constructs,

Memory Type	Specific Cognitive Processes	General Cognitive Functions
Declarative	Probabilistic Logic Networks (PLN) (Goertzel et al. 2008); concept blending (Fauconnier and Turner 2002)	pattern creation
Procedural	MOSES (a novel probabilistic evolutionary program learning algorithm) (Looks 2006)	pattern creation
Episodic	internal simulation engine (Goertzel and Et Al 2008)	association, pattern creation
Attentional	Economic Attention Networks (ECAN) (Goertzel et al. 2010)	association, credit assignment
Intentional	probabilistic goal hierarchy refined by PLN and ECAN, structured according to MicroPsi (Bach 2009)	credit assignment, pattern creation
Sensory	In OpenCogBot, this will be supplied by the DeSTIN component	association, attention allocation, pattern creation, credit assignment

Table 1: Memory Types and Cognitive Processes in OpenCog Prime. The third column indicates the general cognitive function that each specific cognitive process carries out, according to the patternist theory of cognition.

is that in OCP this level of knowledge representation is not the only important one. CLARION uses production rules for explicit knowledge representation and then uses a totally separate subsymbolic knowledge store for implicit knowledge. In OCP both explicit and implicit knowledge are stored in the same graph of nodes and links, with explicit knowledge stored in probabilistic logic based nodes and links such as cognitive schematics; and implicit knowledge stored in patterns of activity among these same nodes and links, defined via the activity of the "importance" values associated with nodes and links and propagated by the ECAN attention allocation process.

The meaning of a cognitive schematic in OCP is hence not entirely encapsulated in its explicit logical form, but resides largely in the activity patterns that ECAN causes its activation or exploration to give rise to. And this fact is important because the synergetic interactions of system components are in large part modulated by ECAN activity. Without the real-time combination of explicit and implicit knowledge in the system's knowledge graph, the synergetic interaction of different cognitive processes would not work so smoothly.

Current and Prior Applications of OpenCog OpenCog has been used for commercial applications in the area of natural language processing and data mining; e.g. see (Goertzel et al. 2006) where OpenCog's PLN reasoning and RelEx language processing are combined to do automated biological hypothesis generation based on information gathered from PubMed abstracts. Most relevantly to the present proposal, has also been used to control virtual agents in virtual worlds (Goertzel and Et Al 2008), using an OpenCog variant called the OpenPetBrain (see `http://novamente.net/example` for some videos of these virtual dogs in action).

While the OpenCog virtual dogs do not display intelligence closely comparable to that of real dogs (or human children), they do demonstrate a variety of interesting and relevant functionalities including learning new behaviors based on imitation and reinforcement; responding to natural language commands and questions, with appropriate actions and natural language replies; and spontaneous exploration of their world, remembering their experiences and using them to bias future learning and linguistic interaction. These are simpler versions of capabilities we are working to demonstrate with the OpenCogBot system.

Integrating OpenOCP with Hierarchical Temporal Memory OpenOCP is designed to handle most of the types of knowledge important for human like intelligence, in a manner manifesting cognitive synergy, but in its current form it doesn't deal with low-level sensorimotor knowledge. It could be extended to handle such knowledge in various ways, but in the OpenCogBot architecture we have chosen a different approach: hybridizing OCP with another sort of cognitive architecture, hierarchical temporal memory, that is specifically oriented toward sensorimotor learning, and has already been extensively tested in the perception domain.

The DeSTIN architecture (Arel, Rose, and Coop 2009) comprises three interlinked hierarchies:

- a deep spatiotemporal inference network carrying out perception and world-modeling
- a similarly architected critic network that provides feedback on the inference network's performance
- an action network that controls actuators based on the activity in inference network.

The nodes in these networks perform probabilistic pattern recognition according to algorithms; and the nodes in each of the networks may receive states of nodes in the other networks as inputs, providing rich interconnectivity and synergetic dynamics.

The basic logic of the integration of DeSTIN with OCP is depicted in Figure 1 but of course the essence of the integration lies in the dynamics, not the structure.

Cognitive Synergy for Procedural and Declarative Learning

We now present more algorithmic detail regarding the operation and synergetic interaction of OCP's two most so-

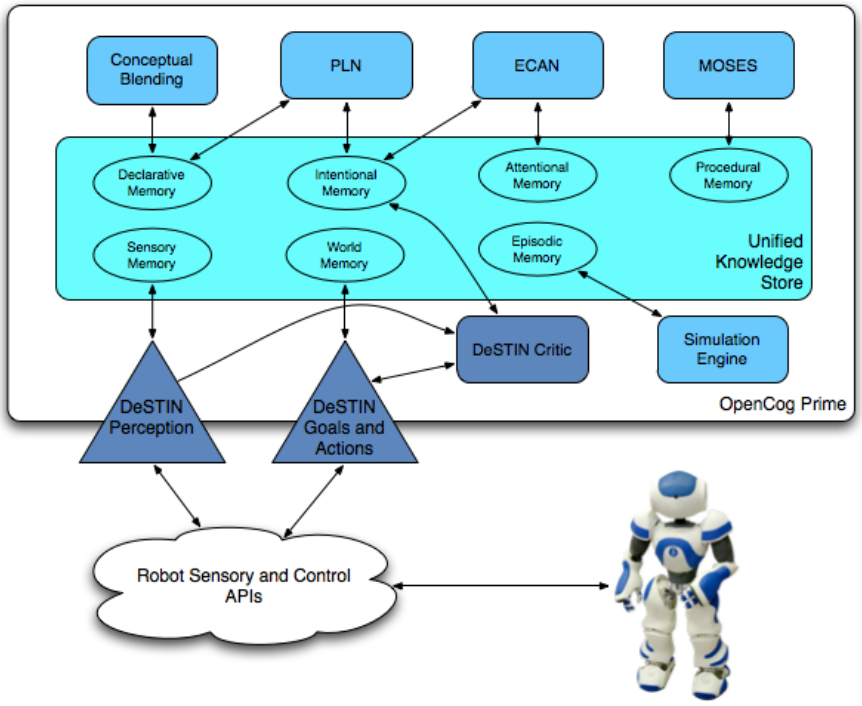

Figure 1: High-Level OpenCogBot Architecture Diagram

phisticated components: the MOSES procedure learning algorithm, and the PLN uncertain inference framework. The ideas discussed here play a significant role in our current work using OCP to control virtual agents, but for space reasons we will not give details here, only the broad ideas.

Cognitive Synergy in MOSES . MOSES, OCP's primary algorithm for learning procedural knowledge, has been tested on a variety of application problems including standard GP test problems, virtual agent control, biological data analysis and text classification (Looks 2006). It represents procedures internally as program trees. Each node in a MOSES program tree is supplied with a "knob," comprising a set of values that may potentially be chosen to replace the data item or operator at that node. So e.g. a node containing the number 7 may be supplied with a knob that can take on any integer value. A node containing a while loop may be supplied with a knob that can take on various possible control flow operators including conditionals or the identity. A node containing a procedure representing a particular robot movement, may be supplied with a knob that can take on values corresponding to multiple possible movements. The metaphor is that MOSES learning covers both "knob twiddling" (setting the values of knobs) and "knob creation."

MOSES is invoked within OCP in a number of ways, but most commonly for finding a procedure P satisfying a probabilistic implication $C\&P \rightarrow G$ as described above, where C is an observed context and G is a system goal. In this case the probability value of the implication provides the "objective function" that MOSES uses to assess the quality of candidate procedures.

For example, suppose an OCP-controlled agent is trying to learn to play the game of "tag" (an application that has been carried out). Then its context C is that others are trying to play a game they call "tag" with it; and we may assume its goals are to please them and itself, and that it has figured out that in order to achieve this goal it should learn some procedure to follow when interacting with others who have said they are playing "tag." In this case a potential tag-playing procedure might contain nodes for physical actions like $step_forward(speed\ s)$, as well as control flow nodes containing operators like $ifelse$ (e.g. there would probably be a conditional telling the robot to do something different depending on whether someone seems to be chasing it). Each of these program tree nodes would have an appropriate knob assigned to it. And the objective function would evaluate a procedure P in terms of how successfully the robot played tag when controlling its behaviors according to P (noting that it may also be using other control procedures concurrently with P). It's worth noting here that evaluating the objective function in this case involves some inference already, because in order to tell if it is playing tag successfully, in a real-world context, it must watch and understand the behavior of the other players.

MOSES operates according to the following process for evolving a metapopulation of "demes" of programs (each deme being a set of relatively similar programs, forming a sort of island in program space):

1. Construct an initial set of knobs based on some prior (e.g., based on an empty program; or more interestingly, using prior knowledge **supplied by PLN** based on the system's memory) and use it to generate an initial random sampling of programs. Add deme to the metapopulation.
2. Select a deme from the metapopulation and update its sample, as follows:
   (a) Select some promising programs from the deme's existing sample to use for modeling, according to the objective function.
   (b) Considering the promising programs as collections of knob settings, generate new collections of knob settings by applying some (competent) optimization algorithm. For best performance on difficult problems, it is important to use an optimization algorithm that makes use of the system's memory in its choices, **consulting PLN inference** to help estimate which collections of knob settings will work best.
   (c) Convert the new collections of knob settings into their corresponding programs, reduce the programs to normal form, evaluate their scores, and integrate them into the deme's sample, replacing less promising programs. In the case that objective is expensive, score evaluation may be preceded by score estimation, which may use **PLN inference**, enaction of procedures in an **internal simulation environment**, and/or similarity matching against **episodic memory**.
3. For each new program that meet the criterion for creating a new deme, if any:
   (a) Construct a new set of knobs (a process called "representation-building") to define a region centered around the program (the deme's *exemplar)*, and use it to generate a *new* random sampling of programs, producing a new deme.
   (b) Integrate the new deme into the metapopulation, possibly displacing less promising demes.
4. Repeat from step 2.

MOSES is a complex algorithm and each part plays its role; if any one part is removed the performance suffers significantly (Looks 2006). However, the main point we want to highlight here is the role played by synergetic interactions between MOSES and other cognitive components such as PLN, simulation and episodic memory, as indicated in **boldface** in the above pseudocode. MOSES is powerful, but still has scalability issues; one reason we feel it has potential to play a major role in human-level AI is its capacity for productive interoperation with other cognitive components.

Continuing the "tag" example, the power of MOSES's integration with other cognitive processes would come into play if, before learning to play tag, the agent has already played simpler games involving chasing. If the agent already has experience chasing and being chased by other agents, then its episodic and declarative memory will contain knowledge about how to pursue and avoid other agents in the context of running around an environment full of objects, and this knowledge will be deployable within the appropriate parts of MOSES's Steps 1 and 2. Cross-process and cross-memory-type integration make it tractable for MOSES to act as a "transfer learning" algorithm, not just a task-specific machine-learning algorithm. OCP experiments of this nature have been conducted with virtual agents, but not yet robots.

Cognitive Synergy in PLN . While MOSES handles much of OCP's procedural learning, and OpenCog's internal simulation engine handles most episodic knowledge, OCP's primary tool for handling declarative knowledge is an uncertain inference framework called Probabilistic Logic Networks (PLN). The complexities of PLN are the topic of a lengthy technical monograph (Goertzel et al. 2008), and here we will eschew most details and focus mainly on pointing out how PLN seeks to achieve efficient inference control via integration with other cognitive processes.

As a logic, PLN is broadly integrative: it combines certain term logic rules with more standard predicate logic rules, and utilizes both fuzzy truth values and a variant of imprecise probabilities called *indefinite probabilities*. PLN mathematics tells how these uncertain truth values propagate through its logic rules, so that uncertain premises give rise to conclusions with reasonably accurately estimated uncertainty values.

PLN can be used in either forward or backward chaining mode; and in the language introduced above, it can be used for either analysis or synthesis. As an example, we will consider backward chaining analysis, exemplified by the problem of an AI preschool student trying to determine whether a new playmate "Bob" is likely to be a regular visitor to is preschool or not (evaluating the truth value of the implication $Bob \rightarrow regular\_visitor$). The basic backward chaining process for PLN analysis looks like:

1. Given an implication $L \equiv A \rightarrow B$ whose truth value must be estimated (e.g. $L \equiv C\&P \rightarrow G$ as discussed above), create a list $(A_1, ..., A_n)$ of *(inference rule, stored knowledge)* pairs that might be used to produce L
2. Using analogical reasoning to prior inferences, assign each A_i a probability of success
   - If some of the A_i are estimated to have reasonable probability of success at generating reasonably confident estimates of L's truth value, then invoke Step 1 with A_i in place of L (at this point the inference process becomes recursive)
   - If none of the A_i looks sufficiently likely to succeed, then inference has "gotten stuck" and another cognitive process should be invoked, e.g.
     - **Concept creation** may be used to infer new concepts related to A and B, and then Step 1 may be revisited, in the hope of finding a new, more promising A_i involving one of the new concepts
     - **MOSES** may be invoked with one of several special goals, e.g. the goal of finding a procedure P so that $P(X)$ predicts whether $X \rightarrow B$. If MOSES finds such a procedure P then this can be converted to declarative knowledge understandable by PLN and Step 1 may be revisited....

– **Simulations** may be run in OCP's internal simulation engine, so as to observe the truth value of $A \rightarrow B$ in the simulations; and then Step 1 may be revisited....

The combinatorial explosion of inference control is combatted by the capability to defer to other cognitive processes when the inference control procedure is unable to make a sufficiently confident choice of which inference steps to take next. Note that just as MOSES may rely on PLN to model its evolving populations of procedures, PLN may rely on MOSES to create complex knowledge about the terms in its logical implications.

In the "new playmate" example, the interesting case is where the robot initially seems not to know enough about Bob to make a solid inferential judgment (so that none of the A_i seem particularly promising). E.g., it might carry out a number of possible inferences and not come to any reasonably confident conclusion, so that the reason none of the A_i seem promising is that all the decent-looking ones have been tried already. So it might then recourse to MOSES, simulation or concept creation.

For instance, the PLN controller could make a list of everyone who has been a regular visitor, and everyone who has not been, and pose MOSES the task of figuring out a procedure for distinguishing these two categories. This procedure could then used directly to make the needed assessment, or else be translated into logical rules to be used within PLN inference. For example, perhaps MOSES would discover that older males wearing ties tend not to become regular visitors. If the new playmate is an older male wearing a tie, this is directly applicable. But if the current playmate is wearing a tuxedo, then PLN may be helpful via reasoning that even though a tuxedo is not a tie, it's a similar form of fancy dress – so PLN may extend the MOSES-learned rule and infer that the new playmate is not likely to be a regular visitor.

Discussion

From a theoretical standpoint, one key question posed by this material is how critical the cognitive-synergetic aspects actually are for the OCP and OCB systems' practical intelligence. A theoretical demonstration of the necessity of cognitive synergy for general intelligence in certain sorts of environments has been previously attempted (Goertzel 2009b), but remains sketchy and semi-rigorous; so it is very tempting to try to explore this issue empirically. One may well ask: What if one disallowed cognitive synergy and forced the system to utilize its components in a completely separated way – would it operate less intelligently?

As natural as this question may seem, however, it turns out not to make so much sense. The OCP and OCB designs have been constructed around the principle of cognitive synergy, so the answer is that if one disables cross-cognitive-process interactions it won't work nearly as well – but this doesn't resolve the deeper question of whether one could somehow modify the various component cognitive processes so as to lead to highly effective whole-system performance in the absence of significant cognitive synergy. It is extremely difficult to isolate the impact of particular properties of complex, tightly-linked integrative AI systems; much more so than evaluating the effectiveness of such a system at carrying out particular sorts of tasks.[1]

References

Arel, I.; Rose, D.; and Coop, R. 2009. Destin: A scalable deep learning architecture with application to high-dimensional robust pattern recognition. *Proc. AAAI Workshop on Biologically Inspired Cognitive Architectures*.

Bach, J. 2009. *Principles of Synthetic Intelligence*. Oxford University Press.

Fauconnier, G., and Turner, M. 2002. *The Way We Think: Conceptual Blending and the Mind's Hidden Complexities*. Basic.

Goertzel, B., and Bugaj, S. V. 2009. Agi preschool. In *Proceedings of the Second Conference on Artificial General Intelligence*. Atlantis Press.

Goertzel, B., and Et Al, C. P. 2008. An integrative methodology for teaching embodied non-linguistic agents, applied to virtual animals in second life. In *Proceedings of the First Conference on Artificial General Intelligence*. IOS Press.

Goertzel, B., and Pennachin, C. 2005. *Artificial General Intelligence*. Springer.

Goertzel, B.; Pinto, H.; Pennachin, C.; and Goertzel, I. F. 2006. Using dependency parsing and probabilistic inference to extract relationships between genes, proteins and malignancies implicit among multiple biomedical research abstracts. In *Proceedings of Bio-NLP 2006*.

Goertzel, B.; Ikle, M.; Goertzel, I.; and Heljakka, A. 2008. *Probabilistic Logic Networks*. Springer.

Goertzel, B.; Pitt, J.; Ikle, M.; Pennachin, C.; and Liu, R. 2010. Glocal memory: a design principle for artificial brains and minds. *Neurocomputing, Special Issue of Artificial Brain*.

Goertzel, B.; Pennachin, C.; and Geisweiller, N. 2010. *Building Better Minds: Engineering Beneficial General Intelligence*. In preparation.

Goertzel, B. 2006. *The Hidden Pattern*. Brown Walker.

Goertzel, B. 2009a. Cognitive synergy: A universal principle of feasible general intelligence? In *Proceedings of ICCI-09*.

Goertzel, B. 2009b. The embodied communication prior. In *Proceedings of ICCI-09*.

Goertzel, B. e. a. 2010. Opencogbot: An integrative architecture for embodied agi. *Proceedings of ICAI-10, Beijing*.

Looks, M. 2006. *Competent Program Evolution*. PhD Thesis, Computer Science Department, Washington University.

Tulving, E., and Craik, R. 2005. *The Oxford Handbook of Memory*. Oxford University Press.

[1]This work was partially funded by Chinese NSF grant $60975084/F030603$, "Learning by Imitation and Reinforcement in a Humanoid Robot", and a Hong Kong ITF grant for "A Software Toolkit for Creating Intelligent Non-Player Characters in Video Games"