XIA-MAN:
An Extensible, Integrative Architecture for Intelligent Humanoid Robotics

Ben Goertzel[1], Hugo de Garis[2]

[1]Novamente LLC, 1405 Bernerd Place, Rockville MD 20851, USA, ben@goertzel.org

[2]Artificial Brain Lab, School of Information Science and Technology,
Xiamen University, Xiamen, CHINA, profhugodegaris@yahoo.com

Abstract

The XIA-MAN architecture for intelligent humanoid robot control is proposed, a novel design in which perception and action are achieved via a combination of GA-evolved neural-net modules with existing open-source software packages; and cognition is achieved via the OpenCog Prime framework, an open-source variant of the Novamente Cognition Engine. XML is used to communicate between components, enabling simple pluggability of additional or modified components, and leading to the name XIA-MAN (eXtensible Integrative Artificial Man). XIA-MAN's neural net modules are used to convert between high-dimensional numerical representations denoting perceptions and actions, and probabilistic symbolic representations suitable for cognitive manipulation. XIA-MAN's Cognition Engine is used, at each time cycle, to choose a high-level behavioral procedure that is believes is likely to enable the robot to achieve its goals in the current context. This integration provides an extremely pragmatic approach to achieving intelligent robot functionality given currently available technologies; and the multimodular architecture, bound together by intricate nonlinear inter-component feedbacks, is conceptually reminiscent of the complexly interconnected multimodular architecture of the brain. Current work involves an initial incarnation of the XIA-MAN architecture using the Nao humanoid robot, to create an intelligent humanoid called XiaoNao[a].

Introduction

Providing a humanoid robot with a generally intelligent control system is a highly complex task. Even if one sets goals far short of the accurate imitation of human embodied behaviors (the "robotic Turing Test", so to speak), one is faced with myriad interdependent difficulties, the result being that the state of the art in humanoid robot control is best described as sorely disappointing.

Among other problems, contemporary humanoid robotics is mainly concerned with lower-level issues such as seeing, walking and grasping – the matter of robot cognition generally being deferred to the indefinite future in which sensorimotor issues have already been resolved. Put simply: contemporary humanoid robots don't really think, not even to the fairly limited extent that some other AI programs currently do.

The pragmatic motivation for this current near-exclusive focus on sensorimotor issues is obvious, but the key risk of this approach is equally obvious: in the human brain, it seems clear, perception, action and cognition are inextricably entangled. Yes, lower-level vision and movement are carried out independently of cognition, but once you get into areas like object identification or the coordination of complex movements, the boundary between perception, action and cognition becomes impossible to draw.

One may argue for the plausibility of an approach that begins with sensation and action and then builds up to cognition – since this, after all, what evolution seems to have done, to a great extent. But, not every approach to perception and action that seems to marginally work in a robot, is going to be tractably extensible toward perception/action-based cognition. There is, in the robotics literature, a definite lack of convincing arguments as to exactly why current approaches to achieving sensorimotor functionalities are likely to be growable into viable approaches to cognitive robotics.

One way to work around this problem would be to create better approaches to sensation and action: approaches that are more clearly extensible in a cognitive direction. Another route would be to create better argument as to why current approaches actually are appropriate foundations for longer-term work. And a third path is the integrative one.

There already exist fairly successful approaches to computational reasoning, learning and other cognitive faculties. There are even integrative AI systems, with

[a] "XiaoNao" is pronounced "show(er) now" and means "little brain". "Nao" is the product name given by Aldebaran Robotics, Paris, France, to their robot, shown in Fig. 1. "Nao" is also the Chinese word for "brain". (Coincidence?) "Xiao" (i.e. "little") is an affectionate prefix usually given to Chinese children. XiaoNao is only 58 cms tall.

ambitions toward general intelligence, combining multiple cognitive faculties in a psychologically realistic way (Goertzel and Pennachin, 2006; Wang and Goertzel, 2008). So, one potential path to achieving impressively intelligent humanoid robot functionality is to integrate state of the art approaches to robot sensation and action, with state of the art approaches to general cognition.

Perils loom along this path, of course – most notably, there is the risk of creating a system of overly modular components that appear to each other wholly as "black boxes," and hence lack the subtle and dynamic intercomponent feedbacks that are almost surely necessary for intelligent integrated functionality. However, we believe that these perils can be avoided, and that the integrative approach properly executed constitutes the most viable route to achieving intelligent humanoid robotics in the relatively near future.

The XIA-MAN Architecture

In this paper we outline a particular, integrative architecture for intelligent humanoid robotics, which we call XIA-MAN. Pronounced "zha-man," this stands for "eXtensible Integrative Artificial Man," and is also chosen due to the origins of the architecture at a series of meetings at Xiamen University in China. Initially we will be implementing XIA-MAN in the context of controlling a Nao robot, but XIA-MAN is not specifically tied to the Nao, and may be used for any humanoid robot, or in fact for any mobile robot that possesses 3D vision, robust acoustic detection and production, and a sufficient number of actuators with a collectively reasonably large number of degrees of freedom.

A high-level outline of the XIA-MAN architecture is given in Figure 1, with more details added in Figures 2 and 3-5. As shown there, the architecture is a combination of three key aspects:

- Evolved neural net modules for certain perception and action functionalities
- Existing open-source software modules for other perception and action functionalities
- An integrative cognition engine (OpenCog) for abstract cognition, behavior learning, and overall system control

While this may at first glance seem an unruly hodge-podge, in fact it is a carefully chosen combination of tools and methods, with a systematic conceptual foundation. The different components in the architecture are interconnected using XML messaging, making the system extremely extensible and flexible, which is appropriate given the amount of experimentation that will necessarily be done with any such system.

In the XIA-MAN architecture, neural nets are proposed for functionalities involving high-dimensional numerical inputs and/or outputs. The neural nets, in effect, translate back and forth between high-dimensional numerical vectors and probabilistic truth values attached to discrete symbols. In other words, if one accepts a theory of cognition in which the latter consists largely of probabilistic symbol manipulations, then one may say that the role of the neural nets in XIA-MAN is to convert back and forth between the high-dimensional numerical language of the physical world, and the probabilistic symbolic language of the cognitive world. The devil is in the details here as usual, and we have sought to work the details out in a sensible way: for instance, evolving neural nets that output pairs of numbers interpretable as two-component (probability, confidence) truth values, of the sort used within the OpenCog engine's logical inference algorithms.

Complementing use of evolved neural nets, however, the use of existing open-source tools in XIA-MAN is proposed where these tools are clearly functionally adequate. For instance, it would be possible to evolve neural nets carrying out text-to-speech conversion, but there already exist reasonably operational software packages carrying out this function and outputting probability values attached to different textual interpretations of each speech act. Thus, we propose to at least initially make use of this open-source speech-to-text software in our system, perhaps replacing it at a later date if it proves insufficient.

A Two-Phase Development Plan

Referring again to Figure 1, we conceive the realization of the XIA-MAN architecture as taking place in two phases.

In Phase 1, the evolution of neural modules is restricted to three functionalities: object recognition, movement control, and speech emotion/modality analysis. Furthermore, the evolution takes place "offline," rather than in real-time in the course of robot interactions. And, in this phase, language functionality is adaptive only to a limited extent: the probabilistic weights guiding the usage of different linguistic rules is adaptively modified, but the rules themselves are not experientially learned or modified.

The key tools required for Phase 1 already exist: a framework for neural net evolution on ordinary hardware as well as FPGA's; a functioning version of the OpenCog Cognition Engine; OpenCog-compatible software currently used to control virtual animals in virtual worlds; and assorted other open-source components. However, a great deal of customization and a moderate amount of extension of these components will be needed in order to make Phase 1 a reality.

In Phase 2 of XIA-MAN development, on the other hand, we envision lifting the key restrictions posed in Phase 1. Real-time neural module evolution and linguistic rule adaptation are clearly possible without our conceptual and mathematical design and our software framework, and they are deferred until a second phase merely out of a practical desire to simplify the initial implementation and testing.

Examples of Desired XIA-MAN Behaviors

The specific goals for the different phases of the XIA-MAN project are in the process of being defined. The basic idea, however is simple to state: the robot should be able to hold a simple conversation about the things in the laboratory it lives in. Within this limited domain, it should be able to answer simple questions, and respond to simple commands, and understand simple statements and store them in its memory for future use. A very partial, somewhat arbitrary list of relevant examples of behaviors desirable from Phase 1 XIA-MAN would be:

- Someone says "Where is Hugo?" and then the robot points at Hugo (assuming Hugo is in the room)
- Ben says "This is my chair" to the robot ... then, later on, Hugo asks the robot "Where is Ben's chair?" and the robot walks over to the chair.
- The ability to teach the robot simple behaviors by imitation and reinforcement learning -- similar to the virtual controlled by the Novamente Pet Brain (Goertzel et al, 2008, 2008a) to be discussed below. For instance, you can give the robot an example of the behavior "give the ball to Hugo", and then if you ask it later to "give the block to Ruiting" it will understand and be able to do it, by following the previous example.
- The robot can respond to emotion: if you yell at it, it goes away. If you speak to it in a sweet and pleasant tone, it comes toward you. Unless it has a twisted personality, which is also possible.
- The robot has a basic understanding of itself as demonstrated by its conversational responses. For instance, if you ask it "Where are you?", "Who is near you?", "What are you doing?" it understands and can answer.
- Basic mirroring behavior: the robot can copy what it sees people do, and also sets its internal "emotional state" based in part on the emotions it perceives in others (based on voice analysis, as will be described below).

In What Sense Is XIA-MAN Biologically-Inspired?

Next, as this paper is part of a symposium on "Biologically-Inspired Cognitive Architectures," it seems appropriate to give a brief commentary on the two senses in which the XIA-MAN architecture is, in fact, biologically-inspired.

Firstly, several components in the architecture are neural-net based, and hence brain-inspired in a rather direct way. Furthermore, the learning algorithm used to create the details of these components is a process of artificial evolution; and according to Edelman's Neural Darwinism (Edelman, 1989) and related theories, evolutionary learning is an accurate model of many of the neural-net learning processes occurring in the brain.

Secondly, and equally importantly, the qualitative nature of the overall architecture is brain-inspired. According to Mithen (1999) the human brain consists of a set of relatively distinct intelligent modules, focused on different sorts of tasks, and including an integration module in which the outputs of other modules mix together freely. The intricate feedbacks between the modules cause their nonlinear dynamical coupling, yet this does not eliminate their essential distinctness. XIA-MAN involves the same sort of modularity that Mithen hypothesizes to occur in the brain; however, only some of the modules are realized in a brainlike way.

The integration module itself, which in our architecture consists of the OpenCog AtomTable knowledge store and its associated scheduling and maintenance processes, is the bridge between the closely-brainlike and less-brainlike modules of the system, a role that is enabled by its novel knowledge representation scheme which combines aspects of neural nets with aspects of probabilistic logic based semantic nets.

Evolving Neural Nets Carrying Out Perceptual and Motor Functions

We now describe the approach underlying some of the key sensorimotor components of our the XIA-MAN architecture: the use of genetic algorithms to "evolve" neural networks carrying out desired functions. This evolution may be carried out in software, and in certain cases it may also be carried out in hardware as well, using field-programmable gate arrays (e.g. Xilinx 2008).

In principle, it would be possible to create an intelligent humanoid robot control system entirely by evolving neural modules using this methodology, and connecting them together appropriately. However, at present this approach is rendered difficult due to the absence of a detailed theoretical understanding of how to achieve advanced cognition using interconnected neural networks. Thus in the proposed integrative architecture, we utilize neural module evolution in an important yet more limited role, to handle aspects of robot perception and actuation, for which there exists current theoretical clarity regarding the appropriate functionality and interconnection of the neural modules being evolved.

The basic framework for neural net evolution utilized here has been described before (e.g. de Garis and Korkin 2002), so here we will recapitulate it only briefly (with a focus on newly developed aspects, such as a novel neural net representation), and then move on to describing its specific application to three critical aspects of intelligent humanoid robotics: object recognition, movement control, and vocal emotion/modality analysis.

Genetic Algorithm Evolution of Neural Net Modules in Software and Hardware, Using the Parcone Model

Since the late 1980s, researchers have evolved neural networks using genetic algorithms (Goldberg 1989). Typically, this occurs by concatenating the (numerical) weights of a neural net into long bit string "chromosomes" that are then evolved in a genetic algorithm so that the output control signals of a neural net perform some desired function, e.g. motion control of a robot, or visual pattern recognition.

This idea can also be implemented in hardware, typically using FPGAs (field programmable gate arrays). Ordinary high level GA code used to evolve neural nets is "hardware compiled" or "silicon compiled" into the configuring bits of a programmable chip. Once configured, the chip runs at hardware speeds to evolve neural networks.

In his past work on evolvable neural networks (e.g. de Garis and Korkin, 2002), Prof de Garis evolved "fully connected" neural networks, i.e. each neuron had a connection (and hence a weight) with every other neuron in the network. Assuming each neuron had a connection that fed back to itself, a fully connected neural network of N neurons had N^2 total connections. If N was not large, fully connected neural nets were readily evolvable.

In our proposed project however, we wish to use neural net modules to perform such ambitious tasks as object recognition, complex motion control of robots, etc, which necessitate our neural nets having many neurons, e.g. accepting 10,000s of pixels from digital cameras. With such large N, it is impractical to insist that the neural net modules to be evolved are fully connected. They therefore have to be "partially connected", hence our invention of a new neural net model, we call the "Parcone (i.e. partially connected neural evolvable) model".

The basic idea of the Parcone model is that each neuron keeps a list of all the other neurons that send a signal to it. Each neuron in the network has an integer ID. All the IDs of the connecting neurons of a given neuron are stored in a hash table (for quick access and limited storage requirements). This hash table is actually a table of pointers to data structures that store such information as the ID of the neuron that sends a signal, the weight value, and the bit string equivalent of that value, as shown in Fig. 3.

Each neuron calculates its output signal by consulting its own hash table, and a "signal table" that lists all the output signals of all the individual neurons. The output of a neuron is placed into a second signal table that will be used to calculate the output signals of the neurons at the next "round" of calculations. (A round is called a "tick" in our work.)

The output signal(s) of certain neurons serve as the output of the whole network. These output signals are used to calculate the fitness of the neural net.

The Parcone model has 3 types of mutation. One simply flips a random bit of a weight of a random neuron. A second deletes a random connection between two neurons. A third adds a connection between two random neurons, with a random new weight value (and bits).

Currently the Parcone model is being tested on an ordinary PC; but in the near future it will be implemented on a supercomputer, and as far as possible on the latest version of FPGAs, using Xilinx's new "Virtex-5" chip. The Parcone model does have the advantage that it allows large numbers of inputs to a single module, but the down side is that if the neural net module has many neurons, it will usually take longer to evolve than a smaller neural net module. Hence the value of implementing the neural net evolution in hardware, to take advantage of hardware speeds compared to those of ordinary PCs, is imperative.

For years, Prof de Garis has been trying to make the evolution of neural networks faster, so that evolving 10,000s of them to serve as components of artificial brains becomes practical; the combination of the Parcone model with appropriate usage of modern supercomputers and specialized hardware may now finally render this possible.

The question of "artificial brain architecture" – what the various modules involved in an artificial brain do, and how they are connected – is largely separate from the particulars of the neural module evolution methodology. Many approaches are possible here, and as neuroscience advances further and further, it will be possible to draw more and more ideas about artificial brain architecture from that domain. One approach that seems very practical at present is the integrative approach taken in this paper, where neural module evolution is used for certain aspects of artificial-brain function where the fitness function and the intermodule connectivity patterns are particularly clear, and other AI approaches are used to handle the other artificial-brain functions.

Evolving Neural Modules for Object Recognition

In order to apply neural module evolution to object recognition, initially, a supervised learning approach will be taken. We will evolve separate neural modules for each of a large number of object types.

For each object type (e.g., "cup", "table", "person", etc.), we begin by assembling a collection of training images, as follows. Define a "zoomed image" as an image that is dominated by a single object (as opposed to being a large and complex visual scene). To construct the training collection, we have the robot look at a number of realistic situations containing objects of the given type, and we then use heuristic methods to select regions of the perceived image containing objects of the specified type. These regions are then taken as zoomed images, and added to the collection of training images. We then evolve a population of neural nets, each of which takes input consisting of an

appropriately preprocessed[1] zoomed image, and gives output that consists of a two numbers in [0,1]: one denoting the probability that the zoomed image represents an object of the specified type; and the other denoting the confidence that the net has in the probability produced. This two-component output is key to the integrity of the overall integrated architecture, because the OpenCog engine that plays the cognitive role in the architecture labels its internal knowledge-items with multi-component truth values possessing a similar semantics (see Goertzel, Ikle' et al, 2008 for a detailed treatment of multiple-component truth values).

The fitness function assessing the fitness of a candidate neural net is based on the accuracy of the net's output, but weighting the error penalty based on the confidence output. A typical fitness formula used in neural net evolution is of the form

$$F = 1/(\Sigma(tval(t) - oval(t))^2)$$

where the sum is over all output nodes t, and

tval(t) = target value (t)
oval(t) = output value (t)

Here we modify the formula to look like

$$F = 1/(\Sigma(w(t)\ (tval(t) - opval(t))^2))$$

where

opval(t) = output probability value (t)
ocval(t) = output confidence value (t)
w(t) = ocval(t) / N

and the normalizing factor N is the sum of ocval(t) over all output nodes t.

In addition the fitness function may also contain other terms such as a parsimony pressure (a multiplicative factor that gives extra fitness to small trees). In prior work we have found that tuning the parsimony pressure via heuristics inspired by Solomonoff-Levin measure can yield significant value.

The subtlety of the above fitness function, with its incorporation of confidence output values, is that it does not require each neural net to correctly classify every object of the specified type. Rather, it also rewards nets that correctly classify objects within some subtype of the original type (e.g. "cups with handles" instead of general cups), so long as they correctly set their confidence output low when assessing an object of the original type but not their favorite subtype. This means that neural net evolution is serving not only for supervised categorization per se, but also for perception-based concept learning.

[1] In our framework, multiple image processing tools may be used to prepare the image for input into the neural nets, including Fourier transforms, wavelets, PCA, etc.

As noted above this evolution may be carried out in software or else on FPGA boards. The use of FPGA's potentially provides dramatically accelerated evolution, but also places constraints on the complexity of the evolutionary algorithm, and tends to push one toward using image preprocessing methods that result in very compact processed images.

Evolving Neural Modules for Movement Control

The application of neural module evolution to movement control in XIA-MAN is somewhat similar to the above-described application to object recognition, but also has somewhat different complexities. Similarly, we evolve nets corresponding to a number of movements (e.g. "step forward", "step backward", "grab object", etc.), via creating a training collection for each movement. But the inputs and outputs of the nets are more complex, which makes the fitness evaluation commensurately more complex.

The input of each neural net consists of two aspects:

- target data: numbers specifying the goal of the movement
- sensor data, each of which corresponds to a particular time offset relative to the initiation of the neural net's iteration

The target data has a different meaning depending on the specific movement being learned. For instance, for "step forward", the target data simply tells how far ahead the robot is supposed to step. For "grab object", the target data tells the location of the object the robot is supposed to grab.

The sensor data comprises data that is assumed to come into the robot during the course of carrying out the movement. This data may have various types, e.g. force data coming from the sensors on the robot's joints, or visual data in the case of a behavior involving sensorimotor coordination.

To evaluate the fitness of a neural net, one must allow the neural net to act over a period of time (which simulates the period of time over which the actual movement would be carried out by the robot, but may in practice be shorter). At the start of the period, the net is supplied with the target data. Then, the net is allowed to dynamically iterate, and the input nodes corresponding to sensor data are stimulated according to their corresponding time offsets. As it iterates, the net gives output by stimulating its output neurons, each of which corresponds to one degree of freedom of one of the robot's joints.

To assess the fitness of a "movement net" of this type, we first calculate a "phenotype" corresponding to the net. The phenotype consists of the time-series of outputs produced by the output neurons. This phenotype is then compared to training data, and assessed according to a fitness function similar to the one described above in the case of object classification.

How is the training data collected? Initially we have taken a simple approach involving creating it by hand. However, advanced development of the approach requires an automated method. We plan to explore two approaches:

1. Beginning from simple hand-created movements, create a large number of random variations on these, trying each one of these out for practical robot control in a robot simulator. Rate, for each variation, how well it achieves the target in the simulator. We then have a large set of good and bad examples, and the job of evolution is to find a neural net that abstracts the lessons contained in this set.
2. Using motion capture equipment, gather data from humans carrying out the specified movements. Then use this human data to create training examples. This approach becomes complex for movements that involve significant force-based sensor data, because the motion capture equipment does not record the sensations coming into the human's joints. It should work acceptably for movements involving visual sensor data, because in this case the human being recorded via motion capture equipment will also have their face orientation captured, so the visual scene perceived by them at any given time may be calculated.

Evolving Neural Modules for Emotion and Modality Recognition

The third use of evolved neural modules involved in Phase 1 XIA-MAN has to do with the processing of speech data. As noted above, our intention is to use existing open-source software to carry out text-to-speech and text-to-speech conversion. It might well be possible to exceed this software's capability using a combination of evolved neural modules and OpenCog's cognitive faculties; but given limited resources, we have chosen to focus our attention on areas where other approaches fall most severely short. Our intention is to apply neural net evolution to learn neural modules that carry out two classificatory functions: identifying the emotional content and intensity of a speech act, and identifying the modality of a speech act (is it a question, a statement, a command or an interjection)?

Again we propose a supervised categorization approach, in which training examples are created, each one consisting of a sound file corresponding to a human speech act, labeled with information as to the emotions it expresses (anger, fear, joy, etc.), the intensity of each expressed emotion, and the modality. As in the object recognition case, appropriate preprocessing must be done to turn the raw sound file into an abstracted feature vector suitable for input to a neural network. The neural net setup and fitness function here is similar to the object recognition case.

We view emotion and modality perception as a highly important aspect of robot perception, especially considering that in young children, this sort of capability normally arises prior to the ability to parse sentences or interpret semantic content.

From Supervised to Unsupervised Learning

All three applications of neural net evolution described above involve a supervised learning methodology, in which training examples are provided via programmer-created computer files. However, this is actually not how we envision neural net evolution being used in XIA-MAN in the long run. Mainly, we view supervised learning as a relatively "quick and dirty" way of supplying the robot with a variety of intelligent functionalities. Furthermore, the same neural net designs and fitness functions used in the above-described supervised learning experiments, may be used for Phase 2 XIA-MAN in the context of different learning approaches: human-teacher-based and unsupervised learning.

What we mean by human-teacher-based learning is supervised learning in which the training examples are pointed out to the robot by a human teacher interacting with the robot through its senses, rather than by human-supplied computer files. For instance, suppose a human points to a cup and tells the robot "This is a cup." Suppose the human teacher does this 100 times for 100 different cups. Then, in fact, the human has constructed a training collection for the robot, which the robot can use in exactly the same way as if the training collection had been supplied in advance in a set of files.

Or, consider the case of imitative learning (a kind of human-teacher-based learning, with which we have experimented extensively using virtual pets and the Novamente Cognition Engine). Suppose the human teacher tells the robot "I'm jumping" and then jumps. Suppose the human teacher does this 100 times. Then the robot has a large training collection of examples of jumping – but there's a problem, which is that these are images, rather than records of sensory inputs and the exertion of force on joints. So the fitness function has to be different here than in the movement learning approach described above. The fitness function must involve, for each candidate neural net being evolved, assessing the visual appearance of the movement that would result from that net, and comparing it to the training examples (which are observations of a jumping human). This requires the robot to run the candidate net in a robot simulator, and then do some 3D visual processing to compare what it sees in the simulator to the visually-provided training examples.

Next, to transition from supervised to unsupervised learning, what is needed is to enable the system to automatically construct its own training examples. But we have already prototyped this, to a certain extent, in analogous learning cases involving virtual pets and the Novamente Cognition Engine (to be briefly described below). It's not a large step conceptually, it just

introduces an additional source of noise into the process. For instance, suppose that instead of waiting for a human teacher to point to a cup and say "This is a cup," the robot simply notes which objects seem to be present in the visual scene when its human friends say the word "cup." Via this sort of *perceptual pattern mining*, a training corpus may be automatically created. Algorithmically, in the NCE and OpenCog designs, this sort of pattern recognition is carried out via automated *frequent subgraph mining* algorithms (see e.g. Thomas et al, 2006), although this has not yet been experimented with except in some special cases.

Architecturally, in this aspect of the envisioned Phase 2 XIA-MAN, it is the Cognition Engine component of the system that is concerned with automatically creating fitness cases, which are then fed to the neural module evolver component.

The Cognition Engine

In (Goertzel, 2006) general intelligence is described as the capability to achieve complex goals in complex environments, using limited computational resources. Using the specific neural net based methods described above, one may create a robot able to recognize objects, hear speech, and carry out movements and speech acts. These powerful capabilities are necessary but not sufficient for robotic general intelligence. What is needed is some way to connect the robot's goals with its perceptions and actions: in a word, cognition. The various aspects that need to be integrated to carry out this task are roughly outlined in Figure 6.

From this perspective, the basic purpose of the Cognition Engine component in XIA-MAN is to, at each time step in the robot's life: Enact a procedure so that, according to its best guess, the probabilistic logical impliation

Context & Procedure ==> Goals

is true with a high truth value. Here "Context" refers to the current situation as perceived by the robot; and a "Procedure" refers to an "internal program" running within the Cognition Engine, that executes a series of behaviors that the behavior postprocessor knows how to execute. Generally speaking such a procedure will contain an internal control flow involving loops, conditionals, etc. (e.g. "go get the object with ID #764 and bring it to the agent with ID #12"). Such a procedure may be internally represented by a data structure such as a tree or a graph, and may be textualized as a program in a language such as LISP (or, in XIA-MAN, the LISP-like language Combo), but with special primitives corresponding to the perceptions and behaviors of the robot.

The question then becomes how these probabilistic implications are learned. This of course depends on what Cognition Engine one is using. Here we propose to use OpenCog Prime (OCP), a new system which is a variant of the older Novamente Cognition Engine (Goertzel, 2006a). OCP has two ways of learning implications of the above nature: using an evolutionary learning mechanism (MOSES) that evolves procedures using the truth value of the implication as its fitness function; and a probabilistic logic engine (PLN) that derives the strength of the implication from background knowledge. In the following subsection we describe the NCE approach to cognition in a little more detail (still barely scratching the surface, however.)

Note that multiple representations of truth value are possible; however, one of the distinguishing features of the NCE/OCP approach is the use of multi-component probabilistic truth values such as indefinite probabilities (Goertzel, Ikle' et al, 2008). The two-component outputs of the neural nets describe above are highly appropriate in this context because there exist mathematical formulae for converting them into indefinite probabilities as used in OCP; this would not be the case for neural nets with single-number probability outputs. This is an example of the kind of detail that must be gotten right in order to create a highly effective integrative AI system.

The Novamente Cognition Engine

One way to conceptualize the NCE is to decompose into five aspects (which of course are not entirely distinct, but still are usefully distinguished):

- **Cognitive architecture** (the overall design of an AGI system: what parts does it have, how do they connect to each other)
- **Knowledge representation** (how does the system internally store declarative, procedural and episodic knowledge; and how does it create its own representation for knowledge of these sorts in new domains it encounters)
- **Knowledge creation** (how does it learn new knowledge of the types mentioned above; and how does it learn how to learn, and so on)
- **Instructional methodology** (how is it coupled with other systems so as to enable it to gain new knowledge about itself, the world and others)
- **Emergent structures and dynamics** (which arise from the combination of the four previous)

We now briefly review how these four aspects are handled in the NCE. For a more in-depth discussion of the NCE the reader is referred to (Goertzel, 2006) and the OpenCog wiki site (at opencog.org).

The NCE's high-level cognitive architecture is motivated by human cognitive science and is roughly analogous to Stan Franklin's LIDA architecture (Friedlander and Franklin, 2008). It consists of a division into a number of interconnected functional units corresponding to different specialized capabilities such as perception, motor control and language, and also an "attentional focus" unit corresponding to intensive

integrative processing. A diagrammatic depiction is given in (Goertzel et al, 2004).

Within each functional unit, declarative knowledge representation is enabled via an AtomTable software object that contains nodes and links (collectively called Atoms) of various types representing declarative, procedural and episodic knowledge both symbolically and subsymbolically. Each Atom is labeled with a multi-component probabilistic truth value object; and also with a multi-component attention value object indicating its short and long term importance.

Procedural knowledge is represented via program trees in a simple LISP-like language called Combo; and methods exist for translating between Combo and declarative Atoms. Episodic knowledge is represented by the use of Atoms and Combo programs to trigger internal simulations in a UI-free internal virtual world called Third Life, which may be thought of as the system's "mind's eye" running internal (memory-based or hypothetical) movies.

Each unit also contains a collection of MindAgent objects implementing cognitive, perception or action processes that act on this AtomTable, and/or interact with the outside world.

In addition to a number of specialized learning algorithms associated with particular functional units, the NCE is endowed with two powerful learning mechanisms embedded in MindAgents: the MOSES probabilistic-program-evolution module (based on Looks, 2006), and the Probabilistic Logic Networks module for probabilistic logical inference (Goertzel, Ikle' et al, 2008). These are used both to learn procedural and declarative knowledge, and to regulate the attention of the MindAgents as they shift from one focus to another, using an economic attention-allocation mechanism (Goertzel, 2006a) that leads to subtle nonlinear dynamics and associated emergent complexity including spontaneous creative emergence of new concepts, plans, procedures, etc.

Regarding teaching methodology, the NCE has been developed in the context of a physically or virtually embodied approach, which integrates linguistic with nonlinguistic instruction, and also autonomous learning via spontaneous exploration of the physical or virtual world. It is the exploration of the world, and the interaction with other (human) minds in the context of the world, that will, we suggest, allow the system's knowledge-based to adapt in such a way as to give rise to the high-level emergent structures characterizing a human-like mind: the phenomenal self (Metzinger, 2004), the illusion of will (Wegner, 2003), the theater of reflective awareness (Baars, 2001).

Novamente Pet Brain

The NCE is a highly general architecture, and various of its aspects have been implemented to various degrees. One practical application that has recently been constructed utilizing the NCE architecture is the Novamente Pet Brain (see (Goertzel et al, 2008) where it is called the VAB or Virtual Animal Brain). This application is particularly relevant to the present paper because it is the closest we have come to applying a Cognition Engine to controlling a humanoid robot: applying the NCE to controlling a dog-oid virtual-robot.

Figure 7 shows two virtual animals controlled by the Pet Brain. Figure 8 gives a high-level architecture diagram for the Pet Brain, which is a simplification of the overall NCE architecture as diagrammed in (Goertzel et al, 2004).

The capabilities of the Pet Brain-controlled virtual animals, in their current form, include

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

A prototype integration of the Pet Brain with Second Life was carried out (Goertzel et al, 2008), but the most thorough integration has been with the Multiverse virtual world.

Instruction of Pet Brain-controlled agents takes place according to a methodology we call IRC, involving three interacting aspects:

- *Imitative* learning: The teacher acts out a behavior, showing the student by example what he wants the student to do
- *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

The combination of these three sorts of instruction appears to us critical, for learning of complex embodied behaviors and also, further along, for language learning. Current experimentation with the IRC methodology has been interesting and successful, resulting in a framework allowing human-controlled avatars to teach Pet Brain-controlled agents a variety of behaviors such as fetching objects, delivering objects, going to desired locations, doing dances, and so forth. We believe the IRC learning approach is equally relevant to teaching humanoid robots as to virtual animals.

OpenCog Prime

The NCE and Pet Brain are proprietary software systems, but an open-source system has also been created, founded on a number of software objects donated by Novamente LLC and drawn from the NCE codebase. This OSS system is called OpenCog (Hart and Goertzel, 2008), and has two aspects: it is a fairly general framework for the development of AI and AGI systems; and it is also a specific means for the implementation of an AGI design called OpenCog Prime, which is an open-source analogue to the NCE. Our plan in the current architecture is to utilize the OpenCog framework, and in the context of this project to build out key aspects of the OpenCog Prime design.

The RelEx System for Natural Language Comprehension and Generation

OpenCog also contains a significant piece of software, donated by Novamente LLC, which is not strictly a part of the NCE although it interoperates with the NCE: this is the RelEx engine for natural language comprehension and generation. The comprehension aspect of RelEx is more mature and has been briefly described in (Goertzel et al, 2006) in the context of its application to interpreting biomedical research abstracts; the generation aspect is still in prototype phase, but has already been shown to work on simple sentences.

RelEx, in its comprehension aspect, takes English text and maps that text into abstract logical relations, in the Atom format utilized internally by the NCE and OpenCog. Generally speaking it produces multiple interpretations (logical relation sets) for each sentence it processes, and the task of selecting the contextually appropriate interpretation is left to the cognition engine itself. Also, the cognition engine is relied upon to correct errors RelEx may make in areas such as word sense disambiguation and reference resolution. It is anticipated that the sensory data gathered by a robot, regarding the physical and social context of instances of linguistic usages it produces or hears, may provide data helpful to the cognition engine in executing the linguistic tasks of interpretation-selection, reference resolution and sense disambiguation.

Next, in its generation aspect, RelEx maps logical relation sets (Atom sets) into sets of English sentences. Note that RelEx does not contain any facility for discourse management: this is assumed to be handled within the cognition engine. A design exists for controlling dialogue within OpenCog utilizing a probabilistic implementation of ideas from Rhetorical Structure Theory (Mann and Thompson, 1988), but this still awaits implementation.

RelEx also does not contain any facility for converting speech to text or vice versa. In the proposed integrated architecture for robot control, these conversions will be carried out by existing open-source software and RelEx will be used (together with the Cognition Engine) to select between the multiple outputs of speech-to-text software in an intelligent way.

Bringing Robot Simulators and Virtual Worlds Together

Finally, it is worth mentioning one point that has come to our attention in designing the XIA-MAN architecture, pertaining to the technical infrastructure available for teaching intelligent robots. At the present time, due to various difficulties involved in working with physical robots, a great deal of practical robotics work actually involved working with simulated rather than real robots. Robot simulators are fairly sophisticated these days, providing accurate simulations of physics as manifested in laboratory environments, if not real-world situations like forests, oceans and city streets. However, they lack one of the key strengths of virtual worlds such as Second Life and Multiverse (where the Novamente Pet Brain controlled animals have been prototyped): this is the capability for massive multiplayer interaction.

On the other hand, current game engines and entertainment virtual worlds have their own shortcomings. Figure 9 shows the simplified overall architecture via which one connects an AI engine to a contemporary virtual world or game. Note that the interaction occurs at a quite high level: the AI engine sees a very abstracted version of the visual scene, and it sends high-level motion commands rather than detailed actuator-control signals.

Robot simulators exceed virtual worlds and game engines in terms of allowing detailed control of simulated robot joints, and physically realistic interactions between objects (as needed for instance to enable tool use), but don't allow large numbers of individuals to log on and help teach robots. It seems that the integration of robot simulator and virtual world technology has a great potential to advance intelligent robotics, via supplying robots with a large number of teachers. We are exploring ways of bringing this integration about, such as modifying the OpenSim (opensimulator.org) virtual world to use the Gazebo (see playerstage.sourceforge.net) robot simulator in place of its current physics engine.

Conclusion and Further Work

The architecture described here is a complex one, and no doubt many surprises will emerge during the course of its practical development. We have plotted out the future course of the XIA-MAN project through Phases 1 and 2 as described above, and even those phases will doubtless be adapted as the work progresses; but the greatest uncertainty arises after those phases.

However, both the XIA-MAN architecture and the OpenCog engine itself are highly flexible and extensible, and the neural net evolution approach is also a very flexible and broadly applicable one. We are confident that

the structures and dynamics involved in XIA-MAN are flexible enough to be adaptively improved as robot technology improves and as avenues for achieving yet more sophisticated approaches to aspects of perception, cognition and action become evident.

Acknowledgements

The ideas described here originated in the course of discussions with a number of individuals at Xiamen University, including Rui Liu, Guo Junfei, Chen Shuo, Pan Wei, Lian Ruiting, Min Jiang, Miao Kehua, and others.

References

1. (Baars, 2001). Baars, Bernard (2001). In the Theater of Consciousness. Oxford University Press USA: New York
2. (de Garis and Korkin 2002) Neurocomputing, "THE CAM-BRAIN MACHINE (CBM), An FPGA Based Hardware Tool which Evolves a 1000 Neuron Net Circuit Module in Seconds and Updates a 75 Million Neuron Artificial Brain for Real Time Robot Control", Hugo de Garis, Michael Korkin, Neurocomputing journal, Elsevier, Vol. 42, Issue 1-4, February, 2002. Special issue on Evolutionary Neural Systems, guest editor Prof. Hugo de Garis.
3. (Edelman, 1989) Edelman, Gerald (1989). Neural Darwinism. Basic Books: New York.
4. (Friedlander and Franklin, 2008) Friedlander, David and Stan Franklin. 2008. LIDA and a Theory of Mind. In Artificial General Intelligence (AGI-08), ed. Ben Goertzel and Pei Wang. Memphis, TN, USA: IOS Press.
5. (Goertzel and Pennachin, 2006) Goertzel, Ben and Cassio Pennachin (2006). Artificial General Intelligence. Springer Verlag: New York.
6. (Mithen, 1999) Mithen, Stephen (1999). The Prehistory of Mind. Thames and Hudson: New York
7. (Wang and Goertzel, 2008) Wang, Pei and Ben Goertzel (2008). Advances in Artificial General Intelligence. IOS Press: Amsterdam.
8. (Wegner, 2003). Wegner, Daniel. The Illusion of Conscious Will. MIT Press: Cambridge MA
9. (Goertzel, 2004). Goertzel, Ben, Moshe Looks and Cassio Pennachin (2004). Novamente: An Integrative Architecture for Artificial General Intelligence. Proceedings of AAAI Symposium on Achieving Human-Level Intelligence through Integrated Systems and Research, Washington DC, August 2004
10. (Goertzel, 2006a) Goertzel, Ben (2006). Patterns, Hypergraphs and General Intelligence. Proceedings of International Joint Conference on Neural Networks, IJCNN 2006, Vancouver CA
11. (Goertzel et al, 2006) Goertzel, Ben, Hugo Pinto, Ari Heljakka, Michael Ross, Izabela Goertzel, Cassio Pennachin (2006). 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
12. (Goertzel, 2008) Goertzel, Ben (2008). A Pragmatic Path Toward Endowing Virtually-Embodied AIs with Human-Level Linguistic Capability, Special Session on Human-Level Intelligence, IEEE World Congress on Computational Intelligence (WCCI) Hong Kong, 2008
13. (Goertzel et al, 2008) Goertzel, Ben (2008). An Integrative Methodology for Teaching Embodied Non-Linguistic

Agents, Applied to Virtual Animals in Second Life, in Proceedings of the First AGI Conference, Ed. Wang et al, IOS Press
14. (Goertzel et al, 2008a) Goertzel, Ben, Cassio Pennachin and Carlos Lopes. An Inferential Dynamics Approach to Personality and Emotion Driven Behavior Determination for Virtual Animals. The Reign of Catz and Dogz Symposium, AI and the Simulation of Behavior (AISB), Edinburgh, 2008
15. (Goertzel, 2006) Goertzel, Ben (2006). The Hidden Pattern. Brown Walker: New York
16. (Goertzel, 2006a). Goertzel, Ben (2006). 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, IOS Press.
17. (Goertzel, Ikle' et al, 2008). Goertzel, Ben, Matthew Ikle', Izabela Goertzel and Ari Heljakka (2008). Probabilistic Logic Networks. Springer Verlag: New York.
18. (Goldberg 1989) David Goldberg, "Genetic Algorithms in Search, Optimization, and Machine Learning", Addison- Wesley, 1989.
19. (Hart and Goertzel, 2008). Hart, David and Ben Goertzel (2008). OpenCog.
20. (Looks, 2006). Looks, Moshe (2006). Competent Program Evolution. PhD thesis, Computer Science Department, Washington University in St. Louis.
21. (Mann and Thompson, 1988). Mann, W. and S. Thompson (1988): Rhetorical Structure Theory: Toward a Functional Theory of text Organization. In: Text 8(3), S. 243-281
22. (Metzinger, 2004). Metzinger, Thomas (2004). Being No One. MIT Press: Cambridge MA
23. (Thomas et al, 2006). Thomas, L.T.; Valluri, S.R.; Karlapalem, K. (2006). MARGIN: Maximal Frequent Subgraph Mining. Sixth International Conference on Data Mining, pp. 1097 - 1101
24. (Xilinx 2008) Xilinx Inc., http://www.xilinx.com

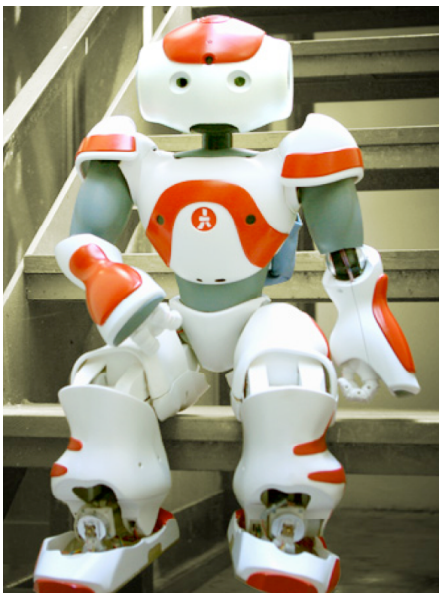

Figure 1. The Nao humanoid robot, XIA-MAN's first body

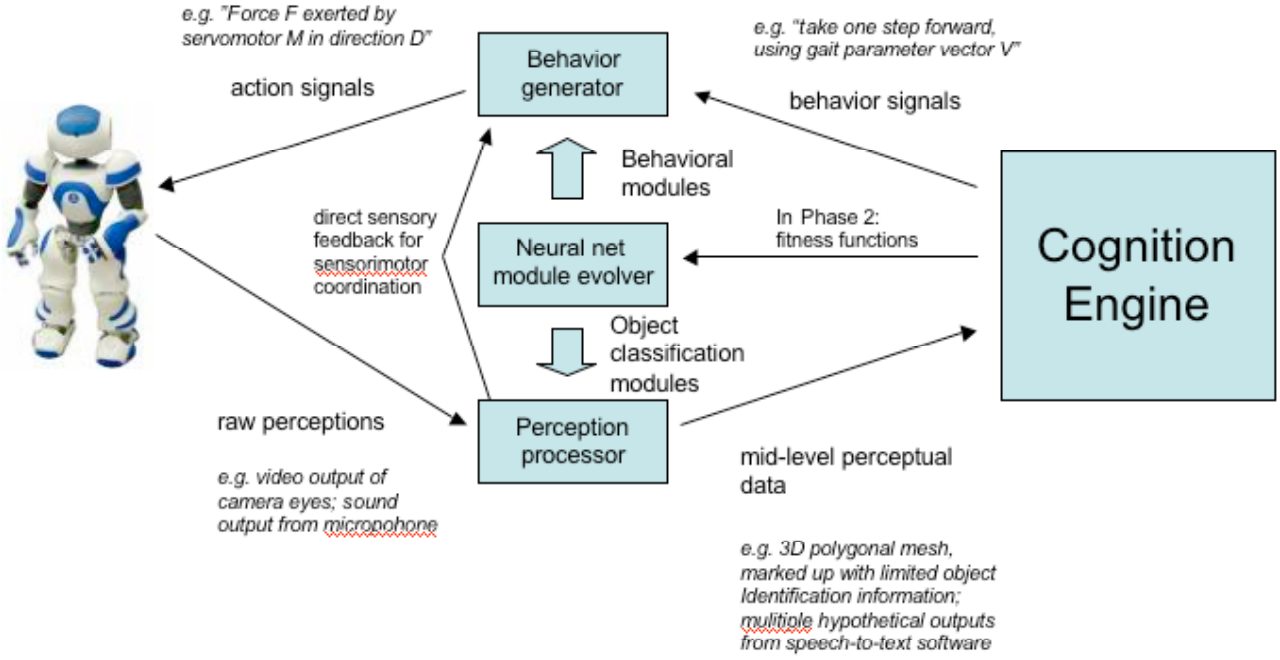

Figure 2. High-level diagram depicting key elements of the proposed integrative architecture for humanoid robotics.

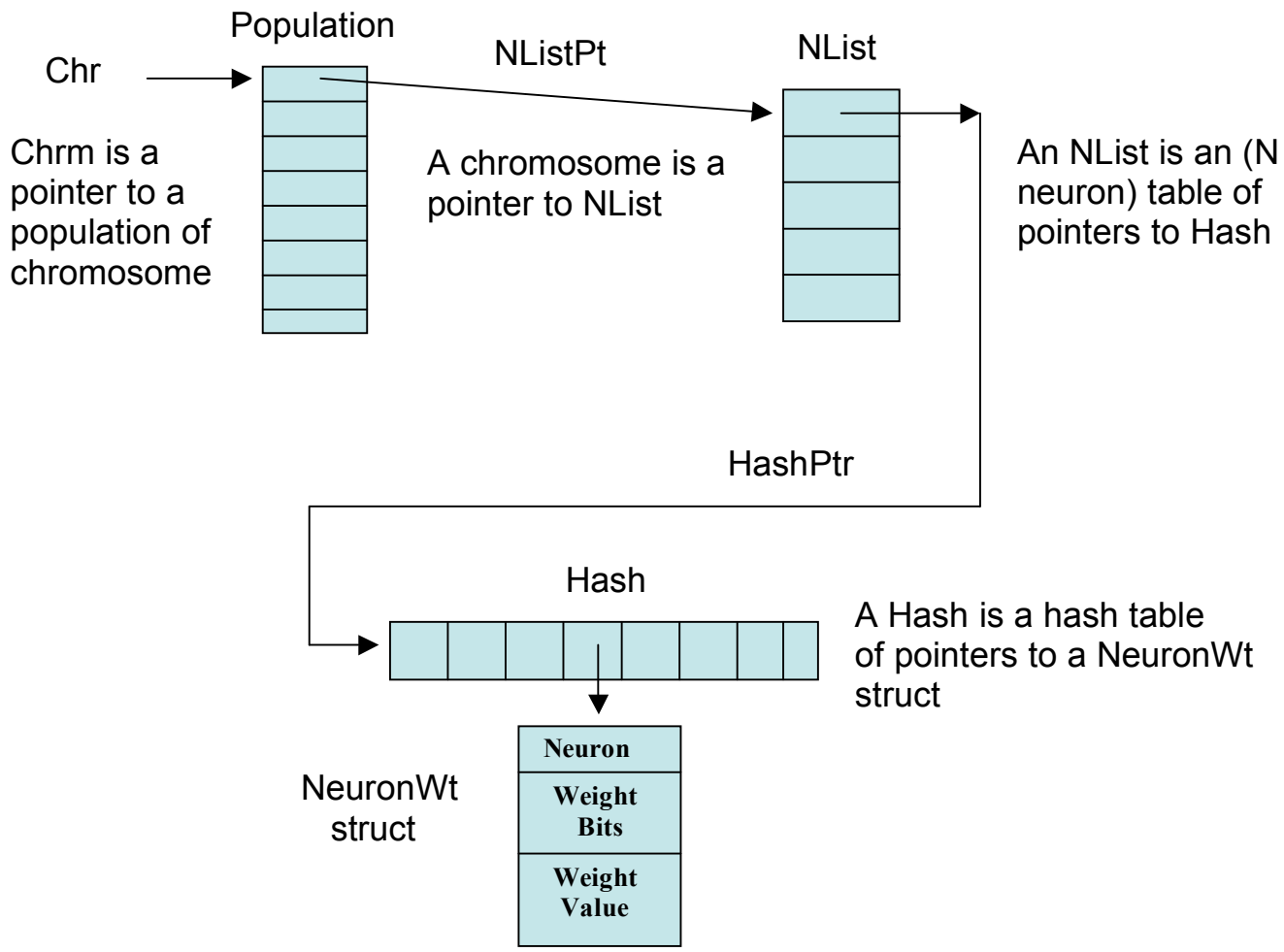

Fig. 3 Data Structures for the Parcone Model, which encodes flexible-topology neural nets in data structures that are well-optimized for evolution on FPGA's as well as via software.

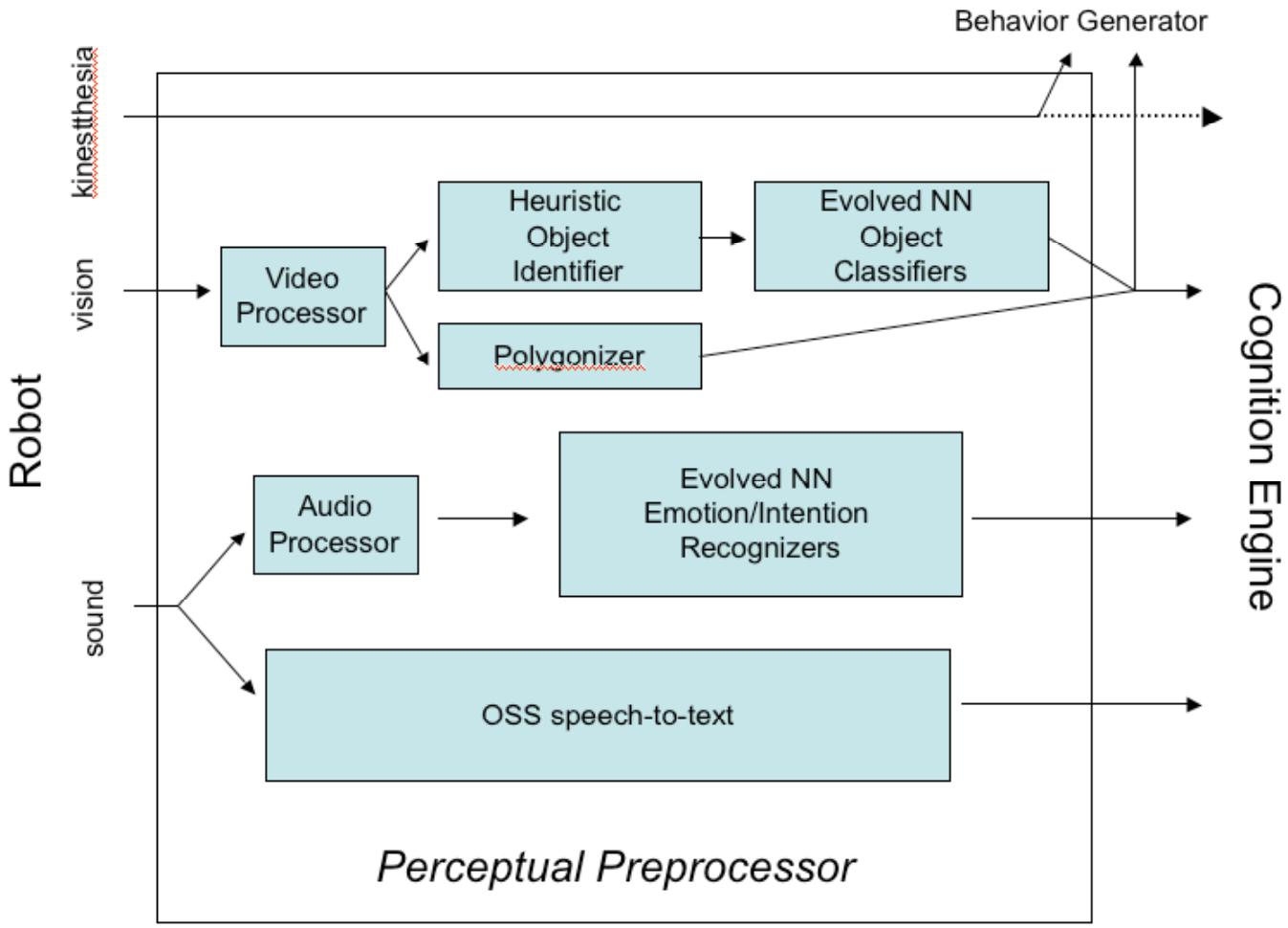

Figure 4. Internals of the Perception Processor

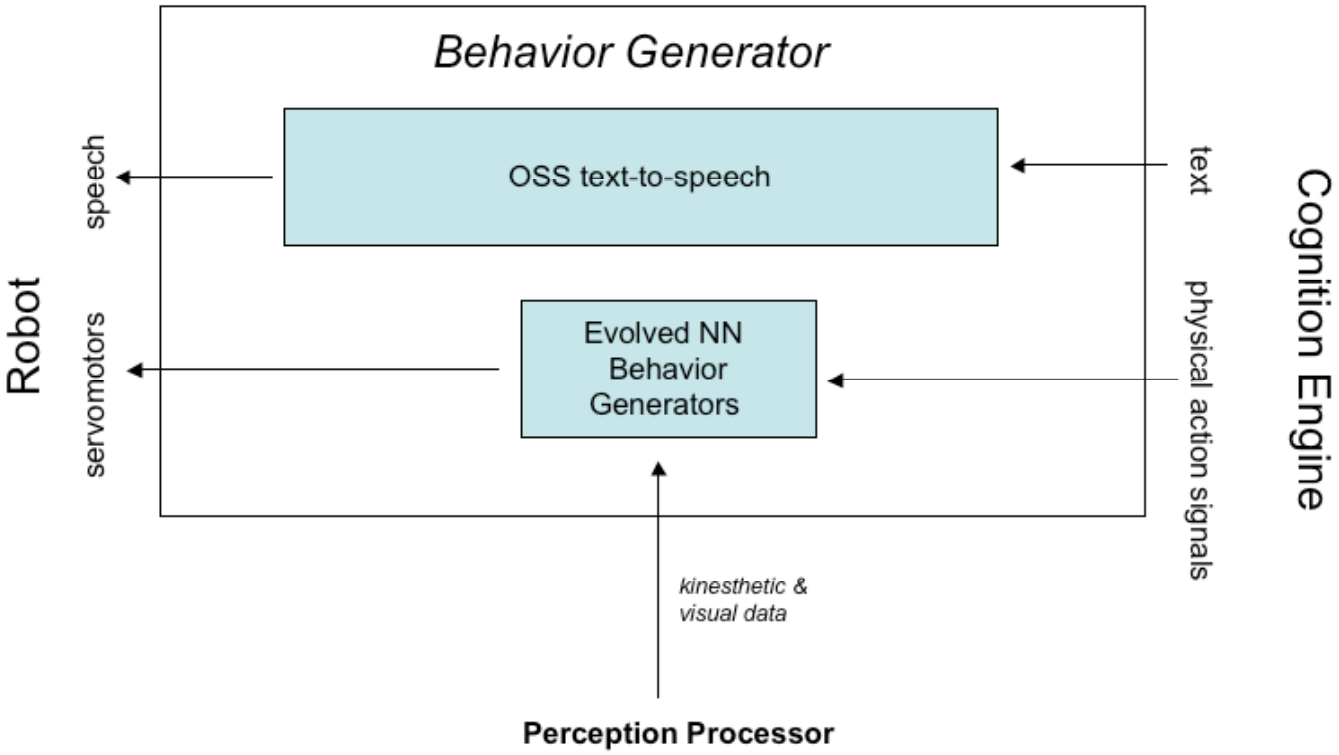

Figure 5. Internals of the Behavior Generator

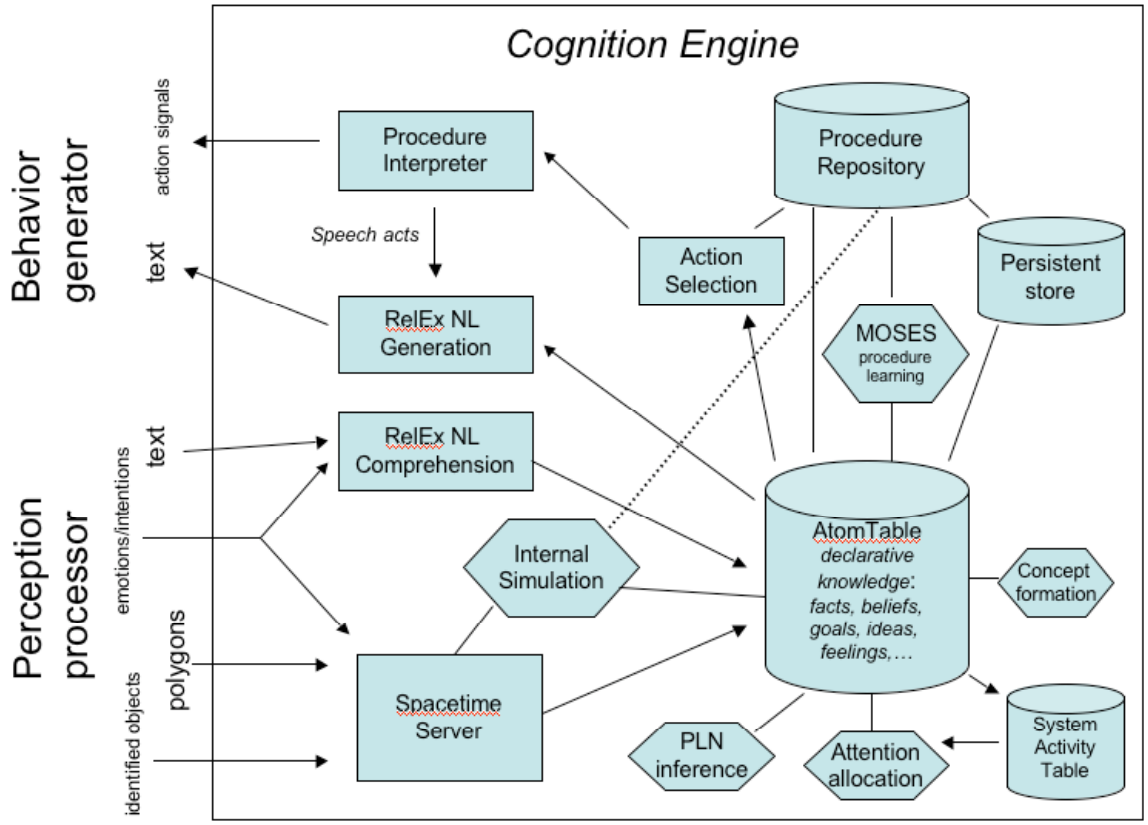

Figure 6. Internals of the Cognitive Engine, according to the Novamente / OpenCog Prime approach

Figure 7. Left: AI-controlled virtual dog in the Second Life, virtual world, learning the behavior "sit." Right: Display of the current emotional and physiological status of an AI-controlled virtual dog in the Multiverse virtual world.

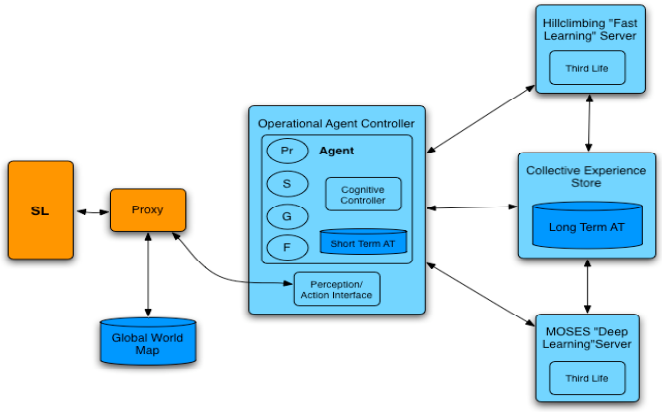

Figure 8. High-level architecture of Novamente Pet Brain, illustrating its connection to the Second Life (SL) virtual world. One aspect is included here that is not discussed in the main text: the use of a collective memory database to store the memories of a large population of virtual pets all together.

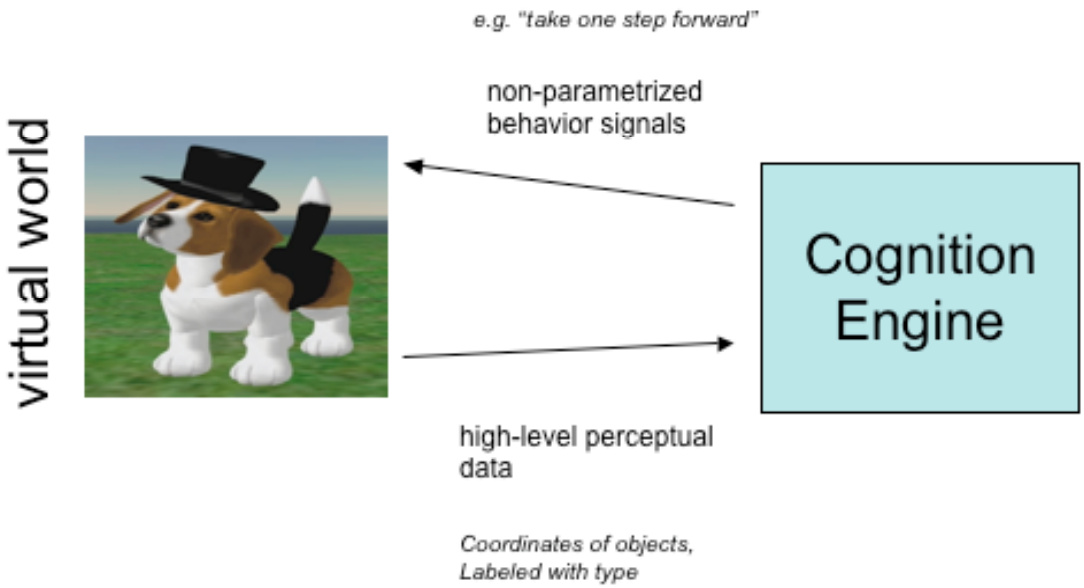

Figure 9. Simplified world-interaction scheme used with current virtual worlds and game engines