OpenPsi: Realizing Dörner's "Psi" Cognitive Model in the OpenCog Integrative AGI Architecture

Zhenhua Cai[1], Ben Goertzel[1,2] and Nil Geisweiller[2]

[1] BLISS Lab, Xiamen University, China [2]Novamente LLC, Rockville MD

Abstract. Dietrich Doerner's "Psi" cognitive model, which was used as the basis for Joscha Bach's MicroPsi AGI system, is expressed in a quite different terminology and conceptual framework from the one normally used to discuss the OpenCog AGI system. However, the two systems are fundamentally conceptually compatible, and we describe here the basis of a realization of the Psi model within OpenCog, which we call "OpenPsi." Currently OpenPsi is being used to control non-player characters in a game world, and application to humanoid robotics is also underway.

1 Introduction

The field of Artificial General Intelligence is remarkably fragmented. There are multiple theoretical frameworks that are often, at least on the surface, incompatible. There are multiple practical AGI architectures, founded on different theoretical bases, with overlapping functionalities and goals; and there are no clear mappings from components or aspects of one architecture into another. To some extent, this fragmentation may be considered a feature rather than a bug. AGI is a young field, and a diverse variety of approaches is healthy and expectable. Sometimes, however, differences in language or focus between the works of different groups in the field, obscure deeper underlying commonalities that could otherwise be profitably exploited. Because of this it behooves us as AGI researchers to aggressively explore possible relationships between our own preferred approaches and those of other researchers.

We present here the early results of one such exploration. The authors are involved with the development of the OpenCogPrime (OCP) AGI architecture [6] within the OpenCog software framework, and when first introduced to Joscha Bach's MicroPsi AGI architecture [2], the two approaches appeared incompatible in major respects. More careful study of MicroPsi, however, and extensive in-person dialogues with Bach, revealed a great number of parallels between the approaches, and led to the conclusion that many key aspects of MicroPsi could in fact be replicated within OpenCog and integrated with OCP.

MicroPsi is conceptually founded on Dietrich Dörner's "Psi," a cognitive model of human and animal intelligence focusing on the role of motivation and emotion in guiding behavior and cognition. The overall Psi theory comprises a comprehensive model of the human brain and mind, in principle encompassing all aspects of human-level general intelligence; but, Psi is typically presented in the context of a relatively simplistic agent choosing behaviors in the world in a manner driven by several well-defined motives (e.g. need for food, water, the avoidance of pain, certainty, competence and affiliation). Psi was originally implemented by Dörner in a helpful and illustrative but somewhat "toy" computer system; MicroPsi embodied Psi in a serious software architecture, enabling more thorough exploration of the model's properties.

2 Motivation and Emotion in Psi

First we outline those aspects of Psi that we have chosen for realization in OpenPsi. As illustrated in Figure 1, the Psi's motivational system begins with **Demands**, which could be the mimic of physiological demands of real animals, such as food, water, sex and so forth, or even fairly abstract demands. Psi theory also specifies three fairly abstract demands: **Competence**, the effectiveness of the agent at fulfilling its Urges; **Certainty**, the confidence of the agent's knowledge; and **Affiliation** ,the acceptance by other agents or social groups. Each demand comes with a certain "target range", which may vary over time, or may change as a system develops. An **Urge** develops when a demand deviates from its target range: the urge seeks to return the demand to its target range.

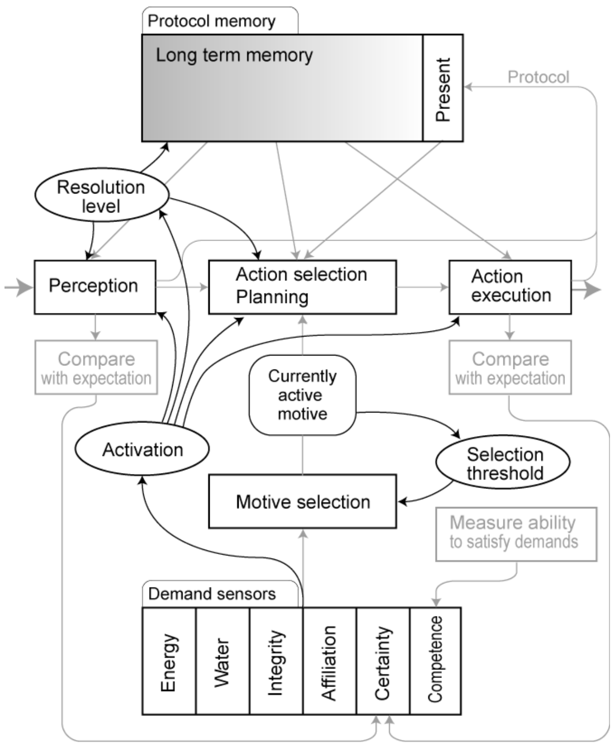

Fig. 1. Influence of Modulators

The most distinctive feature of Psi theory is its perspective on the autonomous choice and regulation of behaviors. It suggests that all goal-directed action have their source in a motive that is connected to an urge, which stands for a physiological, cognitive or social demand [2]. When a positive goal is reached, a demand may be partially or completely fulfilled, which creates a **Pleasure** signal that is used for leaning, by strengthening the associations of the goal with the actions carried out and situations that have led to the fulfillment. While not all the actions are directed immediately hooked to a goal, many actions are carried out to serve an exploratory goal or to avoid an aversive situation. It is also make sense when reaching sub-goals, because in those cases a pleasure signal of competence demand would be created. After finally reaching a consumptive goal, the system may strength all the associations along the chain of actions that has lead to the target goal.

In the Psi model emotion is not considered as an isolated component. Instead it emerges from the whole system, where the process of perception, cognition and action selection interact together. These are accomplished by **modulators**. A modulator is a parameter that characterize how the emotion affects the process of perception, cognition and action selection. Figure 1 shows how the emotion effects these processes via modulators. Dörner currently specifies four modulators: **Activation**, which makes the agent balance between rapid, intensive activity and reflective, cognitive activity; **Resolution level**, which determines how accurately the agent tries to perceive the world; **Certainty**, which stands for the difficulty of the agent tries to achieve definite, certain knowledge; **Selection threshold**, which determines how easily the agent switches between conflicting intentions. Individual agents may differ in their "personalities" because of different settings for the default and ranges of modulators.

3 OpenCog Prime

Now we describe the OCP (OCP) AGI architecture, implemented within the open-source OpenCog AI framework. Conceptually founded on the "patternist" systems theory of intelligence outlined in [5], OCP 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, shown in Figure 2, 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. OCP handles low-level

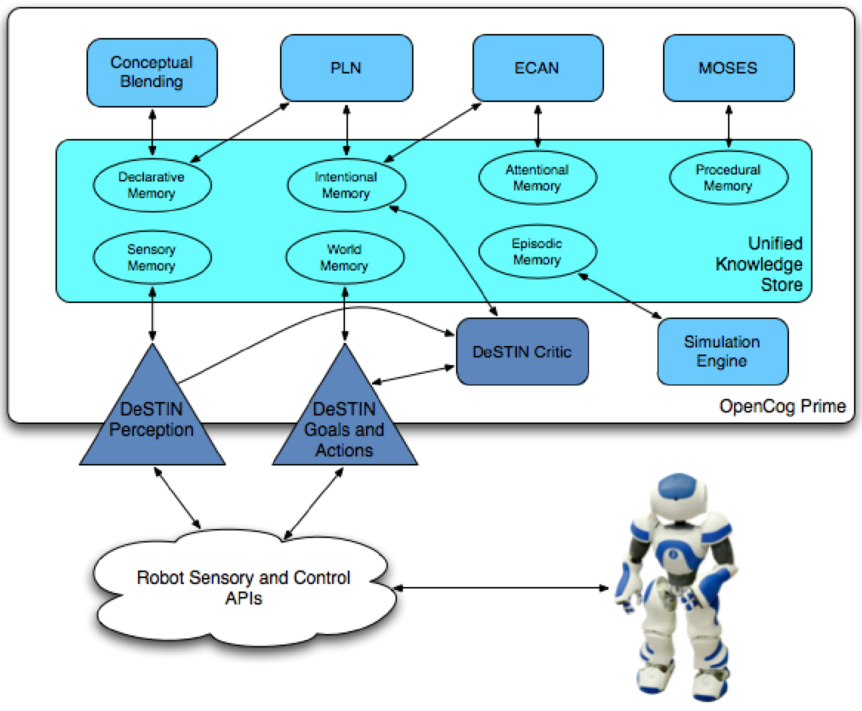

Fig. 2. High-Level OCP Architecture Diagram

perception and action via an extension called OpenCogBot, which integrates a hierarchical temporal memory system, DeSTIN [1].

Memory Types in OpenOCP OCP's memory types are the declarative, procedural, sensory, and episodic memory types that are widely discussed in cognitive neuroscience [12], 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 also indicating 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: 1) conversion between different types of memory is possible, though sometimes computationally costly (e.g. an item of declarative knowledge may with some

Memory Type	**Specific Cognitive Processes**	**General Cognitive Functions**
Declarative	Probabilistic Logic Networks (PLN) [4]; concept blending [3]	pattern creation
Procedural	MOSES (a novel probabilistic evolutionary program learning algorithm) [11]	pattern creation
Episodic	internal simulation engine [7]	association, pattern creation
Attentional	Economic Attention Networks (ECAN) [10]	association, credit assignment
Intentional	probabilistic goal hierarchy refined by PLN and ECAN, structured according to Psi	credit assignment, pattern creation
Sensory	Supplied by DeSTIN integration	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.

effort be interpreted procedurally or episodically, etc.); 2) 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**.

Declarative knowledge representation is handled by a weighted labeled hypergraph called the Atomspace, which consists of multiple types of nodes and links, generally weighted with probabilistic truth values and also attention values (ShortTermImportance (STI) and LongTermImportance values, regulating processor and memory use). ConceptNodes are defined via their links (including logical and associative links), whereas e.g. GroundedPredicateNodes are defined via associated procedures or "schema," which are small programs expressed in a LISP-like language called Combo and stored in a special ProcedureRepository data structure.

OCP's dynamics has both goal-oriented and "spontaneous" aspects. The spontaneous dynamic is driven by the ECAN component, which propagates STI values in a manner reminiscent of an attractor neural network; cognitive processes or knowledge items that get more importance spread to them are then used to trigger action or cognition or to guide perception. 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." 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. Each cognitive schematic is labeled with an uncertain truth value; and cognitive schematics may be incomplete, missing one or two of the terms, which may then be filled in by various cognitive processes (generally in an uncertain way). Goal dynamics also utilizes STI, in that the system's top-level goals are given STI to spend on nominating procedures for execution or to pass to subgoals.

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 [9] 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 [7], using an OpenCog variant called the OpenPetBrain (see `http://novamente.net/example` for some videos of these virtual dogs in action). These agents 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.

4 Psi versus OpenCogPrime

The basic motivation/emotion architecture of Psi, as described above, comprises the basis of OpenPsi, which has been realized within OCP. However, Psi has a number of other aspects that are somewhat different from their OCP analogues, and which have not been carried over to OpenPsi, either because the latter uses a different method to accomplish the same thing, or because it contains an equivalent mechanism. We summarize these here only briefly:

– Representation of knowledge using special 5-neuron clusters called "quads."
– Arrangement of quads into three networks, corresponding to sensation, motor control and motivation
– Use of an algorithm called HyPercept for hypothesis-based perception (a similar principle is used for perception in OCP, but via quite different algorithms)
– Imaginary perceptions are handled via a "mental stage" analogous to OpenCog's internal simulation world.
– Action selection in Psi works based on what are called "triplets," each of which consists of a sensor schema (pre-conditions, "condition schema"; like OCP's "context"), a subsequent motor schema (action, effector; like OCP's "procedure"), and a final sensor schema (post-conditions, expectations; like an OCP predicate or goal).

– Action selection in Psi is carried out via a "Rasmussen ladder" process that first attempts to find an automated routine carrying out the given task, then tries to choose derive a course of action based on rules, then if that fails tries to creatively compose a new behavior based on its background knowledge. These same possibilities exist in OpenCog but are applied in a different way, scheduled by ECAN.
– Psi plans actions using a fairly simple hill-climbing planner. While it's hypothesized that a more complex planner may be needed for advanced intelligence, part of the Psi theory is the hypothesis that most real-life planning an organism needs to do is fairly simple, once the organism has the right perceptual representations and goals. OCP carries out planning integrated with its PLN probabilistic logic engine, a component that has no direct analogue in Psi.

.

Overall, on a high level, the similarities between Psi and OCP are quite strong, including: interlinked declarative, procedural and intentional knowledge structures, represented using neural-symbolic methods; perception via prediction and perception/action integration; action selection via triplets that resemble uncertain, potentially partial production rules. These similarities are what makes the explicit integration of Psi-based motivation and emotion into OpenCog, i.e. OpenPsi, sensible. On the other hand, the deepest difference between the systems lies in the way the inter-operation between different cognitive processes is pursued in the two different approaches. Psi and MicroPsi rely on very simple learning algorithms that are closely tied to the "quad" neurosymbolic knowledge representation, and hence interoperate in a fairly natural way without need for subtle methods of "synergy engineering." OCP uses much more diverse and sophisticated learning algorithms which thus require more sophisticated methods of interoperation in order to achieve cognitive synergy.

5 OpenPsi

We now describe how the basic concepts of the Psi approach to motivation have been incorporated in OCP, constituting "OpenPsi". We give simple examples of each concept, drawn from our use of OpenPsi to help OCP control virtual agents in a game world, playing with blocks and carrying out other activities.

Memory . Psi's memory corresponds to OCP's AtomTable, with associated structures like the ProcedureRepository, the SpaceServer and TimeServer. *Examples:* The knowledge of what blocks look like and the knowledge that tall structures often fall down, go in the AtomTable; specific procedures for picking up blocks of different shapes go in the ProcedureRepository; the layout of a room or a pile of blocks at a specific point in time go in the SpaceServer; the series of events involved in the building-up of a tower are temporally indexed in the TimeServer. In Psi and MicroPsi, these same phenomena are stored in memory in a rather different way, yet the basic Psi motivational dynamics are independent of these representational choices

Demands are GroundedPredicateNodes (GPNs), i.e. Nodes that have their truth value computed at each time by some internal C++ code or some Combo procedure stored in the OpenCog ProcedureRepository. *Examples:* Alertness, perceived novelty, internal novelty, reward from teachers, social stimulus.

Urges (called Ubergoals in OCP) are also GPNs, with their truth values defined in terms of the truth values of the Nodes for corresponding Demands. *Examples:* Now and in the future: stay alert and alive now and in the future; experience and learn new things; get reward from the teachers; enjoy rich social interactions

Importance . The ShortTermImportance of an Ubergoal indicates the urgency of the goal, so if the Demand corresponding to an Ubergoal is within its target range, then the Ubergoal will have zero STI. But all Ubergoals can be given maximal LTI to guarantee they don't get deleted. *Examples:* If the system is in an environment continually providing an adequate level of novelty (according to its Ubergoal), then the Ubergoal corresponding to external novelty with have low STI but high LTI. The system won't expend resources seeking novelty. But then, if the environment becomes more monotonous, the urgency of the external novelty goal will increase, and its STI will increase correspondingly, and resources will begin getting allocated toward improving the novelty of the stimuli received by the agent.

Pleasure is a GPN, and its internal truth value computing program compares the satisfaction of the system's Ubergoals to their expected satisfaction

Goals are Nodes or Links that are on the system's list of goals (the GoalPool); these include but are not restricted to Ubergoals. *Examples:* The Ubergoal of getting reward from teachers might spawn subgoals like "getting reward from Bob" (if Bob is a teacher), or "making teachers smile" or "create surprising new structures" (if the latter often garners teacher reward). The subgoal of "create surprising new structures" might, in the context of a new person entering the agent's environment with a bag of toys, lead to the creation of a subgoal of asking for a new toy of the sort that could be used to help create new structures.

Motive selection as defined in Psi is carried out in OCP by economic attention allocation, which allocates ShortTermImportance to Goal nodes. *Example:* The flow of importance from "Get reward from teachers" to "get reward from Bob" to "make an interesting structure with blocks" is an instance of what Psi calls "motive selection". No action is being taken yet, but choices are being made regarding what specific goals are going to be used to guide action selection.

Action selection Psi's action selection plays the same role as OCP's action selection, with the clarification that in OCP this is a matter of selecting which *procedures* to run, rather than which individual actions to execute. However, this notion exists in Psi as well, which accounts for "automatized behaviors" that are similar to OCP procedures; the main difference here is that in OCP automatized behaviors are the default case. *Example:* If the goal "make an interesting structure with blocks" has a high STI, then it may be used to motivate choice of a procedure to execute, e.g. a procedure that finds an interesting picture or object seen before and approximates it with blocks, or a procedure that randomly constructs something and then filters it based on interestingness. Once a blocks-structure-building procedure is chosen, this procedure may invoke sub-procedures, e.g. those involved with picking up and positioning particular blocks.

Planning in Psi is carried out via various OCP learning processes, including PLN with special control mechanisms plus procedure learning methods like MOSES or hillclimbing. *Example:* If the agent has decided to build a blocks structure emulating a pyramid (which it saw in a picture), and it knows how to manipulate and position individual blocks, then it must figure out a procedure for carrying out individual-block actions that will result in production of the pyramid. In this case, a very inexperienced agent might use MOSES or hillclimbing and "guidedly-randomly" fiddle with different construction procedures until it hit on something workable. A slightly more experienced agent would use reasoning based on prior structures it had built, to figure out a rational plan (like: "start with the base, then iteratively pile on layers, each one slightly smaller than the previous.")

Modulators are system parameters which may be represented in OpenCog by PredicateNodes, and which must be incorporated appropriately in the dynamics of various MindAgents, e.g.

- *activation* affects action selection. For instance this may be effected by a process that, each cycle, causes a certain amount of STI to pass to schema satisfying certain properties (those involving physical action, or terminating rapidly). The amount of currency passed in this way would be proportional to the *activation*
- *resolution level* affects perception schema and MindAgents, causing them to expend less effort in processing perceptual data
- *certainty* affects inference and pattern mining and concept creation processes, causing them to place less emphasis on certainty in guiding their activities, i.e. to be more accepting of uncertain conclusions.
- *selection threshold* may be used to effect a process that, each cycle, causes a certain amount of STI (proportional to the selection threshold) to pass to the Goal Atoms that were wealthiest at the previous cycle.

6 Conclusion

We have described here the logic via which the motivation, emotion and action selection aspects of the Psi cognitive model have been integrated into the OpenCog system, for use within the integrative OCP cognitive model. While this may appear straightforward as laid out, one should not underestimate the difficulties in reconciling the conceptual frameworks of two AGI architectures with very different roots.

The OpenPsi system resultant from this integration is currently being used to control a virtual dog in a virtual world, broadly similarly to the OpenCog application described in [7], but with richer functionality, as will be described in subsequent publications. In another project, OpenCog is being used to control a humanoid (Nao) robot, and it is anticipated that OpenPsi will play a key role in this as well. Rigorous evaluation of the contribution of the OpenPsi in particular, in this sort of application, is not trivial, because what really matters (and what is easier to measure) is the overall intelligence of the virtual or robotic agent. Qualitatively, however, we have already found that it is easier to create agents with realistic-seeming motivational and emotional behavior using OpenPsi than using the previous OpenCog personality/behavior rule engine as described in [8].

References

1. Arel, I., Rose, D., Coop, R.: Destin: A scalable deep learning architecture with application to high-dimensional robust pattern recognition. Proc. AAAI Workshop on Biologically Inspired Cognitive Architectures (2009)
2. Bach, J.: Principles of Synthetic Intelligence. Oxford University Press (2009)
3. Fauconnier, G., Turner, M.: The Way We Think: Conceptual Blending and the Mind's Hidden Complexities. Basic (2002)
4. Goertzel, B., M. Ikl, I.G., Heljakka, A.: Probabilistic Logic Networks. Springer (2008)
5. Goertzel, B.: The Hidden Pattern. Brown Walker (2006)
6. Goertzel, B.: Opencog prime: A cognitive synergy based architecture for embodied artificial general intelligence. In: ICCI 2009, Hong Kong (2009)
7. Goertzel, B., Et Al, C.P.: 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 (2008)
8. Goertzel, B., Pennachin, C.: The collective pet unconscious: Balancing intelligence and individuality in populations of learning-enabled virtual pets. In: The Reign of Catz and Dogz Symposium, ACM-CHI, Boston (2009)
9. Goertzel, B., Pinto, H., Pennachin, C., Goertzel, I.F.: 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 (2006)
10. Goertzel, B., Pitt, J., Ikle, M., Pennachin, C., Liu, R.: Glocal memory: a design principle for artificial brains and minds. Neurocomputing (Apr 2010)
11. Looks, M.: Competent Program Evolution. PhD Thesis, Computer Science Department, Washington University (2006)
12. Tulving, E., Craik, R.: The Oxford Handbook of Memory. Oxford U. Press (2005)