CogPrime: An Integrative Architecture for Embodied Artificial General Intelligence

Ben Goertzel

October 2, 2012

Abstract

The CogPrime architecture for embodied AGI is overviewed, covering the core architecture and algorithms, the underlying conceptual motivations, and the emergent structures, dynamics and functionalities expected to arise in a completely implemented CogPrime system once it has undergone appropriate experience and education. A qualitative argument is sketched, in favor of the assertion that a completed CogPrime system, given a modest amount of experience in an embodiment enabling it to experience a reasonably rich human-like world, will give rise to human-level general intelligence (with significant difference from humans, and with potential for progress beyond this level).

1 Introduction

This is a lengthy paper with a substantial ambition: to overview CogPrime, a conceptual and technical design for a thinking machine, a software program capable of the same qualitative sort of general intelligence as human beings. Given the uncertainties attendant on all research, we cannot know for sure how far the CogPrime design will be able to take us; but it seems plausible that once fully implemented, tuned and tested, it will be able to achieve general intelligence at the human level and in some respects perhaps beyond. CogPrime is described in more detail in a forthcoming book titled *Building Better Minds: The CogPrime Architecture for Artificial General Intelligence* [GPGtOT13], which exceeds 1000 pages including appendices; the goal of this paper is to outline some of the key points in a more compact format.

To allay potential confusion we offer two caveats from the outset. First, CogPrime is not a model of human neural or cognitive structure or activity. It draws heavily on knowledge about human intelligence, especially cognitive psychology; but it also deviates from the known nature of human intelligence in many ways, with a goal of providing maximal humanly-meaningful general intelligence using available computer hardware. Second, CogPrime is not proposed as the one and only holy grail path to advanced AGI. We feel confident there are multiple possible paths to advanced AGI, and that in following any of these paths, multiple theoretical and practical lessons will be learned, leading to modifications of the ideas developed and possessed along the early stages of the path. The goal here is to articulate **one** path that we believe makes sense to follow, one overall design that we believe can work, for achieving general intelligence that is qualitatively human-level and in many respects human-like, without emulating human neural or cognitive function in detail.

1.1 AI versus AGI

An outsider to the AI field might think this sort of paper commonplace in the research literature, but insiders know that's far from the truth. The field of Artificial Intelligence (AI) was founded in the mid 1950s with the aim of constructing "thinking machines" - that is, computer systems with human-like general intelligence, including humanoid robots that not only look but act and think with intelligence equal to and ultimately greater than that of human beings. But in the intervening years, the field has drifted far from its ambitious roots, and this book represents part of a movement aimed at restoring the initial goals of the AI field, but in a manner powered by new tools and new ideas far beyond those available half a century ago.

After the first generation of AI researchers found the task of creating human-level AGI very difficult given the technology of their time, the AI field shifted focus toward what Ray Kurzweil [Kur06] has called

"narrow AI" – the understanding of particular specialized aspects of intelligence; and the creation AI systems displaying intelligence regarding specific tasks in relatively narrow domains. In recent years, however, the situation has been changing. More and more researchers have recognized the necessity – and feasibility – of returning to the original goals of the field.

In the decades since the 1950s, cognitive science and neuroscience have taught us a lot about what a cognitive architecture needs to look like to support roughly human-like general intelligence. Computer hardware has advanced to the point where we can build distributed systems containing large amounts of RAM and large numbers of processors, carrying out complex tasks in real time. The AI field has spawned a host of ingenious algorithms and data structures, which have been successfully deployed for a huge variety of purposes.

Due to all this progress, increasingly, there has been a call for a transition from the current focus on highly specialized "narrow AI" problem solving systems, back to confronting the more difficult issues of "human level intelligence" and more broadly "artificial general intelligence (AGI)." Recent years have seen a growing number of special sessions, workshops and conferences devoted specifically to AGI, including the annual BICA (Biologically Inspired Cognitive Architectures) Conference, and the international AGI conference series (AGI-08 , AGI-09, AGI-10, AGI-11). And, even more exciting, there are a number of contemporary R&D projects focused directly and explicitly on AGI (sometimes under the name "AGI", sometimes using related terms such as "Human Level Intelligence").

In spite of all this progress, however, no one has yet clearly articulated a detailed, systematic design for an AGI, with potential to yield general intelligence at the human level and ultimately beyond. Perhaps the most comprehensive attempts in this direction have been the works of Stan Franklin [BF09] and Joscha Bach [Bac09], or the more classical SOAR [Lai12] and ACT-R [And96] architectures. While we feel there is much to be learned from these designs, we also feel they have significant shortcomings and lacunae alongside their considerable strengths. Detailed discussion and comparison of these and other alternative AGI approaches will not be presented here, as the paper is long enough already, but are given in the above-mentioned book that constitutes a greatly expanded version of this paper [GPGtOT13].

1.2 What's the Secret Sauce?

There is no consensus on why all the related technological and scientific progress mentioned above has not yet yielded AI software systems with human-like general intelligence. However, we hypothesize that the core reason boils down to the following three points:

- Intelligence depends on the emergence of certain high-level structures and dynamics across a system's whole knowledge base;
- We have not discovered any one algorithm or approach capable of yielding the emergence of these structures;
- Achieving the emergence of these structures within a system formed by integrating a number of different AI algorithms and structures is tricky. It requires careful attention to the manner in which these algorithms and structures are integrated; and so far the integration has not been done in the correct way.

The human brain appears to be an integration of an assemblage of diverse structures and dynamics, built using common components and arranged according to a sensible cognitive architecture. However, its algorithms and structures have been honed by evolution to work closely together – they are very tightly inter-adapted, in somewhat the same way that the different organs of the body are adapted to work together. Due their close interoperation they give rise to the overall systemic behaviors that characterize human-like general intelligence. We believe that the main missing ingredient in AI so far is **cognitive synergy**: the fitting-together of different intelligent components into an appropriate cognitive architecture, in such a way that the components richly and dynamically support and assist each other, interrelating very closely in a similar manner to the components of the brain or body and thus giving rise to appropriate emergent structures and dynamics. Which leads us to one of the central hypotheses underlying the CogPrime approach to AGI: that **the cognitive synergy ensuing from integrating multiple symbolic and subsymbolic learning**

and memory components in an appropriate cognitive architecture and environment, can yield robust intelligence at the human level and ultimately beyond.

The reason this sort of intimate integration has not yet been explored much is that it's difficult on multiple levels, requiring the design of an architecture and its component algorithms with a view toward the structures and dynamics that will arise in the system once it is coupled with an appropriate environment. Typically, the AI algorithms and structures corresponding to different cognitive functions have been developed based on divergent theoretical principles, by disparate communities of researchers, and have been tuned for effective performance on different tasks in different environments. Making such diverse components work together in a truly synergetic and cooperative way is a tall order, yet we believe that this – rather than some particular algorithm, structure or architectural principle – is the "secret sauce" needed to create human-level AGI based on technologies available today.

1.3 What Kind of "Intelligence" is CogPrime Aimed At?

We have mentioned "intelligence" frequently in the preceding paragraphs, but haven't specified precisely what we mean by it. In fact, a host of different definitions of intelligence have been proposed in the AGI, narrow AI, psychology, engineering and philosophy communities; Legg and Hutter [LH07] have enumerated over 70. The CogPrime design is not particularly tied to any of these; instead, it was primarily motivated by a commonsensical interpretation of intelligence, comprising a mixture of theoretical general problem solving capability, and the practical ability to display the same sorts of intelligence as humans do. However, CogPrime is consistent with a number of different recognized definitions of general intelligence.

There are psychological definitions of intelligence (e.g. the g-factor), based on the ability to carry out particular kinds of common human problem-solving tasks effectively [KWRK05]. The design intention of CogPrime is that a fully implemented and reasonably experienced system would be able to do well at these sorts of tasks.

There are pragmatic tests for measuring whether an AGI has achieved human-level, human-like general intelligence. In the paper "Mapping the Landscape of Human-Level AI" [AAB+11], resultant from the 2009 AGI Roadmap Workshop, the author and a number of co-authors discussed several examples of practical goals plausibly characterizing "human level AGI", e.g.

- *Turing Test*: the classic, involving passing as a human being in an everyday conversation
- *Virtual World or Telerobotic Turing Test* : pass as a human being, in a conversation involving controlling a virtual world avatar or a robot, alongside verbal interaction
- *Online University Test*: attend an online university just like a human student, and graduate
- *Physical University Test*: control a robot that attends university just like a human student, and graduates
- *Artificial Scientist Test*: write and publish original science papers, based on ideas conceived by the AI due to its own reading of the literature

Again, while we don't necessarily consider all of these as useful for motivating the early steps of AGI research, we intend that a fully realized CogPrime system would be able to succeed at them.

There are mathematical definitions of general intelligence, involving formalizations of the idea that an intelligence should be judged via its average capability to achieve goals in environments (where the average is weighted so that more complex goals and environments get less weight) [Goe10b]. CogPrime is also intended to display intelligence in this sense, but with the caveat that a complete and mature CogPrime functioning as intended, would perform better according to such a measure if the goals and environments similar to everyday human goals and environments were weighted more highly.

A variant of this mathematical approach to intelligence measures not only the ability of a system to achieve certain goals in certain environments, but also the amount of space and time resources expended in doing so [Goe10b]. A system that can do the same things using fewer resources would be rated more intelligent. Since CogPrime is not a theoretical design like AIXI [Hut05], but one intended for real-world near-term implementation, it is intended to be intelligent in this sense as well. That is, informally speaking:

CogPrime is intended as "wholly general in principle", but, able to carry out certain kinds of intelligence-requiring tasks more rapidly and using fewer computational resources. And the kinds of tasks it can do more easily, are biased by design to reflect the nature of the human everyday world.

There is a school of thought that defines intelligence via adaptiveness: the capability of a system to adapt to its environment effectively [Wan06]. CogPrime is also intended to display intelligence in this sense. Indeed, it seems to us that achieving goals in environments requires adaptation to these environments, and vice versa; so that qualitatively the various approaches to defining intelligence are largely getting at the same thing.

All in all, we pragmatically conceive of general intelligence as "the ability to achieve complex goals in complex environments, using limited resources"; and we accept that any real-world general intelligence is going to be more intelligent with respect to some sorts of goals and environments than others. CogPrime has been designed with careful attention toward human-relevant goals and environment, yet is also expected to have a particular set of intellectual strengths and weaknesses different from that of humans.

1.4 Key Claims

We conclude this Introduction with a systematic list of some of the key claims to be argued for here. Not all the terms and ideas in these claims will be familiar to the reader in their technical details, but we hope their qualitative sense will be reasonably clear nonetheless. This list of claims will be revisited toward the end of the paper, where we will look back at the ideas and arguments that have been put forth in favor of them, in the intervening sections.

In essence this is a list of claims such that, if the reader accepts these claims, they should probably accept that the CogPrime approach to AGI is a viable one. On the other hand if the reader rejects one or more of these claims, they may find one or more aspects of CogPrime unacceptable for a related reason:

1. General intelligence (at the human level and ultimately beyond) can be achieved via creating a computational system that uses much of its resources seeking to achieve its goals, via using perception and memory to predict which actions will achieve its goals in the contexts in which it finds itself.

2. To achieve general intelligence in the context of human-intelligence-friendly environments and goals using feasible computational resources, it's important that an AGI system can handle different kinds of memory (declarative, procedural, episodic, sensory, intentional, attentional) in customized but interoperable ways.

3. Cognitive synergy: It's important that the cognitive processes associated with different kinds of memory can appeal to each other for assistance in overcoming bottlenecks in a manner that enables each cognitive process to act in a manner that is sensitive to the particularities of each others' internal representations, and that doesn't impose unreasonable delays on the overall cognitive dynamics.

4. As a general principle, neither purely localized nor purely global memory is sufficient for general intelligence under feasible computational resources; "glocal" memory will be required.

5. To achieve human-like general intelligence, it's important for an intelligent agent to have sensory data and motoric affordances that roughly emulate those available to humans. We don't know exactly how close this emulation needs to be, which means that our AGI systems and platforms need to support fairly flexible experimentation with virtual-world and/or robotic infrastructures.

6. To work toward adult human-level, roughly human-like general intelligence, one fairly easily comprehensible path is to use environments and goals reminiscent of human childhood, and seek to advance one's AGI system along a path roughly comparable to that followed by human children.

7. It is most effective to teach an AGI system aimed at roughly human-like general intelligence via a mix of spontaneous learning and explicit instruction, and to instruct it via a combination of imitation, reinforcement and correction, and a combination of linguistic and nonlinguistic instruction.

8. One effective approach to teaching an AGI system human language is to supply it with some in-built linguistic facility, in the form of rule-based and statistical-linguistics-based NLP systems, and then allow it to improve and revise this facility based on experience.

9. An AGI system with adequate mechanisms for handling the key types of knowledge mentioned (in item 2) above, and the capability to explicitly recognize large-scale pattern in itself, should, **upon sustained interaction with an appropriate environment in pursuit of appropriate goals**, emerge a variety of complex structures in its internal knowledge network, including, but not limited to:

    - a hierarchical network, representing both a spatiotemporal hierarchy and an approximate "default inheritance" hierarchy, cross-linked;
    - a heterarchical network of associativity, roughly aligned with the hierarchical network;
    - a self network which is an approximate micro image of the whole network;
    - inter-reflecting networks modeling self and others, reflecting a "mirrorhouse" design pattern [GASP08].

10. Given the strengths and weaknesses of current and near-future digital computers,

    (a) a (loosely) neural-symbolic network is a good representation for directly storing many kinds of memory, and interfacing between those that it doesn't store directly;

    (b) Uncertain logic is a good way to handle declarative knowledge. To deal with the problems facing a human-level AGI, an uncertain logic must integrate imprecise probability and fuzziness with a broad scope of logical constructs. PLN is one good realization.

    (c) Programs are a good way to represent procedures (both cognitive and physical-action, but perhaps not including low-level motor-control procedures).

    (d) Evolutionary program learning is a good way to handle difficult program learning problems. Probabilistic learning on normalized programs is one effective approach to evolutionary program learning. MOSES is one good realization of this approach.

    (e) Multistart hill-climbing, with a strong Occam prior, is a good way to handle relatively straightforward program learning problems.

    (f) Activation spreading and Hebbian learning comprise a reasonable way to handle attentional knowledge (though other approaches, with greater overhead cost, may provide better accuracy and may be appropriate in some situations).

        - Artificial economics is an effective approach to activation spreading and Hebbian learning in the context of neural-symbolic networks;
        - ECAN is one good realization of artificial economics;
        - A good trade-off between comprehensiveness and efficiency is to focus on two kinds of attention: processor attention (represented in CogPrime by ShortTermImportance) and memory attention (represented in CogPrime by LongTermImportance).

    (g) Simulation is a good way to handle episodic knowledge (remembered and imagined). Running an internal world simulation engine is an effective way to handle simulation.

    (h) Hybridization of one's integrative neural-symbolic system with a spatiotemporally hierarchical deep learning system is an effective way to handle representation and learning of low-level sensorimotor knowledge. DeSTIN is one example of a deep learning system of this nature that can be effective in this context.

    (i) One effective way to handle goals is to represent them declaratively, and allocate attention among them economically. CogPrime 's PLN/ECAN based framework for handling intentional knowledge is one good realization.

11. It is important for an intelligent system to have some way of recognizing large-scale patterns in itself, and then embodying these patterns as new, localized knowledge items in its memory (a dynamic called the "cognitive equation" in [Goe94]),. Given the use of a neural-symbolic network for knowledge representation, a graph-mining based "map formation" heuristic is one good way to do this.

12. Occam's Razor: Intelligence is closely tied to the creation of procedures that achieve goals in environments *in the simplest possible way*. Each of an AGI system's cognitive algorithms should embody a simplicity bias in some explicit or implicit form.

13. An AGI system, if supplied with a commonsensically ethical goal system and an intentional component based on rigorous uncertain inference, should be able to reliably achieve a much higher level of commonsensically ethical behavior than any human being.

14. Once sufficiently advanced, an AGI system with a logic-based declarative knowledge approach and a program-learning-based procedural knowledge approach should be able to radically self-improve via a variety of methods, including supercompilation and automated theorem-proving.

2 CogPrime and OpenCog

Many of the technical details of the CogPrime design have been previously presented online in a wikibook [Goe10a]; and the basic ideas of the design have been presented briefly in a series of conference papers [GPPG06, Goe09c]. But the overall design has not been presented in a coherent and systematic way before this paper and the associated book [GPGtOT13].

CogPrime is closely allied with the OpenCog open-source AI software framework. But the two are not synonymous. OpenCog is a more general framework, suitable for implementation of a variety of specialized AI applications as well as, potentially, alternate AGI designs. And CogPrime could potentially be implemented other than within the OpenCog framework. The particular implementation of CogPrime in OpenCog is called OpenCogPrime. OpenCog was designed with the purpose, alongside others, of enabling efficient, scalable implementation of the full CogPrime design. [1]

2.1 Current and Prior Applications of OpenCog

To give greater understanding regarding the practical platform for current work aimed at realizing CogPrime, we now briefly discuss some of the practicalities of work done with the OpenCog system that currently implements parts of the CogPrime architecture.

OpenCog, the open-source software framework underlying the "OpenCogPrime" (currently partial) implementation of the CogPrime architecture, has been used for commercial applications in the area of natural language processing and data mining. For instance, see [GPPG06] where OpenCogPrime's PLN reasoning and RelEx language processing are combined to do automated biological hypothesis generation based on information gathered from PubMed abstracts. [Loo06] describes the use of OpenCog's MOSES component for biological data analysis; this use has been extended considerably in a variety of unpublished commercial applications since that point, in domains such as financial prediction, genetics, marketing data analysis and natural language processing. Most relevantly to the present work, OpenCog has also been used to control virtual agents in virtual worlds [GEA08].

Prototype work done during 2007-2008 involved using an OpenCog variant called the OpenPetBrain to control virtual dogs in a virtual world (see Figure 1 for a screenshot of an OpenPetBrain-controlled virtual dog). While these OpenCog virtual dogs did not display intelligence closely comparable to that of real dogs (or human children), they did 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

[1] This brings up a terminological note: At several places in this Volume and the next we will refer to the current CogPrime or OpenCog implementation; in all cases this refers to OpenCog as of mid 2012. We realize the risk of mentioning the state of our software system at time of writing: for future readers this may give the wrong impression, because if our project goes well, more and more of CogPrime will get implemented and tested as time goes on (the system is under active development at time of writing). However, not mentioning the current implementation at all seems an even worse course to us, since we feel readers will be interested to know which of our ideas – at time of writing – have been honed via practice and which have not. Online resources such as `http://opencog.org` may be consulted by readers curious about the current state of the main OpenCog / OpenCogPrime implementation; though in future forks of the code may be created, or other systems may be built using some or all of the ideas in this book, etc.

- spontaneous exploration of their world, remembering their experiences and using them to bias future learning and linguistic interaction

Figure 1: Screenshot of OpenCog-controlled virtual dog

One current OpenCog initiative [GPC+11] involves extending the virtual dog work via using OpenCog to control virtual agents in a game world inspired by the game Minecraft. These agents are initially specifically concerned with achieving goals in a game world via constructing structures with blocks and carrying out simple English communications. Representative example tasks would be:

- Learning to build steps or ladders to get desired objects that are high up
- Learning to build a shelter to protect itself from aggressors
- Learning to build structures resembling structures that its shown (even if the available materials are a bit different)
- Learning how to build bridges to cross chasms

Of course, the AI significance of learning tasks like this all depends on what kind of feedback the system is given, and how complex its environment is. It would be relatively simple to make an AI system do things like this in a highly specialized way, but that is not the intent of the project – the goal is to have the system learn to carry out tasks like this using general learning mechanisms and a general cognitive architecture, based on embodied experience and only scant feedback from human teachers. If successful, this will provide an outstanding platform for ongoing AGI development, as well as a visually appealing and immediately meaningful demo for OpenCog.

A few of the specific tasks that are the focus of this project teams current work at time of writing include:

- Watch another character build steps to reach a high-up object
- Figure out via imitation of this that, in a different context, building steps to reach a high up object may be a good idea
- Also figure out that, if it wants a certain high-up object but there are no materials for building steps available, finding some other way to get elevated will be a good idea that may help it get the object (including e.g. building a ladder, or asking someone tall to pick it up, etc.)

- Figure out that, if the character wants to hide its valued object from a creature much larger than it, it should build a container with a small hole that the character can get through, but the creature cannot

2.2 Transitioning from Virtual Agents to a Physical Robot

In 2009-2010, preliminary experiments were conducted using OpenCog to control a Nao robot [GdG08]. These involved hybridizing OpenCog with a separate subsystem handling low-level perception and action. This hybridization was accomplished in an extremely simplistic way, however. How to do this right is a topic treated in detail in [GPGtOT13] and [Goe12] and only briefly touched here.

We suspect that reasonable level of capability will be achievable by simply interposing DeSTIN [ARC09] (or some other reasonably capable "hierarchical temporal memory" type sensorimotor system) as a perception/action "black box" between OpenCog and a robot. However, we also suspect that to achieve robustly intelligent robotics we must go beyond this approach, and connect robot perception and actuation software with OpenCogPrime in a "white box" manner that allows intimate dynamic feedback between perceptual, motoric, cognitive and linguistic functions. We suspect this may be achievable, for example, via the creation and real-time utilization of links between the nodes in CogPrime 's and DeSTIN's internal networks.

3 Philosophical Background

The creation of advanced AGI systems is an engineering endeavor, whose achievement will require significant input from science and mathematics; and also, we believe, guidance from philosophy. Having an appropriate philosophy of mind certainly is no guarantee of creating advanced AGI system; philosophy only goes so far. However, having a badly inappropriate philosophy of mind may be a huge barrier in the creation of AGI systems. For instance, we believe that philosophical views holding that

- the contents of a mind are best understood purely as a set of logical propositions, terms or predicates; or that
- brains and other intelligence-substrate systems are necessarily so complex and emergence-dependent that it's hopeless to try to understand how they represent any particular thing, or carry out any particular action

are particularly poorly suited to guide AGI development, and are likely to directly push adherents in the wrong directions AGI-design-wise.

The development of the CogPrime design has been substantially guided by a philosophy of mind called "patternism" [Goe06]. This guidance should not be overstated; CogPrime is an integrative design formed via the combination of a number of different philosophical, scientific and engineering ideas, and the success or failure of the design doesn't depend on any particular philosophical understanding of intelligence. In that sense, the more abstract notions summarized in this section should be considered "optional" rather than critical in a CogPrime context. However, due to the core role patternism has played in the development of CogPrime , understanding a few things about general patternist philosophy will be helpful for understanding CogPrime , even for those readers who are not philosophically inclined. Those readers who *are* philosophically inclined, on the other hand, are urged to read *The Hidden Pattern* [Goe06] and then interpret the particulars of CogPrime in this light.

The patternist philosophy of mind is a general approach to thinking about intelligent systems. It is based on the very simple premise that *mind is made of pattern* – and that a mind is a system for recognizing patterns in itself and the world, critically including patterns regarding which procedures are likely to lead to the achievement of which goals in which contexts.

Pattern as the basis of mind is not in itself is a very novel idea; it is present, for instance, in the 19'th-century philosophy of Charles Peirce [Pei34], in the writings of contemporary philosophers Daniel Dennett [Den91] and Douglas Hofstadter [Hof79, Hof96], in Benjamin Whorf's [Who64] linguistic philosophy and Gregory Bateson's [Bat79] systems theory of mind and nature. Bateson spoke of the Metapattern: "that it is pattern which connects." In our prior writings on philosophy of mind, an effort has been made to pursue

this theme more thoroughly than has been done before, and to articulate in detail how various aspects of human mind and mind in general can be well-understood by explicitly adopting a patternist perspective. [2]

In the patternist perspective, "pattern" is generally defined as "representation as something simpler." Thus, for example, if one measures simplicity in terms of bit-count, then a program compressing an image would be a pattern in that image. But if one uses a simplicity measure incorporating run-time as well as bit-count, then the compressed version may or may not be a pattern in the image, depending on how one's simplicity measure weights the two factors. This definition encompasses simple repeated patterns, but also much more complex ones. While pattern theory has typically been elaborated in the context of computational theory, it is not intrinsically tied to computation; rather, it can be developed in any context where there is a notion of "representation" or "production" and a way of measuring simplicity. One just needs to be able to assess the extent to which $\mathcal{P}$ represents or produces X, and then to compare the simplicity of $\mathcal{P}$ and X; and then one can assess whether $\mathcal{P}$ is a pattern in X. A formalization of this notion of pattern is given in [Goe06]; the crux is simply

Definition 1 *Given a metric space (M, d), and two functions $c : M \to [0, \infty]$ (the "simplicity measure") and $F : M \to M$ (the "production relationship"), we say that $\mathcal{P} \in M$ is a* **pattern** *in $X \in M$ to the degree*

$$\iota_X^{\mathcal{P}} = \left(\left(1 - \frac{d(F(\mathcal{P}), X)}{c(X)} \right) \frac{c(X) - c(\mathcal{P})}{c(X)} \right)^+$$

This degree is called the **pattern intensity** *of $\mathcal{P}$ in X.*

Next, in patternism the mind of an intelligent system is conceived as the (fuzzy) set of patterns in that system, and the set of patterns emergent between that system and other systems with which it interacts. The latter clause means that the patternist perspective is inclusive of notions of distributed intelligence [Hut96]. Basically, the mind of a system is the fuzzy set of different simplifying representations of that system that may be adopted.

In the patternist perspective, intelligence is conceived as roughly indicated above: as the ability to achieve complex goals in complex environments; where complexity itself may be defined as the possession of a rich variety of patterns. A mind is thus a collection of patterns that is associated with a persistent dynamical process that achieves highly-patterned goals in highly-patterned environments.

An additional hypothesis made within the patternist philosophy of mind is that reflection is critical to intelligence. This lets us conceive an intelligent system as a dynamical system that recognizes patterns in its environment and itself, as part of its quest to achieve complex goals.

While this approach is quite general, it is not vacuous; it gives a particular structure to the tasks of analyzing and synthesizing intelligent systems. About any would-be intelligent system, we are led to ask questions such as:

- How are patterns represented in the system? That is, how does the underlying infrastructure of the system give rise to the displaying of a particular pattern in the system's behavior?
- What kinds of patterns are most compactly represented within the system?
- What kinds of patterns are most simply learned?
- What learning processes are utilized for recognizing patterns?
- What mechanisms are used to give the system the ability to introspect (so that it can recognize patterns in itself)?

Now, these same sorts of questions could be asked if one substituted the word "pattern" with other words like "knowledge" or "information". However, we have found that asking these questions in the context of pattern leads to more productive answers, avoiding unproductive byways and also tying in very nicely with the details of various existing formalisms and algorithms for knowledge representation and learning.

Among the many kinds of patterns in intelligent systems, *semiotic* patterns are particularly interesting ones. Peirce decomposed these into three categories:

[2] In some prior writings the term "psynet model of mind" has been used to refer to the application of patternist philosophy to cognitive theory, but this term has been "deprecated" in recent publications as it seemed to introduce more confusion than clarification.

- **iconic** patterns, which are patterns of contextually important internal similarity between two entities (e.g. an iconic pattern binds a picture of a person to that person)
- **indexical** patterns, which are patterns of spatiotemporal co-occurrence (e.g. an indexical pattern binds a wedding dress and a wedding)
- **symbolic** patterns, which are patterns indicating that two entities are often involved in the same relationships (e.g. a symbolic pattern between the number "5" (the symbol) and various sets of 5 objects (the entities that the symbol is taken to represent))

Of course, some patterns may span more than one of these semiotic categories; and there are also some patterns that don't fall neatly into any of these categories. But the semiotic patterns are particularly important ones; and symbolic patterns have played an especially large role in the history of AI, because of the radically different approaches different researchers have taken to handling them in their AI systems. Mathematical logic and related formalisms provide sophisticated mechanisms for combining and relating symbolic patterns ("symbols"), and some AI approaches have focused heavily on these, sometimes more so than on the identification of symbolic patterns in experience or the use of them to achieve practical goals.

Pursuing the patternist philosophy in detail leads to a variety of particular hypotheses and conclusions about the nature of mind. Following from the view of intelligence in terms of achieving complex goals in complex environments, comes a view in which the dynamics of a cognitive system are understood to be governed by two main forces:

- self-organization, via which system dynamics cause existing system patterns to give rise to new ones
- goal-oriented behavior, which has been defined more rigorously in [Goe10b], but basically amounts to a system interacting with its environment in a way that appears like an attempt to maximize some reasonably simple function

Self-organized and goal-oriented behavior must be understood as cooperative aspects. For instance – to introduce an example that will be elaborated in more detail below – an agent is asked to build a surprising structure out of blocks and does so, this is goal-oriented. But the agent's ability to carry out this goal-oriented task will be greater if it has previously played around with blocks a lot in an unstructured, spontaneous way. And the "nudge toward creativity" given to it by asking it to build a surprising blocks structure may cause it to explore some novel patterns, which then feed into its future unstructured blocks play.

Based on these concepts, as argued in detail in [Goe06], several primary dynamical principles may be posited, including the following. For consistency of explanation, we will illustrate these principles with examples from the "playing with blocks" domain, which has the advantage of simplicity, and also of relating closely to our current work with OpenCog-controlled video game agents. However, the readers should not get the impression that CogPrime has somehow been specialized for this sort of domain; it has not been. The principles:

- **Evolution** , conceived as a general process via which patterns within a large population thereof are differentially selected and used as the basis for formation of new patterns, based on some "fitness function" that is generally tied to the goals of the agent
  - *Example:* If trying to build a blocks structure that will surprise Bob, an agent may simulate several procedures for building blocks structures in its "mind's eye", assessing for each one the expected degree to which it might surprise Bob. The search through procedure space could be conducted as a form of evolution, via an evolutionary algorithm such as CogPrime's MOSES (to be discussed below).
- **Autopoiesis**: the process by which a system of interrelated patterns maintains its integrity, via a dynamic in which whenever one of the patterns in the system begins to decrease in intensity, some of the other patterns increase their intensity in a manner that causes the troubled pattern to increase in intensity again

  - *Example:* An agent's set of strategies for building the base of a tower, and its set of strategies for building the middle part of a tower, are likely to relate autopoietically. If the system partially forgets how to build the base of a tower, then it may regenerate this missing knowledge via using its knowledge about how to build the middle part (i.e., it knows it needs to build the base in a way that will support good middle parts). Similarly if it partially forgets how to build the middle part, then it may regenerate this missing knowledge via using its knowledge about how to build the base (i.e. it knows a good middle part should fit in well with the sorts of base it knows are good).
  - This same sort of interdependence occurs between pattern-sets containing more than two elements
  - Sometimes (as in the above example) autopoietic interdependence in the mind is tied to interdependencies in the physical world, sometimes not.

- **Association**. Patterns, when given attention, spread some of this attention to other patterns that they have previously been associated with in some way. Furthermore, there is Peirce's law of mind [Pei34], which could be paraphrased in modern terms as stating that the mind is an associative memory network, whose dynamics dictate that every idea in the memory is an active agent, continually acting on those ideas with which the memory associates it.
  - *Example:* Building a blocks structure that resembles a tower, spreads attention to memories of prior towers the agents has seen, and also to memories of people whom the agent knows has seen towers, and structures it has built at the same time as towers, structures that resemble towers in various respects, etc.
- **Differential attention allocation / credit assignment**. Patterns that have been valuable for goal-achievement are given more attention, and are encouraged to participate in giving rise to new patterns.
  - *Example:* Perhaps in a prior instance of the task "build me a surprising structure out of blocks," searching through memory for non-blocks structures that the agent has played with has proved a useful cognitive strategy. In that case, when the task is posed to the agent again, it should tend to allocate disproportionate resources to this strategy.
- **Pattern creation**. Patterns that have been valuable for goal-achievement are mutated and combined with each other to yield new patterns.
  - *Example:* Building towers has been useful in a certain context, but so has building structures with a large number of triangles. Why not build a tower out of triangles? Or maybe a vaguely tower-like structure that uses more triangles than a tower easily could?
  - *Example:* Building an elongated block structure resembling a table was successful in the past, as was building a structure resembling a very flat version of a chair. Generalizing, maybe building distorted versions of furniture is good. Or maybe it is building distorted version of *any* previously perceived objects that is good. Or maybe both, to different degrees....

Next, for a variety of reasons outlined in [Goe06] it becomes appealing to hypothesize that the network of patterns in an intelligent system must give rise to the following large-scale emergent structures

- Hierarchical network. Patterns are habitually in relations of control over other patterns that represent more specialized aspects of themselves.
  - *Example:* The pattern associated with "tall building" has some control over the pattern associated with "tower", as the former represents a more general concept ... and "tower" has some control over "Eiffel tower", etc.
- Heterarchical network. The system retains a memory of which patterns have previously been associated with each other in any way.

  - *Example:* "Tower" and "snake" are distant in the natural pattern hierarchy, but may be associatively/heterarchically linked due to having a common elongated structure. This heterarchical linkage may be used for many things, e.g. it might inspire the creative construction of a tower with a snake's head.

- Dual network. Hierarchical and heterarchical structures are combined, with the dynamics of the two structures working together harmoniously. Among many possible ways to hierarchically organize a set of patterns, the one used should be one that causes hierarchically nearby patterns to have many meaningful heterarchical connections; and of course, there should be a tendency to search for heterarchical connections among hierarchically nearby patterns.
  - *Example:* While the set of patterns hierarchically nearby "tower" and the set of patterns heterarchically nearby "tower" will be quite different, they should still have more overlap than random pattern-sets of similar sizes. So, if looking for something else heterarchically near "tower", using the hierarchical information about "tower" should be of some use, and vice versa
  - In PLN, hierarchical relationships correspond to Atoms A and B so that $InheritanceAB$ and $InheritanceBA$ have highly dissimilar strength; and heterarchical relationships correspond to IntensionalSimilarity relationships. The dual network structure then arises when intensional and extensional inheritance approximately correlate with each other, so that inference about either kind of inheritance assists with figuring out about the other kind.

- Self structure. A portion of the network of patterns forms into an approximate image of the overall network of patterns.
  - *Example:* Each time the agent builds a certain structure, it observes itself building the structure, and its role as "builder of a tall tower" (or whatever the structure is) becomes part of its self-model. Then when it is asked to build something new, it may consult its self-model to see if it believes itself capable of building that sort of thing (for instance, if it is asked to build something very large, its self-model may tell it that it lacks persistence for such projects, so it may reply "I can try, but I may wind up not finishing it").

If the patternist theory of mind presented in [Goe06] is indeed appropriate as a guide for AGI work, then the success of CogPrime as a design will depend largely on whether these high-level structures and dynamics can be made to emerge from the synergetic interaction of CogPrime 's representation and algorithms, when they are utilized to control an appropriate agent in an appropriate environment. The extended treatment of CogPrime given in [GPGtOT13], takes care to specifically elaborate how each of these abstract concepts arises concretely from CogPrime 's structures and algorithms. In the more concise treatment given here, we will touch this aspect only lightly.

3.1 A Mind-World Correspondence Principle

Beyond patternist philosophy per se, an additional philosophical principle has guided CogPrime design; this is the "mind-world correspondence principle", which enlarges on the notion of "intelligence as adaptation to environments" mentioned above.

Real-world minds are always adapted to certain classes of environments and goals. As we have noted above, even a system of vast general intelligence, subject to real-world space and time constraints, will necessarily be more efficient at some kinds of learning than others. Thus, one approach to analyzing general intelligence is to look at the relationship between minds and worlds where a world is conceived as an environment and a set of goals defined in terms of that environment.

An informal version of the "mind-world correspondence principle" given in [Goe11] is as follows: For intelligence to occur, there has to be a natural correspondence between the transition-sequences of world-states and the corresponding transition-sequences of mind-states, at least in the cases of transition-sequences leading to relevant goals. A slightly more rigorous version is:

MIND-WORLD CORRESPONDENCE-PRINCIPLE . For a mind to work intelligently toward certain goals in a certain world, there should be a nice mapping from goal-directed sequences of world-states into sequences of mind-states, where nice means that a world-state-sequence W composed of two parts W_1 and W_2, gets mapped into a mind-state-sequence M composed of two corresponding parts M_1 and M_2.

What's nice about this principle is that it relates the decomposition of the world into parts, to the decomposition of the mind into parts. (A more fully formalized statement of the principle involves mathematical concepts that won't be introduced here for risk of digression. [3]

Each component of CogPrime has been carefully thought-through in terms of this conceptual principle; i.e. it has been designed so that its internal dynamics are decomposable in a way that maps into everyday human goals and environment in a way that matches the natural decomposition of the dynamics of these goals and environments. This aspect of the CogPrime design will not be highlighted here due to space considerations, but is a persistent theme in the longer treatment in [GPGtOT13].

4 High-Level Architecture of CogPrime

The above philosophical principles would be consistent with a very wide variety of concrete AGI designs. CogPrime has not been directly derived from these philosophical principles; rather, it has been created via beginning with a combination of human cognitive psychology and computer science algorithms and structures, and then shaping this combination so as to yield a system that appears likely to be conformant with these philosophical principles, as well as being computationally feasible on current hardware and containing cognitive structures and dynamics roughly homologous to the key human ones.

Figures 2, 3 , 5 and 6 depict the high-level architecture of CogPrime. A key underlying principle is: **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 robot preschool context, for example, the top-level goals would be everyday things such as pleasing the teacher, learning new information and skills, and protecting the robot's body. Figure 4 shows part of the architecture via which cognitive processes interact with each other, via commonly acting on the AtomSpace knowledge repository.

It is interesting to compare these diagrams to the integrative human cognitive architecture diagram given in [GIW12], which is intended to compactly overview the structure of human cognition as currently understood. The main difference is that the CogPrime diagrams commit to specific structures (e.g. knowledge representations) and processes, whereas the generic integrative architecture diagram refers merely to types of structures and processes. For instance, the integrative diagram refers generally to declarative knowledge and learning, whereas the CogPrime diagram refers to PLN, as a specific system for reasoning and learning about declarative knowledge. In [GPGtOT13] a table is provided articulating the key connections between the components of the CogPrime diagram and the well-known human cognitive structures/processes represented in integrative diagram, thus indicating the general cognitive functions instantiated by each of the CogPrime components.

5 Local and Global Knowledge Representation

One of the biggest decisions to make in designing an AGI system is how the system should represent knowledge. Naturally any advanced AGI system is going to synthesize a lot of its own knowledge representations for handling particular sorts of knowledge – but still, an AGI design typically makes *at least* some sort of commitment about the category of knowledge representation mechanisms toward which the AGI system will be biased.

OpenCog's knowledge representation mechanisms are all based fundamentally on *networks*. This is because we feel that, on a philosophical level, one of the most powerful known metaphors for understanding minds is to view them as networks of interrelated, interconnected elements. The view of mind as network is

[3] For the curious reader, a more rigorous statement of the principle looks like: **For an organism with a reasonably high level of intelligence in a certain world, relative to a certain set of goals, the mind-world path transfer function is a goal-weighted approximate functor**

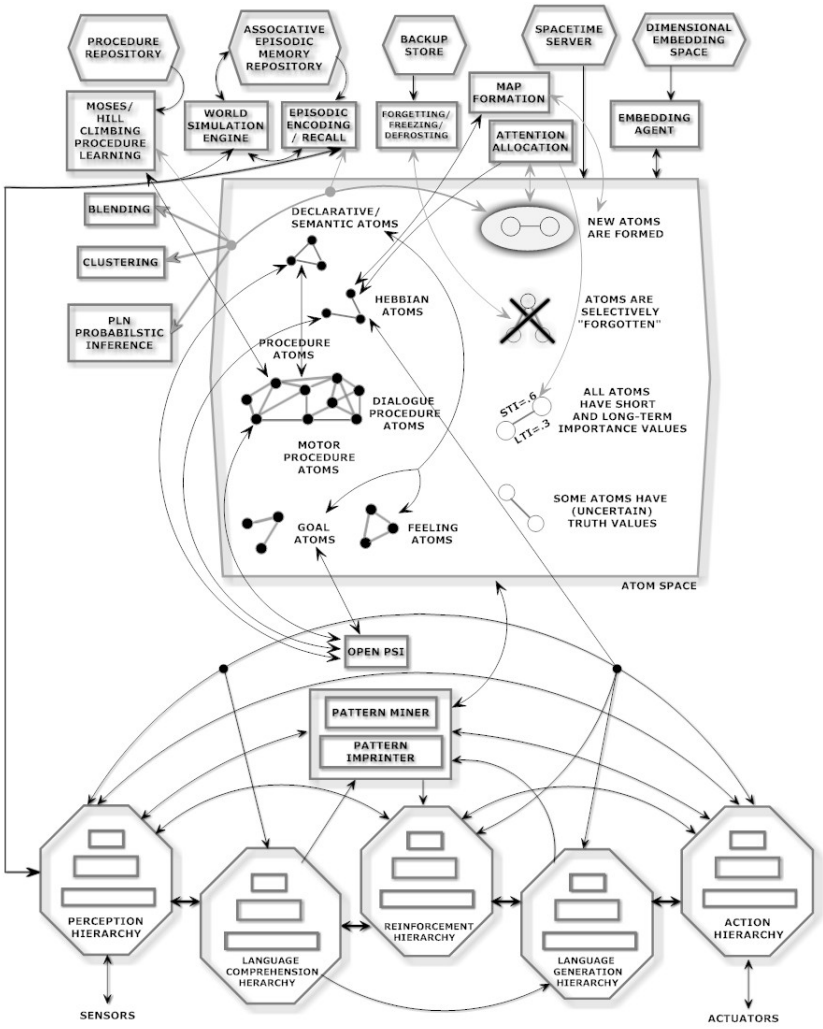

Figure 2: **High-Level Architecture of CogPrime.** This is a conceptual depiction, not a detailed flowchart (which would be too complex for a single image). Figures **??** , **??** and **??** highlight specific aspects of this diagram.

implicit in the patternist philosophy, because every pattern can be viewed as a pattern *in* something, or a pattern of arrangement *of* something – thus a pattern is always viewable as a relation between two or more things. A collection of patterns is thus a pattern-network. Knowledge of all kinds may be given network representations; and cognitive processes may be represented as networks also, for instance via representing them as programs, which may be represented as trees or graphs in various standard ways. The emergent patterns arising in an intelligence as it develops may be viewed as a pattern network in themselves; and the relations between an embodied mind and its physical and social environment may be viewed in terms of ecological and social networks.

The two major supercategories of knowledge representation systems are *local* (also called *explicit*) and *global* (also called *implicit*) systems, with a hybrid category we refer to as *glocal* that combines both of these. In a local system, each piece of knowledge is stored using a small percentage of cognitive system elements; in a global system, each piece of knowledge is stored using a particular pattern of arrangement, activation, etc. of a large percentage of cognitive system elements; in a glocal system, the two approaches are used together. All three of these knowledge representation types may be realized using networks. In CogPrime, all three are realized using the *same* (Atomspace) network.

In the first part of this section we discuss the symbolic, semantic-network aspects of knowledge repre-

sentation in CogPrime. Then, at the end of the section we turn to distributed, neural-net-like knowledge representation, focusing largely on CogPrime's "glocal" knowledge representation mechanisms.

5.1 Weighted, Labeled Hypergraphs

There are many different mechanisms for representing knowledge in AI systems in an explicit, localized way, most of them descending from various variants of formal logic. Here we briefly describe how it is done in CogPrime. On the surface, CogPrime's explicit representation scheme is not that different from a number of prior approaches. However, the particularities of CogPrime's explicit knowledge, representation, however, are carefully tuned to match CogPrime's cognitive processes, which are more distinctive in nature than the corresponding representational mechanisms.

One useful way to think about CogPrime's explicit, localized knowledge representation is in terms of hypergraphs. A hypergraph is an abstract mathematical structure [Bol98], which consists of objects called Nodes and objects called Links which connect the Nodes. In computer science, a graph traditionally means a bunch of dots connected with lines (i.e. Nodes connected by Links, or nodes connected by links). A hypergraph, on the other hand, can have Links that connect more than two Nodes.

In CogPrime it is most useful to consider "generalized hypergraphs" that extend ordinary hypergraphs by containing two additional features:

- Links that point to Links instead of Nodes
- Nodes that, when you zoom in on them, contain embedded hypergraphs.

Properly, such "hypergraphs" should always be referred to as generalized hypergraphs, but this is cumbersome, so we will persist in calling them merely hypergraphs. In a hypergraph of this sort, Links and Nodes are not as distinct as they are within an ordinary mathematical graph (for instance, they can both have Links connecting them), and so it is useful to have a generic term encompassing both Links and Nodes; for this purpose, we use the term Atom.

A weighted, labeled hypergraph is a hypergraph whose Links and Nodes come along with labels, and with one or more numbers that are generically called weights. A label associated with a Link or Node may sometimes be interpreted as telling you what type of entity it is, or alternatively as telling you what sort of data is associated with a Node. On the other hand, an example of a weight that may be attached to a Link or Node is a number representing a probability, or a number representing how important the Node or Link is.

Obviously, hypergraphs may come along with various sorts of dynamics. Minimally, one may think about:

- Dynamics that modify the properties of Nodes or Links in a hypergraph (such as the labels or weights attached to them.)
- Dynamics that add new Nodes or Links to a hypergraph, or remove existing ones.

Both types of dynamics are very important in CogPrime.

5.1.1 Atoms: Their Types and Weights

This section reviews a variety of CogPrime Atom types and gives simple examples of each of them. The Atom types considered are drawn from those currently in use in the OpenCog system. This does not represent a complete list of Atom types required for optimally efficient implementation of a complete CogPrime system, nor a complete list of those used in OpenCog currently (though it does cover a substantial majority of those used in OpenCog currently, omitting only some with specialized importance or intended only for temporary use).

The partial nature of the list given here reflects a more general point: The specific collection of Atom types in an OpenCog system is bound to change as the system is developed and experiment with. CogPrime specifies a certain collection of representational approaches and cognitive algorithms for acting on them; any of these approaches and algorithms may be implemented with a variety of sets of Atom types. The specific set of Atom types in the OpenCog system currently does not necessarily have a profound and lasting

significance – the list might look a bit different five years from time of writing, based on various detailed changes.

The treatment here is informal and intended to get across the general idea of what each Atom type does. A longer and more formal treatment of the Atom types is given in the online OpenCog wikibook [Goe10a] and in [GPGtOT13].

5.1.2 Some Basic Atom Types

We begin with ConceptNode – and note that a ConceptNode does not necessarily refer to a whole concept, but may refer to part of a concept; it is essentially a "basic semantic node" whose meaning comes from its links to other Atoms. It would be more accurately, but less tersely, named "concept or concept fragment or element node." A simple example would be a ConceptNode grouping nodes that are somehow related, e.g.

```
ConceptNode: C
InheritanceLink (ObjectNode: BW) C
InheritanceLink (ObjectNode: BP) C
InheritanceLink (ObjectNode: BN) C
ReferenceLink BW (PhraseNode "Ben's watch")
ReferenceLink BP (PhraseNode "Ben's passport")
ReferenceLink BN (PhraseNode "Ben's necklace")
```

indicates the simple and uninteresting ConceptNode grouping three objects owned by Ben (note that the above-given Atoms don't indicate the ownership relationship, they just link the three objects with textual descriptions). In this example, the ConceptNode links transparently to physical objects and English descriptions, but in general this won't be the case – most ConceptNodes will look to the human eye like groupings of links of various types, that link to other nodes consisting of groupings of links of various types, etc.

There are Atoms referring to basic, useful mathematical objects, e.g. NumberNodes like

```
NumberNode #4
NumberNode #3.44
```

The numerical value of a NumberNode is explicitly referenced within the Atom.

A core distinction is made between ordered links and unordered links; these are handled differently in the Atomspace software. A basic unordered link is the SetLink, which groups its arguments into a set. For instance, the ConceptNode C defined by

```
ConceptNode C
MemberLink A C
MemberLink B C
```

is equivalent to

```
SetLink A B
```

On the other hand, ListLinks are like SetLinks but ordered, and they play a fundamental role due to their relationship to predicates. Most predicates are assumed to take ordered arguments, so we may say e.g.

```
EvaluationLink
   PredicateNode eat
   ListLink
        ConceptNode cat
        ConceptNode mouse
```

to indicate that cats eat mice.

Note that by an expression like

```
ConceptNode cat
```

is meant

```
ConceptNode C
ReferenceLink W C
WordNode W #cat
```

since it's WordNodes rather than ConceptNodes that refer to words. (And note that the strength of the ReferenceLink would not be 1 in this case, because the word "cat" has multiple senses.) However, there is no harm nor formal incorrectness in the "ConceptNode cat" usage, since "cat" is just as valid a name for a ConceptNode as, say, "C."

We've already introduced above the MemberLink, which is a link joining a member to the set that contains it. Notable is that the truth value of a MemberLink is fuzzy rather than probabilistic, and that PLN (CogPrime's logical reasoning component [GIGH08]) is able to inter-operate fuzzy and probabilistic values.

SubsetLinks also exist, with the obvious meaning, e.g.

```
ConceptNode cat
ConceptNode animal
SubsetLink cat animal
```

Note that SubsetLink refers to a purely *extensional* subset relationship, and that InheritanceLInk should be used for the generic "intensional + extensional" analogue of this – more on this below. SubsetLink could more consistently (with other link types) be named ExtensionalInheritanceLink, but SubsetLink is used because it's shorter and more intuitive.

There are links representing Boolean operations AND, OR and NOT. For instance, we may say

```
ImplicationLink
    ANDLink
         ConceptNode young
         ConceptNode beautiful
     ConceptNode attractive
```

or, using links and VariableNodes instead of ConceptNodes,

```
AverageLink $X
   ImplicationLink
       ANDLink
            EvaluationLink young $X
            EvaluationLink beautiful $X
        EvaluationLink attractive $X
```

NOTLink is a unary link, so e.g. we might say

```
AverageLink $X
   ImplicationLink
       ANDLink
            EvaluationLink young $X
            EvaluationLink beautiful $X
            EvaluationLink
                 NOT
                 EvaluationLink poor $X
        EvaluationLink attractive $X
```

ContextLink allows explicit contextualization of knowledge, which is used in PLN, e.g.

```
ContextLink
   ConceptNode golf
   InheritanceLink
      ObjectNode BenGoertzel
      ConceptNode incompetent
```

says that Ben Goertzel is incompetent in the context of golf.

5.1.3 Variable Atoms

We have already introduced VariableNodes above; it's also possible to specify the type of a VariableNode via linking it to a VariableTypeNode via a TypedVariableLink, e.g.

```
VariableTypeLink
   VariableNode $X
   VariableTypeNode ConceptNode
```

which specifies that the variable $X should be filled with a ConceptNode.

Variables are handled via quantifiers; the default quantifier being the AverageLink, so that the default interpretation of

```
ImplicationLink
   InheritanceLink $X animal
   EvaluationLink
       PredicateNode: eat
       ListLink
           \$X
           ConceptNode: food
```

is

```
AverageLink $X
   ImplicationLink
      InheritanceLink $X animal
      EvaluationLink
          PredicateNode: eat
          ListLink
              \$X
              ConceptNode: food
```

The AverageLink invokes an estimation of the average TruthValue of the embedded expression (in this case an ImplicationLink) over all possible values of the variable $X. If there are type restrictions regarding the variable $X, these are taken into account in conducting the averaging. ForAllLink and ExistsLink may be used in the same places as AverageLink, with uncertain truth value semantics defined in PLN theory using third-order probabilities. There is also a ScholemLink used to indicate variable dependencies for existentially quantified variables, used in cases of multiply nested existential quantifiers.

EvaluationLink and MemberLink have overlapping semantics, allowing expression of the same conceptual/logical relationships in terms of predicates or sets, i.e.

```
EvaluationLink
   PredicateNode: eat
       ListLink
              $X
              ConceptNode: food
```

has the same semantics as

```
MemberLink
   ListLink
          $X
          ConceptNode: food
    ConceptNode: EatingEvents
```

The relation between the predicate "eat" and the concept "EatingEvents" is formally given by

```
ExtensionalEquivalenceLink
   ConceptNode: EatingEvents
   SatisfyingSetLink
         PredicateNode: eat
```

In other words, we say that "EatingEvents" is the SatisfyingSet of the predicate "eat": it is the set of entities that satisfy the predicate "eat". Note that the truth values of MemberLink and EvaluationLink are fuzzy rather than probabilistic.

5.1.4 Logical Links

There is a host of link types embodying logical relationships as defined in the PLN logic system [GIGH08], e.g.

- InheritanceLink
- SubsetLink (aka ExtensionalInheritanceLink)
- Intensional InheritanceLink

which embody different sorts of inheritance, e.g.

```
SubsetLink salmon fish
IntensionalInheritanceLink whale fish
InheritanceLink fish animal
```

and then

- SimilarityLink
- ExtensionalSimilarityLink
- IntensionalSimilarityLink

which are symmetrical versions, e.g.

```
SimilarityLink shark barracuda
IntensionalSimilarityLink shark dolphin
ExtensionalSimiliarityLink American obese_person
```

There are also higher-order versions of these links, both asymmetric

- ImplicationLink
- ExtensionalImplicationLink
- IntensionalImplicationLink

and symmetric

- EquivalenceLink
- ExtensionalEquivalenceLink
- IntensionalEquivalenceLink

These are used between predicates and links, e.g.

```
ImplicationLink
    EvaluationLink
         eat
        ListLink
             $X
             dirt
    EvaluationLink
         feel
         ListLink
              $X
              sick
```

or

```
ImplicationLink
    EvaluationLink
         eat
        ListLink
             $X
             dirt
    InheritanceLink $X sick
```

or

```
ForAllLink $X, $Y, $Z
   ExtensionalEquivalenceLink
       EquivalenceLink
           $Z
          EvaluationLink
              +
              ListLink
                  $X
                  $Y
       EquivalenceLink
           $Z
          EvaluationLink
              +
              ListLink
                  $Y
                  $X
```

Note, the latter is given as an extensional equivalence because it's a pure mathematical equivalence. This is not the only case of pure extensional equivalence, but it's an important one.

5.1.5 Temporal Links

There are also temporal versions of these links, such as

- PredictiveImplicationLink
- PredictiveAttractionLink
- SequentialANDLink
- SimultaneousANDLink

which combine logical relation between the argument with temporal relation between their arguments. For instance, we might say

```
PredictiveImplicationLink
    PredicateNode: JumpOffCliff
    PredicateNode: Dead
```

or including arguments,

```
PredictiveImplicationLink
    EvaluationLink JumpOffCliff $X
    EvaluationLink Dead $X
```

The former version, without variable arguments given, shows the possibility of using higher-order logical links to join predicates without any explicit variables. Via using this format exclusively, one could avoid VariableAtoms entirely, using only higher-order functions in the manner of pure functional programming formalisms like combinatory logic. However, this purely functional style has not proved convenient, so the Atomspace in practice combines functional-style representation with variable-based representation.

Temporal links often come with specific temporal quantification, e.g.

```
PredictiveImplicationLink <5 seconds>
    EvaluationLink JumpOffCliff $X
    EvaluationLink Dead $X
```

indicating that the conclusion will generally follow the premise within 5 seconds. There is a system for managing fuzzy time intervals and their interrelationships, based on a fuzzy version of Allen Interval Algebra.

SequentialANDLink is similar to PredictiveImplicationLink but its truth value is calculated differently. The truth value of

```
SequentialANDLink <5 seconds>
    EvaluationLink JumpOffCliff $X
    EvaluationLink Dead $X
```

indicates the likelihood of the sequence of events occurring in that order, with gap lying within the specified time interval. The truth value of the PredictiveImplicationLink version indicates the likelihood of the second event, conditional on the occurrence of the first event (within the given time interval restriction).

There are also links representing basic temporal relationships, such as BeforeLink and AfterLink. These are used to refer to specific events, e.g. if X refers to the event of Ben waking up on July 15 2012, and Y refers to the event of Ben getting out of bed on July 15 2012, then one might have

```
AfterLink X Y
```

And there are TimeNodes (representing time-stamps such as temporal moments or intervals) and AtTimeLinks, so we may e.g. say

```
AtTimeLink
   X
   TimeNode: 8:24AM Eastern Standard Time, July 15 2012 AD
```

5.2 Associative Links

There are links representing associative, attentional relationships,

- HebbianLink
- AsymmetricHebbianLink
- InverseHebbianLink
- SymmetricInverseHebbianLink

These connote associations between their arguments, i.e. they connote that the entities represented by the two arguments occurred in the same situation or context, for instance

```
HebbianLink happy smiling
AsymmetricHebbianLink dead rotten
InverseHebbianLink dead breathing
```

The asymmetric HebbianLink indicates that when the first argument is present in a situation, the second is also often present. The symmetric (default) version indicates that this relationship holds in both directions. The inverse versions indicate the negative relationship: e.g. when one argument is present in a situation, the other argument is often not present.

5.3 Procedure Nodes

There are nodes representing various sorts of procedures; these are kinds of ProcedureNode, e.g.

- SchemaNode, indicating any procedure
- GroundedSchemaNode, indicating any procedure associated in the system with a Combo program or C++ function allowing the procedure to be executed

- PredicateNode, indicating any predicate that associates a list of arguments with an output truth value
- GroundedPredicateNode, indicating a predicate associated in the system with a Combo program or C++ function allowing the predicate's truth value to be evaluated on a given specific list of arguments

ExecutionLinks and EvaluationLinks record the activity of SchemaNodes and PredicateNodes. We have seen many examples of EvaluationLinks in the above. Example ExecutionLinks would be:

```
ExecutionLink step\_forward
ExecutionLink step\_forward 5
ExecutionLink
    +
    ListLink
        NumberNode: 2
        NumberNode: 3
```

The first example indicates that the schema "step forward" has been executed. The second example indicates that it has been executed with an argument of "5" (meaning, perhaps, that 5 steps forward have been attempted). The last example indicates that the "+" schema has been executed on the argument list (2,3), presumably resulting in an output of 5.

The output of a schema execution may be indicated using an ExecutionOutputLink, e.g.

```
ExecutionOutputLink
    +
    ListLink
        NumberNode: 2
        NumberNode: 3
```

refers to the value "5" (as a NumberNode).

5.4 Links for Special External Data Types

Finally, there are also Atom types referring to specific types of data important to using OpenCog in specific contexts.

For instance, there are Atom types referring to general natural language data types, such as

- WordNode
- SentenceNode
- WordInstanceNode
- DocumentNode

plus more specific ones referring to relationships that are part of link-grammar parses of sentences

- FeatureNode
- FeatureLink
- LinkGrammarRelationshipNode
- LinkGrammarDisjunctNode

or RelEx semantic interpretations of sentences

- DefinedLinguisticConceptNode
- DefinedLinguisticRelationshipNode
- PrepositionalRelationshipNode

There are also Atom types corresponding to entities important for embodying OpenCog in a virtual world, e.g.

- ObjectNode
- AvatarNode
- HumanoidNode
- UnknownObjectNode
- AccessoryNode

5.4.1 Truth Values and Attention Values

CogPrime Atoms (Nodes and Links) of various types are quantified with truth values that, in their simplest form, have two components, one representing probability (*strength*) and the other representing *weight of evidence*; and also with *attention values* that have two components, short-term and long-term importance, representing the estimated value of the Atom on immediate and long-term time-scales.

In practice many Atoms are labeled with CompositeTruthValues rather than elementary ones. A composite truth value contains many component truth values, representing truth values of the Atom in different contexts and according to different estimators.

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

5.5 Glocal Memory

Now we move on from localized memory to the "glocal" coordination of local and global memory, which plays a large role in the OpenCog architecture. A glocal memory is one that transcends the global/local dichotomy and incorporates both aspects in a tightly interconnected way. The notion of glocal memory has implicitly occurred in a number of neural theories (without use of the neologism "glocal"), e.g. [Cal96] and [Goe01], but was never extensively developed prior to the advent of CogPrime theory. In[HG08] [GPI+10] we describe glocality in attractor neural nets, a close analogue to CogPrime's attention allocation subsystem.

Glocal memory overcomes the dichotomy between localized memory (in which each memory item is stored in a single location within an overall memory structure) and global, distributed memory (in which a memory item is stored as an aspect of a multi-component memory system, in such a way that the same set of multiple components stores a large number of memories). In a glocal memory system, most memory items are stored both locally and globally, with the property that eliciting either one of the two records of an item tends to also elicit the other one.

Glocal memory applies to multiple forms of memory; however we will focus largely on perceptual and declarative memory in our detailed analyses here, so as to conserve space and maintain simplicity of discussion.

The central idea of glocal memory is that (perceptual, declarative, episodic, procedural, etc.) items may be stored in memory in the form of paired structures that are called (key, map) pairs. Of course the idea of a "pair" is abstract, and such pairs may manifest themselves quite differently in different sorts of memory systems (e.g. brains versus non-neuromorphic AI systems). The key is a localized version of the item, and records some significant aspects of the items in a simple and crisp way. The map is a dispersed, distributed version of the item, which represents the item as a (to some extent, dynamically shifting) combination of fragments of other items. The map includes the key as a subset; activation of the key generally (but not

necessarily always) causes activation of the map; and changes in the memory item will generally involve complexly coordinated changes on the key and map level both.

Memory is one area where animal brain architecture differs radically from the von Neumann architecture underlying nearly all contemporary general-purpose computers. Von Neumann computers separate memory from processing, whereas in the human brain there is no such distinction. Human memories are generally constructed in the course of remembering [Ros88], which gives human memory a strong capability for "filling in gaps" of remembered experience and knowledge; and also causes problems with inaccurate remembering in many contexts [BF71, RM95] e.g. We believe the constructive aspect of memory is largely associated with its glocality.

5.5.1 Neural-Symbolic Glocality in CogPrime

In CogPrime, we have explicitly sought to span the symbolic/emergentist pseudo-dichotomy, via creating an integrative knowledge representation that combines logic-based aspects with neural-net-like aspects. As reviewed above, these function not in the manner of a multimodular system, but rather via using (probabilistic logical) truth values and (attractor neural net like) attention values as weights on nodes and links of the same (hyper) graph. The nodes and links in this hypergraph are typed, like a standard semantic network approach for knowledge representation, so they're able to handle all sorts of knowledge, from the most concrete perception and actuation related knowledge to the most abstract relationships. But they're also weighted with values similar to neural net weights, and pass around quantities (importance values) similar to neural net activations, allowing emergent attractor/assembly based knowledge representation similar to attractor neural nets.

The concept of glocality lies at the heart of this combination, in a way that spans the pseudo-dichotomy:

- Local knowledge is represented in abstract logical relationships stored in explicit logical form, and also in Hebbian-type associations between nodes and links.
- Global knowledge is represented in large-scale patterns of node and link weights, which lead to large-scale patterns of network activity, which often take the form of attractors qualitatively similar to Hopfield net attractors. These attractors are called *maps*.

The result of all this is that a concept like "cat" might be represented as a combination of:

- A small number of logical relationships and strong associations, that constitute the "key" subnetwork for the "cat" concept.
- A large network of weak associations, binding together various nodes and links of various types and various levels of abstraction, representing the "cat map".

The activation of the key will generally cause the activation of the map, and the activation of a significant percentage of the map will cause the activation of the rest of the map, including the key. Furthermore, if the key were for some reason forgotten, then after a significant amount of effort, the system would likely to be able to reconstitute it (perhaps with various small changes) from the information in the map. We conjecture that this particular kind of glocal memory will turn out to be very powerful for AGI, due to its ability to combine the strengths of formal logical inference with those of self-organizing attractor neural networks.

As a simple example, consider the representation of a "tower", in the context of an artificial agent that has built towers of blocks, and seen pictures of many other kinds of towers, and seen some tall buildings that it knows are somewhat like towers but perhaps not exactly towers. If this agent is reasonably conceptually advanced (say, at Piagetan the concrete operational level) then its mind will contain some declarative relationships partially characterizing the concept of "tower," as well as its sensory and episodic examples, and its procedural knowledge about how to build towers.

The key of the "tower" concept in the agent's mind may consist of internal images and episodes regarding the towers it knows best, the essential operations it knows are useful for building towers (piling blocks atop blocks atop blocks...), and the core declarative relations summarizing "towerness" – and the whole "tower" map then consists of a much larger number of images, episodes, procedures and declarative relationships connected to "tower" and other related entities. If any portion of the map is removed – even if the key is

removed – then the rest of the map can be approximately reconstituted, after some work. Some cognitive operations are best done on the localized representation – e.g. logical reasoning. Other operations, such as attention allocation and guidance of inference control, are best done using the globalized map representation.

6 Memory Types and Associated Cognitive Processes in CogPrime

Now we dig deeper into the internals of the CogPrime approach, turning to aspects of the relationship between structure and dynamics. Architecture diagrams are all very well, but, ultimately it is dynamics that makes an architecture come alive. Intelligence is all about learning, which is by definition about change, about dynamical response to the environment and internal self-organizing dynamics.

CogPrime relies on multiple memory types and, as discussed above, is founded on the premise that the right course in architecting a pragmatic, roughly human-like AGI system is to handle different types of memory differently in terms of both structure and dynamics.

CogPrime's memory types are the declarative, procedural, sensory, and episodic memory types that are widely discussed in cognitive neuroscience [TC05], 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.). Figure 7 illustrates the relationships between several of the key memory types in the context of a simple situation involving an OpenCogPrime -controlled agent in a virtual world.

In terms of patternist cognitive theory, the multiple types of memory in CogPrime should be considered as specialized ways of storing particular types of pattern, optimized for spacetime efficiency. The cognitive processes associated with a certain type of memory deal with creating and recognizing patterns of the type for which the memory is specialized. While in principle all the different sorts of pattern could be handled in a unified memory and processing architecture, the sort of specialization used in CogPrime is necessary in order to achieve acceptable efficient general intelligence using currently available computational resources. And as we have argued in detail in [Goe10b], efficiency is not a side-issue but rather the essence of real-world AGI (since as Hutter has shown, if one casts efficiency aside, arbitrary levels of general intelligence can be achieved via a trivially simple program).

The essence of the CogPrime 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, yielding 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**

6.1 Cognitive Synergy in PLN

To put a little meat on the bones of the "cognitive synergy" idea mentioned above, we now elaborate a little on the role it plays in the interaction between procedural and declarative learning.

While MOSES handles much of CogPrime's procedural learning, and CogPrime's internal simulation engine handles most episodic knowledge, CogPrime'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 [GMIH08]; 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.

Memory Type	Specific Cognitive Processes	General Cognitive Functions
Declarative	Probabilistic Logic Networks (PLN) [GMIH08]; conceptual blending [FT02]	pattern creation
Procedural	MOSES (a novel probabilistic evolutionary program learning algorithm) [Loo06]	pattern creation
Episodic	internal simulation engine [GEA08]	association, pattern creation
Attentional	Economic Attention Networks (ECAN) [GPI+10]	association, credit assignment
Intentional	probabilistic goal hierarchy refined by PLN and ECAN, structured according to MicroPsi [Bac09]	credit assignment, pattern creation
Sensory	In CogBot, this will be supplied by the DeSTIN component	association, attention allocation, pattern creation, credit assignment

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

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. This careful management of uncertainty is critical for the application of logical inference in the robotics context, where most knowledge is abstracted from experience and is hence highly uncertain.

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 a robot 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 (for instance $L \equiv Concept \wedge Procedure \rightarrow Goal$ 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 $X \to B$. 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 CogPrime 's internal simulation engine, so as to observe the truth value of $A \to B$ 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. This is just one example of the multiple ways in which the different cognitive processes in CogPrime interact synergetically; a more thorough treatment of these interactions is given in [Goe09a].

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). For instance, 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 to the present case and infer that the new playmate is not likely to be a regular visitor.

7 Goal-Oriented Dynamics in CogPrime

CogPrime's dynamics has both goal-oriented and "spontaneous" aspects; here for simplicity's sake we will focus more heavily on the goal-oriented ones, although both are important. The basic goal-oriented dynamic of the CogPrime system, within which the various types of memory are utilized, is driven by implications known as "cognitive schematics", which take the form

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

(summarized $C \wedge P \to G$). Semi-formally, this implication may be 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 represented by truth value object 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.

CogPrime'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 CogPrime's functionality. Unlike with classical production rules systems, uncertainty is core to CogPrime's knowledge representation, and each CogPrime cognitive schematic is labeled with an uncertain truth value, which is critical to its utilization by CogPrime's cognitive processes. Also, in CogPrime, 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). A stronger similarity is to MicroPsi's triplets; the differences in this case are more low-level and technical and are reviewed in [GPGtOT13].

Finally, the biggest difference between CogPrime's cognitive schematics and production rules or other similar constructs, is that in CogPrime this level of knowledge representation is not the only important one. CLARION [SZ04], as reviewed above, is an example of a cognitive architecture that uses production rules for explicit knowledge representation and then uses a totally separate subsymbolic knowledge store for implicit knowledge. In CogPrime, 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 (see Figure 8 for a depiction of some explicit linguistic knowledge.)
- implicit knowledge stored in patterns of activity among these same nodes and links, defined via the activity of the "importance" values (see Figure 9 for an illustrative example thereof) associated with nodes and links and propagated by the ECAN attention allocation process

The meaning of a cognitive schematic in CogPrime 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, and the emergence of effective high-level hierarchical, heterarchical and self structures would be less likely.

7.1 Analysis and Synthesis Processes in CogPrime

The cognitive schematic $Context \wedge Procedure \rightarrow Goal$ leads to a conceptualization of the internal action of an intelligent system as involving two key categories of learning:

- **Analysis**: Estimating the probability p of a posited $C \wedge P \rightarrow G$ relationship
- **Synthesis**: Filling in one or two of the variables in the cognitive schematic, given assumptions regarding the remaining variables, and directed by the goal of maximizing the probability of the cognitive schematic

To further flesh these ideas out, we will now use examples from the "virtual dog" application to motivate the discussion.

For example, where synthesis is concerned,

- The MOSES probabilistic evolutionary program learning algorithm is applied to find P, given fixed C and G. Internal simulation is also used, for the purpose of creating a simulation embodying C and seeing which P lead to the simulated achievement of G.
  - *Example: A virtual dog learns a procedure P to please its owner (the goal G) in the context C where there is a ball or stick present and the owner is saying "fetch".*
- PLN inference, acting on declarative knowledge, is used for choosing C, given fixed P and G (also incorporating sensory and episodic knowledge as appropriate). Simulation may also be used for this purpose.
  - *Example: A virtual dog wants to achieve the goal G of getting food, and it knows that the procedure P of begging has been successful at this before, so it seeks a context C where begging can be expected to get it food. Probably this will be a context involving a friendly person.*
- PLN-based goal refinement is used to create new subgoals G to sit on the right hand side of instances of the cognitive schematic.
  - *Example: Given that a virtual dog has a goal of finding food, it may learn a subgoal of following other dogs, due to observing that other dogs are often heading toward their food.*
- Concept formation heuristics are used for choosing G and for fueling goal refinement, but especially for choosing C (via providing new candidates for C). They are also used for choosing P, via a process called "predicate schematization" that turns logical predicates (declarative knowledge) into procedures.
  - *Example: At first a virtual dog may have a hard time predicting which other dogs are going to be mean to it. But it may eventually observe common features among a number of mean dogs, and thus form its own concept of "pit bull," without anyone ever teaching it this concept explicitly.*

Where analysis is concerned:

- PLN inference, acting on declarative knowledge, is used for estimating the probability of the implication in the cognitive schematic, given fixed C, P and G. Episodic knowledge is also used in this regard, via enabling estimation of the probability via simple similarity matching against past experience. Simulation is also used: multiple simulations may be run, and statistics may be captured therefrom.
  - *Example: To estimate the degree to which asking Bob for food (the procedure P is "asking for food", the context C is "being with Bob") will achieve the goal G of getting food, the virtual dog may study its memory to see what happened on previous occasions where it or other dogs asked Bob for food or other things, and then integrate the evidence from these occasions.*
- Procedural knowledge, mapped into declarative knowledge and then acted on by PLN inference, can be useful for estimating the probability of the implication $C \wedge P \rightarrow G$, in cases where the probability of $C \wedge P_1 \rightarrow G$ is known for some P_1 related to P.
  - *Example: knowledge of the internal similarity between the procedure of asking for food and the procedure of asking for toys, allows the virtual dog to reason that if asking Bob for toys has been successful, maybe asking Bob for food will be successful too.*
- Inference, acting on declarative or sensory knowledge, can be useful for estimating the probability of the implication $C \wedge P \rightarrow G$, in cases where the probability of $C_1 \wedge P \rightarrow G$ is known for some C_1 related to C.
  - *Example: if Bob and Jim have a lot of features in common, and Bob often responds positively when asked for food, then maybe Jim will too.*
- Inference can be used similarly for estimating the probability of the implication $C \wedge P \rightarrow G$, in cases where the probability of $C \wedge P \rightarrow G_1$ is known for some G_1 related to G. Concept creation can be useful indirectly in calculating these probability estimates, via providing new concepts that can be used to make useful inference trails more compact and hence easier to construct.
  - *Example: The dog may reason that because Jack likes to play, and Jack and Jill are both children, maybe Jill likes to play too. It can carry out this reasoning only if its concept creation process has invented the concept of "child" via analysis of observed data.*

In these examples we have focused on cases where two terms in the cognitive schematic are fixed and the third must be filled in; but just as often, the situation is that only one of the terms is fixed. For instance, if we fix G, sometimes the best approach will be to collectively learn C and P. This requires either a procedure learning method that works interactively with a declarative-knowledge-focused concept learning or reasoning method; or a declarative learning method that works interactively with a procedure learning method. That is, it requires the sort of cognitive synergy built into the CogPrime design.

8 Clarifying the Key Claims

Having overviewed some of the basics of the CogPrime design, we now return to the "key claims" that were listed at the end of the Introduction As noted there, this is a list of claims such that – roughly speaking – if the reader accepts these claims, they should accept that the CogPrime approach to AGI is a viable one. On the other hand if the reader rejects one or more of these claims, they may well find one or more aspects of CogPrime unacceptable for some related reason. Above, we merely listed these claims; here we briefly discuss each one in the context of CogPrime concepts presented in the intervening sections.

As we clarified in the Introduction, we don't fancy that we have provided an ironclad argument that the CogPrime approach to AGI is guaranteed to work as hoped, once it's fully engineered, tuned and taught. Mathematics isn't yet adequate to analyze the real-world behavior of complex systems like these [4]; and we

[4] Although an Appendix of [GPGtOT13] gives a list of formal propositions echoing many of the ideas in the chapter – propositions such that, if they are true, then the success of CogPrime as an architecture for general intelligence is likely.

have not yet implemented, tested and taught enough of CogPrime to provide convincing empirical validation. So, most of the claims listed here have not been rigorously demonstrated, but only heuristically argued for. That is the reality of AGI work right now: one assembles a design based on the best combination of rigorous and heuristic arguments one can, then proceeds to create and teach a system according to the design, adjusting the details of the design based on experimental results as one goes along.

8.1 Multi-Memory Systems

The first of our key claims is that *to achieve general intelligence in the context of human-intelligence-friendly environments and goals using feasible computational resources, it's important that an AGI system can handle different kinds of memory (declarative, procedural, episodic, sensory, intentional, attentional) in customized but interoperable ways.* The basic idea is that these different kinds of knowledge have very different characteristics, so that trying to handle them all within a single approach, while surely possible, is likely to be unacceptably inefficient.

The tricky issue in formalizing this claim is that "single approach" is an ambiguous notion: for instance, if one has a wholly logic-based system that represents all forms of knowledge using predicate logic, then one may still have specialized inference control heuristics corresponding to the different kinds of knowledge mentioned in the claim. In this case one has "customized but interoperable ways" of handling the different kinds of memory, and one doesn't really have a "single approach" even though one is using logic for everything. To bypass such conceptual difficulties, one may formalize cognitive synergy using a geometric framework as discussed in [GI11], in which different types of knowledge are represented as metrized categories, and cognitive synergy becomes a statement about paths to goals being shorter in metric spaces combining multiple knowledge types than in those corresponding to individual knowledge types.

In CogPrime we use a complex combination of representations, including the Atomspace for declarative, attentional and intentional knowledge and some episodic and sensorimotor knowledge, Combo programs for procedural knowledge, simulations for episodic knowledge, and hierarchical neural nets for some sensorimotor knowledge (and related episodic, attentional and intentional knowledge). In cases where the same representational mechanism is used for different types of knowledge, different cognitive processes are used, and often different aspects of the representation (e.g. attentional knowledge is dealt with largely by ECAN acting on AttentionValues and HebbianLinks in the Atomspace; whereas declarative knowledge is dealt with largely by PLN acting on TruthValues and logical links, also in the AtomSpace). So one has a mix of the "different representations for different memory types" approach and the "different control processes on a common representation for different memory types" approach.

It's unclear how closely dependent the need for a multi-memory approach is on the particulars of "human-friendly environments." We have argued in [Goe09b] that one factor militating in favor of a multi-memory approach is the need for multimodal communication: declarative knowledge relates to linguistic communication; procedural knowledge relates to demonstrative communication; attentional knowledge relates to indicative communication; and so forth. But in fact the multi-memory approach may have a broader importance, even to intelligences without multimodal communication. This is an interesting issue but not particularly critical to the development of human-like, human-level AGI, since in the latter case we are specifically concerned with creating intelligences that *can* handle multimodal communication. So if for no other reason, the multi-memory approach is worthwhile for handling multi-modal communication.

Pragmatically, it is also quite clear that the human brain takes a multi-memory approach, e.g. with the cerebellum and closely linked cortical regions containing special structures for handling procedural knowledge, with special structures for handling motivational (intentional) factors, etc. And (though this point is certainly not definitive, it's meaningful in the light of the above theoretical discussion) decades of computer science and narrow-AI practice strongly suggest that the "one memory structure fits all" approach is not capable of leading to effective real-world approaches.

8.2 Perception, Action and Environment

The more we understand of human intelligence, the clearer it becomes how closely it has evolved to match the particular goals and environments for which the human organism evolved. This is true in a broad sense, as illustrated by the above issues regarding multi-memory systems, and is also true in many particulars, as

illustrated e.g. by Changizi's [Cha09] evolutionary analysis of the human visual system. While it might be possible to create a human-like, human-level AGI by abstracting the relevant biases from human biology and behavior and explicitly encoding them in one's AGI architecture, it seems this would be an inordinately difficult approach in practice, leading to the claim that *to achieve human-like general intelligence, it's important for an intelligent agent to have sensory data and motoric affordances that roughly emulate those available to humans*. We don't claim this is a necessity – just a dramatic convenience. And if one accepts this point, it has major implications for what sorts of paths toward AGI it makes most sense to follow.

Unfortunately, though, the idea of a "human-like" set of goals and environments is fairly vague; and when you come right down to it, *we don't know exactly how close the emulation needs to be* to form a natural scenario for the maturation of human-like, human-level AGI systems. One could attempt to resolve this issue via a priori theory, but given the current level of scientific knowledge it's hard to see how that would be possible in any definitive sense ... which leads to the conclusion that *our AGI systems and platforms need to support fairly flexible experimentation with virtual-world and/or robotic infrastructures.*

Our own intuition is that currently neither current virtual world platforms, nor current robotic platforms, are *quite* adequate for the development of human-level, human-like AGI. Virtual worlds would need to become a lot more like robot simulators, allowing more flexible interaction with the environment, and more detailed control of the agent. Robots would need to become more robust at moving and grabbing – e.g. with Big Dog's movement ability but the grasping capability of the best current grabber arms. We do feel that development of adequate virtual world or robotics platforms is quite possible using current technology, and could be done at fairly low cost if someone were to prioritize this. Even without AGI-focused prioritization, it seems that the needed technological improvements are likely to happen during the next decade for other reasons. So at this point we feel it makes sense for AGI researchers to focus on AGI and exploit embodiment-platform improvements as they come along – at least, this makes sense in the case of AGI approaches (like CogPrime) that can be primarily developed in an embodiment-platform-independent manner.

8.3 Developmental Pathways

But if an AGI system is going to live in human-friendly environments, what should it do there? No doubt very many pathways leading from incompetence to adult-human-level general intelligence exist, but one of them is much better understood than any of the others, and that's the one normal human children take. Of course, given their somewhat different embodiment, it doesn't make sense to try to force AGI systems to take *exactly* the same path as human children, but having AGI systems follow a fairly close approximation to the human developmental path seems the smoothest developmental course ... a point summarized by the claim that: *To work toward adult human-level, roughly human-like general intelligence, one fairly easily comprehensible path is to use environments and goals reminiscent of human childhood, and seek to advance one's AGI system along a path roughly comparable to that followed by human children.*

Human children learn via a rich variety of mechanisms; but broadly speaking one conclusion one may draw from studying human child learning is that it may make sense to *teach an AGI system aimed at roughly human-like general intelligence via a mix of spontaneous learning and explicit instruction, and to instruct it via a combination of imitation, reinforcement and correction, and a combination of linguistic and nonlinguistic instruction.* We have explored exactly what this means in [GEA08] and other prior experiments, via looking at examples of these types of learning in the context of virtual pets in virtual worlds, and exploring how specific CogPrime learning mechanisms can be used to achieve simple examples of these types of learning.

One important case of learning that human children are particularly good at is language learning; and in [Goe08] we have argued that this is a case where it may pay for AGI systems to take a route somewhat different from the one taken by human children. Humans seem to be born with a complex system of biases enabling effective language learning, and it's not yet clear exactly what these biases are nor how they're incorporated into the learning process. It is very tempting to give AGI systems a "short cut" to language proficiency via making use of existing rule-based and statistical-corpus-analysis-based NLP systems; and we have fleshed out this approach sufficiently to have convinced ourselves it makes practical as well as conceptual sense, in the context of the specific learning mechanisms and NLP tools built into OpenCog. Thus we have provided a number of detailed arguments and suggestions in support of our claim that *one effective approach to teaching an AGI system human language is to supply it with some in-built linguistic facility, in the form of rule-based and statistical-linguistics-based NLP systems, and then allow it to improve and revise this facility*

based on experience.

8.4 Knowledge Representation

Many knowledge representation approaches have been explored in the AI literature, and ultimately many of these could be workable for human-level AGI if coupled with the right cognitive processes. The key goal for a knowledge representation for AGI should be *naturalness* with respect to the AGI's cognitive processes – i.e. the cognitive processes shouldn't need to undergo complex transformative gymnastics to get information in and out of the knowledge representation in order to do their cognitive work. Toward this end we have come to a similar conclusion to some other researchers (e.g. Joscha Bach and Stan Franklin), and concluded that *given the strengths and weaknesses of current and near-future digital computers, a (loosely) neural-symbolic network is a good representation for directly storing many kinds of memory, and interfacing between those that it doesn't store directly.* CogPrime's AtomSpace is a neural-symbolic network designed to work nicely with PLN, MOSES, ECAN and the other key CogPrime cognitive processes; it supplies them with what they need without causing them undue complexities. It provides a platform that these cognitive processes can use to adaptively, automatically construct specialized knowledge representations for particular sorts of knowledge that they encounter.

8.5 Cognitive Processes

The crux of intelligence is dynamics, learning, adaptation; and so the crux of an AGI design is the set of cognitive processes that the design provides. These processes must collectively allow the AGI system to achieve its goals in its environments using the resources at hand. Given CogPrime's multi-memory design, it's natural to consider CogPrime's cognitive processes in terms of which memory subsystems they focus on (although, this is not a perfect mode of analysis, since some of the cognitive processes span multiple memory types).

8.5.1 Uncertain Logic for Declarative Knowledge

One major decision made in the creation of CogPrime was that *given the strengths and weaknesses of current and near-future digital computers, uncertain logic is a good way to handle declarative knowledge.* Of course this is not obvious nor is it the only possible route. Declarative knowledge can potentially be handled in other ways; e.g. in a hierarchical network architecture, one can make declarative knowledge emerge automatically from procedural and sensorimotor knowledge, as is the goal in the HTM and DeSTIN designs, for example. It seems clear that the human brain doesn't contain anything closely parallel to formal logic – even though one can ground logic operations in neural-net dynamics as explored in [GP08], this sort of grounding leads to "uncertain logic enmeshed with a host of other cognitive dynamics" rather than "uncertain logic as a cleanly separable cognitive process."

But contemporary digital computers are not brains – they lack the human brain's capacity for cheap massive parallelism, but have a capability for single-operation speed and precision far exceeding the brain's. In this way computers and formal logic are a natural match (a fact that's not surprising given that Boolean logic lies at the foundation of digital computer operations). Using *uncertain logic* is a sort of compromise between brainlike messiness and fuzziness, and computerlike precision. An alternative to using uncertain logic is using crisp logic and incorporating uncertainty as content within the knowledge base – this is what SOAR [Lai12] does, for example, and it's not a wholly unworkable approach. But given that the vast mass of knowledge needed for confronting everyday human reality is highly uncertain, and that this knowledge often needs to be manipulated efficiently in real-time, it seems to us there is a strong argument for embedding uncertainty in the logic.

Many approaches to uncertain logic exist in the literature, including probabilistic and fuzzy approaches, and one conclusion we reached in formulating CogPrime is that none of them was adequate on its own – leading us, for example, to the conclusion that *to deal with the problems facing a human-level AGI, an uncertain logic must integrate imprecise probability and fuzziness with a broad scope of logical constructs.* The arguments that both fuzziness and probability are needed seem hard to counter – these two notions of uncertainty are qualitatively different yet both appear cognitively necessary.

The argument for using probability in an AGI system is assailed by some AGI researchers such as Pei Wang, but we are swayed by the theoretical arguments in favor of probability theory's mathematically fundamental nature, as well as the massive demonstrated success of probability theory in various areas of narrow AI and applied science. However, we are also swayed by the arguments of Pei Wang, Peter Walley and others that using single-number probabilities to represent truth values leads to untoward complexities related to the tabulation and manipulation of amounts of evidence. This has led us to an imprecise probability based approach; and then technical arguments regarding the limitations of standard imprecise probability formalisms have led us to develop our own "indefinite probabilities" formalism.

The PLN logic framework is one way of integrating imprecise probability and fuzziness in a logical formalism that encompasses a broad scope of logical constructs. It integrates term logic and predicate logic – a feature that we consider not necessary, but very convenient, for AGI. Either predicate or term logic on its own would suffice, but each is awkward in certain cases, and integrating them as done in PLN seems to result in more elegant handling of real-world inference scenarios. Finally, PLN also integrates intensional inference in an elegant manner that demonstrates integrative intelligence – it defines intension using *pattern theory*, which binds inference to pattern recognition and hence to other cognitive processes in a conceptually appropriate way.

Clearly PLN is not the only possible logical formalism capable of serving a human-level AGI system; however, we know of no other existing, fleshed-out formalism capable of fitting the bill. In part this is because PLN has been developed as part of an integrative AGI project whereas other logical formalisms have mainly been developed for other purposes, or purely theoretically. Via using PLN to control virtual agents, and integrating PLN with other cognitive processes, we have tweaked and expanded the PLN formalism to serve all the roles required of the "declarative cognition" component of an AGI system with reasonable elegance and effectiveness.

8.5.2 Program Learning for Procedural Knowledge

Even more so than declarative knowledge, procedural knowledge is represented in many different ways in the AI literature. The human brain also apparently uses multiple mechanisms to embody different kinds of procedures. So the choice of how to represent procedures in an AGI system is not particularly obvious. However, there is one particular representation of procedures that is particularly well-suited for current computer systems, and particularly well-tested in this context: *programs*. In designing CogPrime , we have acted based on the understanding that *programs are a good way to represent procedures – including both cognitive and physical-action procedures, but perhaps not including low-level motor-control procedures.*

Of course, this begs the question of *programs in what programming language*, and in this context we have made a fairly traditional choice, using a special language called Combo that is essentially a minor variant of LISP, and supplying Combo with a set of customized primitives intended to reduce the length of the typical programs CogPrime needs to learn and use. What differentiates this use of LISP from many traditional uses of LISP in AI is that we are *only* using the LISP-ish representational style for procedural knowledge, rather than trying to use it for everything.

One test of whether the use of Combo programs to represent procedural knowledge makes sense is whether the procedures useful for a CogPrime system in everyday human environments have short Combo representations. We have worked with Combo enough to validate that they generally do in the virtual world environment – and also in the physical-world environment *if* lower-level motor procedures are supplied as primitives. That is, we are not convinced that Combo is a good representation for the procedure a robot needs to do to move its fingers to pick up a cup, coordinating its movements with its visual perceptions. It's certainly possible to represent this sort of thing in Combo, but Combo may be an awkward tool. However, if one represents low-level procedures like this using another method, e.g. learned cell assemblies in a hierarchical network like DeSTIN, then it's very feasible to make Combo programs that invoke these low-level procedures, and encode higher-level actions like "pick up the cup in front of you slowly and quietly, then hand it to Jim who is standing next to you."

Having committed to use programs to represent many procedures, the next question is how to learn programs. One key conclusion we have come to via our empirical work in this area is that some form of powerful *program normalization* is essential. Without normalization, it's too hard for existing learning algorithms to generalize from known, tested programs and draw useful uncertain conclusions about untested

ones. We have worked extensively with a generalization of Holman's "Elegant Normal Form" in this regard.

For learning normalized programs, we have come to the following conclusions:

- for relatively straightforward procedure learning problems, *hillclimbing with random restart and a strong Occam bias is an effective method*
- for more difficult problems that elude hillclimbing, *probabilistic evolutionary program learning is an effective method*

The probabilistic evolutionary program learning method we have worked with most in OpenCog is MOSES, and significant evidence has been gathered showing it to be dramatically more effective than genetic programming on relevant classes of problems. However, more work needs to be done to evaluate its progress on complex and difficult procedure learning problems. Alternate, related probabilistic evolutionary program learning algorithms such as PLEASURE have also been considered and may be implemented and tested as well.

8.5.3 Attention Allocation

There is significant evidence that the brain uses some sort of "activation spreading" type method to allocate attention, and many algorithms in this spirit have been implemented and utilized in the AI literature. So, we find ourselves in agreement with many others that *activation spreading is a reasonable way to handle attentional knowledge* (though other approaches, with greater overhead cost, may provide better accuracy and may be appropriate in some situations). We also agree with many others who have chosen **Hebbian learning as one route of learning associative relationships**, with more sophisticated methods such as information-geometric ones potentially also playing a role

Where CogPrime differs from standard practice is in the use of an economic metaphor to regulate activation spreading. In this matter CogPrime is broadly in agreement with Eric Baum's arguments about the value of economic methods in AI, although our specific use of economic methods is very different from his. Baum's work (e.g. Hayek) embodies more complex and computationally expensive uses of artificial economics, whereas we believe that *in the context of a neural-symbolic network, artificial economics is an effective approach to* **activation spreading**; and CogPrime's ECAN framework seeks to embody this idea. ECAN can also make use of more sophisticated and expensive uses of artificial currency when large amount of system resources are involved in a single choice, rendering the cost appropriate.

One major choice made in the CogPrime design is *to focus on two kinds of attention: processor (represented by ShortTermImportance) and memory (represented by LongTermImportance).* This is a direct reflection of one of the key differences between the von Neumann architecture and the human brain: in the former but not the latter, there is a strict separation between memory and processing in the underlying compute fabric. We carefully considered the possibility of using a larger variety of attention values, but for reasons of simplicity and computational efficiency we are currently using only STI and LTI in our OpenCogPrime implementation, with the possibility of extending further if experimentation proves it necessary.

8.5.4 Internal Simulation and Episodic Knowledge

For episodic knowledge, as with declarative and procedural knowledge, CogPrime has opted for a solution motivated by the particular strengths of contemporary digital computers. When the human brain runs through a "mental movie" of past experiences, it doesn't do any kind of accurate physical simulation of these experiences. But that's not because the brain wouldn't benefit from such – it's because the brain doesn't know how to do that sort of thing! On the other hand, any modern laptop can run a reasonable Newtonian physics simulation of everyday events, and more fundamentally can recall and manage the relative positions and movements of items in an internal 3D landscape paralleling remembered or imagined real-world events. With this in mind, we believe that in an AGI context, *simulation is a good way to handle episodic knowledge; and running an internal "world simulation engine" is an effective way to handle simulation.*

CogPrime can work with many different simulation engines; and since simulation technology is continually advancing independently of AGI technology, this is an area where AGI can buy some progressive advancement for free as time goes on. The subtle issues here regard interfacing between the simulation engine and the rest

of the mind: mining meaningful information out of simulations using pattern mining algorithms; and more subtly, figuring out what simulations to run at what times in order to answer the questions most relevant to the AGI system in the context of achieving its goals. We believe we have architected these interactions in a viable way in the CogPrime design, but we have tested our ideas in this regard only in some fairly simple contexts regarding virtual pets in a virtual world, and much more remains to be done here.

8.5.5 Low-Level Perception and Action

The centrality or otherwise of low-level perception and action in human intelligence is a matter of ongoing debate in the AI community. Some feel that the essence of intelligence lies in cognition and/or language, with perception and action having the status of "peripheral devices." Others feel that modeling the physical world and one's actions in it is the essence of intelligence, with cognition and language emerging as side-effects of these more fundamental capabilities. The CogPrime architecture doesn't need to take sides in this debate. Currently we are experimenting both in virtual worlds, and with real-world robot control. The value added by robotic versus virtual embodiment can thus be explored via experiment rather than theory, and may reveal nuances that no one currently foresees.

As noted above, we are unconfident of CogPrime's generic procedure learning or pattern recognition algorithms in terms of their capabilities to handle large amounts of raw sensorimotor data in real time, and so for robotic applications we advocate hybridizing CogPrime with a separate (but closely cross-linked) system better customized for this sort of data, in line with our general hypothesis that *Hybridization of one's integrative neural-symbolic system with a spatiotemporally hierarchical deep learning system is an effective way to handle representation and learning of low-level sensorimotor knowledge.* While this general principle doesn't depend on any particular approach, DeSTIN is one example of a deep learning system of this nature that can be effective in this context

We have not yet done any sophisticated experiments in this regard – our prior experiments using OpenCog to control robots have involved cruder integration of OpenCog with perceptual and motor subsystems, rather than the tight hybridization described in [Goe12] and envisioned for our future work. Creating such a hybrid system is "just" a matter of software engineering, but testing such a system may lead to many surprises!

8.5.6 Goals

Given that we have characterized general intelligence as "the ability to achieve complex goals in complex environments," it should be plain that goals play a central role in our work. However, we have chosen not to create a separate subsystem for intentional knowledge, and instead have concluded that *one effective way to handle goals is to represent them declaratively, and allocate attention among them economically.* An advantage of this approach is that it automatically provides integration between the goal system and the declarative and attentional knowledge systems.

Goals and subgoals are related using logical links as interpreted and manipulated by PLN, and attention is allocated among goals using the STI dynamics of ECAN, and a specialized variant described in [GPGtOT13] based on RFS's (requests for service). Thus the mechanics of goal management is handled using uncertain inference and artificial economics, whereas the figuring-out of how to achieve goals is done integratively, relying heavily on procedural and episodic knowledge as well as PLN and ECAN.

The combination of ECAN and PLN seems to overcome the well-known shortcomings found with purely neural-net or purely inferential approaches to goals. Neural net approaches generally have trouble with abstraction, whereas logical approaches are generally poor at real-time responsiveness and at tuning their details quantitatively based on experience. At least in principle, our hybrid approach overcomes all these shortcomings; though of current, it has been tested only in fairly simple cases in the virtual world.

8.6 Fulfilling the "Cognitive Equation"

A key claim based on the notion of the "Cognitive Equation" posited in *Chaotic Logic* [Goe94] is that *it is important for an intelligent system to have some way of recognizing large-scale patterns in itself, and then embodying these patterns as new, localized knowledge items in its memory.* This dynamic introduces a feedback dynamic between emergent pattern and substrate, which is hypothesized to be critical to general intelligence under feasible computational resources. It also ties in nicely with the notion of "glocal memory"

– essentially positing a localization of some global memories, which naturally will result in the formation of some glocal memories. One of the key ideas underlying the CogPrime design is that *given the use of a neural-symbolic network for knowledge representation, a graph-mining based "map formation" heuristic is one good way to do this.*

Map formation seeks to fulfill the Cognitive Equation quite directly, probably more directly than happens in the brain. Rather than relying on other cognitive processes to implicitly recognize overall system patterns and embody them in the system as localized memories (though this implicit recognition may also happen), the MapFormation MindAgent explicitly carries out this process. Mostly this is done using fairly crude greedy pattern mining heuristics, though if really subtle and important patterns seem to be there, more sophisticated methods like evolutionary pattern mining may also be invoked.

It seems possible that this sort of explicit approach could be less efficient than purely implicit approaches; but, there is no evidence for this, and it may also provided *increased* efficiency. And in the context of the overall CogPrime design, the explicit MapFormation approach seems most natural.

8.7 Occam's Razor

The key role of "Occam's Razor" or the urge for simplicity in intelligence has been observed by many before (going back at least to Occam himself, and probably earlier!), and is fully embraced in the CogPrime design. Our favored theoretical analysis of intelligence portrays intelligence as closely tied to the creation of procedures that achieve goals in environments *in the simplest possible way.* And this quest for simplicity is present in many places throughout the CogPrime design, for instance

- In MOSES and hillclimbing, where program compactness is an explicit component of program tree fitness
- In PLN, where the backward and forward chainers explicitly favor shorter proof chains, and intensional inference explicitly characterizes entities in terms of their patterns (where patterns are defined as compact characterizations)
- In pattern mining heuristics, which search for compact characterizations of data
- In the forgetting mechanism, which seeks the smallest set of Atoms that will allow the regeneration of a larger set of useful Atoms via modestly-expensive application of cognitive processes
- Via the encapsulation of procedural and declarative knowledge in simulations, which in many cases provide a vastly compacted form of storing real-world experiences

Like cognitive synergy and emergent networks, Occam's Razor is not something that is implemented in a single place in the CogPrime design, but rather an overall design principle that underlies nearly every part of the system.

8.8 Cognitive Synergy

To understand more specifically how cognitive synergy works in CogPrime, in the following subsections we will review some synergies related to the key components of CogPrime as discussed above. These synergies are absolutely critical to the proposed functionality of the CogPrime system. Without them, the cognitive mechanisms are not going to work adequately well, but are rather going to succumb to combinatorial explosions. The other aspects of CogPrime - the cognitive architecture, the knowledge representation, the embodiment framework and associated developmental teaching methodology - are all critical as well, but none of these will yield the critical emergence of intelligence without cognitive mechanisms that effectively scale. And, in the absence of cognitive mechanisms that effectively scale *on their own*, we must rely on cognitive mechanisms that *effectively help each other to scale.* The reasons why we believe these synergies will exist are essentially qualitative: we have not proved theorems regarded these synergies, and we have observed them in practice only in simple cases so far. However, we do have some ideas regarding how to potentially prove theorems related to these synergies, and some of these are described in [GPGtOT13].

8.8.1 Synergies that Help Inference

The combinatorial explosion in PLN is obvious: forward and backward chaining inference are both fundamentally explosive processes, reined in only by pruning heuristics. This means that for nontrivial complex inferences to occur, one needs **really, really clever** pruning heuristics. The CogPrime design combines simple heuristics with pattern mining, MOSES and economic attention allocation as pruning heuristics. Economic attention allocation assigns importance levels to Atoms, which helps guide pruning. Greedy pattern mining is used to search for patterns in the stored corpus of inference trees, to see if there are any that can be used as analogies for the current inference. And MOSES comes in when there is not enough information (from importance levels or prior inference history) to make a choice, yet exploring a wide variety of available options is unrealistic. In this case, MOSES tasks may be launched, pertinently to the leaves at the fringe of the inference tree, under consideration for expansion. For instance, suppose there is an Atom A at the fringe of the inference tree, and its importance hasn't been assessed with high confidence, but a number of items B are known so that:

```
MemberLink A B
```

Then, MOSES may be used to learn various relationships characterizing A, based on recognizing patterns across the set of B that are suspected to be members of A. These relationships may then be used to assess the importance of A more confidently, or perhaps to enable the inference tree to match one of the patterns identified by pattern mining on the inference tree corpus. For example, if MOSES figures out that:

```
SimilarityLink G A
```

then it may happen that substituting G in place of A in the inference tree, results in something that pattern mining can identify as being a good (or poor) direction for inference.

8.8.2 Synergies that Help MOSES

MOSES's combinatorial explosion is obvious: the number of possible programs of size N increases very rapidly with N. The only way to get around this is to utilize prior knowledge, and as much as possible of it. When solving a particular problem, the search for new solutions must make use of prior candidate solutions evaluated for that problem, and also prior candidate solutions (including successful and unsuccessful ones) evaluated for other related problems.

But, extrapolation of this kind is in essence a contextual analogical inference problem. In some cases it can be solved via fairly straightforward pattern mining; but in subtler cases it will require inference of the type provided by PLN. Also, attention allocation plays a role in figuring out, for a given problem A, which problems B are likely to have the property that candidate solutions for B are useful information when looking for better solutions for A.

8.8.3 Synergies that Help Attention Allocation

Economic attention allocation, without help from other cognitive processes, is just a very simple process analogous to "activation spreading" and "Hebbian learning" in a neural network. The other cognitive processes are the things that allow it to more sensitively understand the attentional relationships between different knowledge items (e.g. which sorts of items are often usefully thought about in the same context, and in which order).

8.8.4 Further Synergies Related to Pattern Mining

Statistical, greedy pattern mining is a simple process, but it nevertheless can be biased in various ways by other, more subtle processes.

For instance, if one has learned a population of programs via MOSES, addressing some particular fitness function, then one can study which items tend to be utilized in the same programs in this population. One may then direct pattern mining to find patterns combining these items found to be in the MOSES population. And conversely, relationships denoted by pattern mining may be used to probabilistically bias the models used within MOSES.

Statistical pattern mining may also help PLN by supplying it with information to work on. For instance, conjunctive pattern mining finds conjunctions of items, which may then be combined with each other using PLN, leading to the formation of more complex predicates. These conjunctions may also be fed to MOSES as part of an initial population for solving a relevant problem.

Finally, the main interaction between pattern mining and MOSES/PLN is that the former may recognize patterns in links created by the latter. These patterns may then be fed back into MOSES and PLN as data. This virtuous cycle allows pattern mining and the other, more expensive cognitive processes to guide each other. Attention allocation also gets into the game, by guiding statistical pattern mining and telling it which terms (and which combinations) to spend more time on.

8.8.5 Synergies Related to Map Formation

The essential synergy regarding map formation is obvious: Maps are formed based on the HebbianLinks created via PLN and simpler attentional dynamics, which are based on which Atoms are usefully used together, which is based on the dynamics of the cognitive processes doing the "using." On the other hand, once maps are formed and encapsulated, they feed into these other cognitive processes. This synergy in particular is critical to the emergence of self and attention.

What has to happen, for map formation to work well, is that the cognitive processes must *utilize* encapsulated maps in a way that gives rise overall to relatively clear clusters in the network of HebbianLinks. This will happen if the encapsulated maps are not too complex for the system's other learning operations to understand. So, there must be useful coordinated attentional patterns whose corresponding encapsulated-map Atoms are not too complicated. This has to do with the system's overall parameter settings, but largely with the settings of the attention allocation component. For instance, this is closely tied in with the limited size of "attentional focus" (the famous 7 +/- 2 number associated with humans' and other mammals short term memory capacity). If only a small number of Atoms are typically very important at a given point in time, then the maps formed by grouping together all simultaneously highly important things will be relatively small predicates, which will be easily reasoned about - thus keeping the "virtuous cycle" of map formation and comprehension going effectively.

8.9 Emergent Structures and Dynamics

We have spent much more time in this book on the engineering of cognitive processes and structures, than on the cognitive processes and structures that must *emerge* in an intelligent system for it to display human-level AGI. However, this focus should not be taken to represent a lack of appreciation for the importance of emergence. Rather, it represents a practical focus: engineering is what we must do to create a software system potentially capable of AGI, and emergence is then what happens inside the engineered AGI to allow it to achieve intelligence. Emergence must however be taken carefully into account when deciding what to engineer!

One of the guiding ideas underlying the CogPrime design is that *an AGI system with adequate mechanisms for handling the key types of knowledge mentioned above, and the capability to explicitly recognize large-scale pattern in itself, should* **upon sustained interaction with an appropriate environment in pursuit of appropriate goals**, *emerge a variety of complex structures in its internal knowledge network, including (but not limited to): a hierarchical network, representing both a spatiotemporal hierarchy and an approximate "default inheritance" hierarchy, cross-linked; a heterarchical network of associativity, roughly aligned with the hierarchical network; a self network which is an approximate micro image of the whole network; and inter-reflecting networks modeling self and others, reflecting a "mirrorhouse" design pattern.*

The dependence of these posited emergences on the environment and goals of the AGI system should not be underestimated. For instance, PLN and pattern mining don't have to lead to a hierarchical structured Atomspace. But if the AGI system is placed in an environment which it hierarchically structured via its own efforts, thenPLN and pattern mining very likely will lead to a hierarchically structured Atomspace. And if this environment consists of hierarchically structured *language and culture*, then what one has is a system of minds with hierarchical networks, each reinforcing the hierarchality of each others' networks. Similarly, integrated cognition doesn't have to lead to mirrorhouse structures, but integrated cognition about situations involving other minds studying and predicting and judging each other, is very likely to do

so. What is needed for appropriate emergent structures to arise in a mind, is mainly that the knowledge representation is sufficiently flexible to allow these structures, and the cognitive processes are sufficiently intelligent to observe these structures in the environment and then mirror them internally. Of course, it also doesn't hurt if the internal structures and processes are at least slightly *biased* toward the origination of the particular high-level emergent structures that are characteristic of the system's environment/goals; and this is indeed the case with CogPrime ... biases toward hierarchical, heterarchical, dual and mirrorhouse networks are woven throughout the system design, in a thoroughgoing though not extremely systematic way.

9 Measuring Incremental Progress Toward Human-Level AGI

We discussed, toward the start of this paper, various proposed tests for gauging achievement of human-level general intelligence. Perhaps more critical, however, is to think about how to measure incremental progress toward this goal. How do you tell when you're 25% or 50% of the way to having an AGI that can pass the Turing Test, or get an online university degree. Fooling 50% of the Turing Test judges is not a good measure of being 50% of the way to passing the Turing Test (that's too easy); and passing 50% of university classes is not a good measure of being 50% of the way to getting an online university degree (it's too hard – if one had an AGI capable of doing that, one would almost surely be very close to achieving the end goal). Measuring incremental progress toward human-level AGI is a subtle thing, and we argue that the best way to do it is to focus on particular scenarios and the achievement of specific competencies therein.

As we have argued in [GW11], there are some theoretical reasons to doubt the possibility of creating a rigorous objective test for partial progress toward AGI – a test that would be convincing to skeptics, and impossible to "game" via engineering a system specialized to the test. Fortunately, though, we don't need a test of this nature for the purposes of assessing our own incremental progress toward advanced AGI, based on our knowledge about our own approach.

Based on the nature of the grand goals articulated above, there seems to be a very natural approach to creating a set of incremental capabilities building toward AGI: *to draw on our copious knowledge about human cognitive development*. This is by no means the only possible path; one can envision alternatives that have nothing to do with human development (and those might also be better suited to non-human AGIs). However, so much detailed knowledge about human development is available – as well as solid knowledge that the human developmental trajectory does lead to human-level AI – that the motivation to draw on human cognitive development is quite strong.

The main problem with the human development inspired approach is that cognitive developmental psychology is not as systematic as it would need to be for AGI to be able to translate it directly into architectural principles and requirements. While early thinkers like Piaget and Vygotsky outlined systematic theories of child cognitive development, these are no longer considered fully accurate, and one currently faces a mass of detailed theories of various aspects of cognitive development, but without an unified understanding. Nevertheless we believe it is viable to work from the human-development data and understanding currently available, and craft a workable AGI roadmap therefrom.

In this vein, Sam Adams and his team at IBM have outlined a so-called "Toddler Turing Test," in which one seeks to use AI to control a robot qualitatively displaying similar cognitive behaviors to a young human child (say, a 3 year old) [AABL02]. In fact this sort of idea has a long and venerable history in the AI field – Alan Turing's original 1950 paper on AI [Tur50], where he proposed the Turing Test, contains the suggestion that

> *"Instead of trying to produce a programme to simulate the adult mind, why not rather try to produce one which simulates the child's?"*

We find this childlike cognition based approach promising for many reasons, including its integrative nature: what a young child does involves a combination of perception, actuation, linguistic and pictorial communication, social interaction, conceptual problem solving and creative imagination. Specifically, inspired by these ideas, in [GB09] we have suggested the approach of teaching and testing early-stage AGI systems in environments that emulate the preschools used for teaching human children.

Human intelligence evolved in response to the demands of richly interactive environments, and a preschool is specifically designed to be a richly interactive environment with the capability to stimulate diverse mental

growth. So, we are currently exploring the use of CogPrime to control virtual agents in preschool-like virtual world environments, as well as commercial humanoid robot platforms such as the Nao or Robokind in physical preschool-like robot labs.

Another advantage of focusing on childlike cognition is that child psychologists have created a variety of instruments for measuring child intelligence. In [GPGtOT13], we present some details of an approach to evaluating the general intelligence of human childlike AGI systems via combining tests typically used to measure the intelligence of young human children, with additional tests crafted based on cognitive science and the standard preschool curriculum. According to this approach, we don't suggest to place a CogPrime system in an environment that is an exact imitation of a human preschool – this would be inappropriate since current robotic or virtual bodies are very differently abled than the body of a young human child. But we aim to place CogPrime in an environment emulating the basic diversity and educational character of a typical human preschool.

9.1 Competencies and Tasks on the Path to Human-Level AI

With this preschool focus in mind, we give in this subsection a fairly comprehensive list of the competencies that we feel AI systems should be expected to display in one or more of these scenarios in order to be considered as full-fledged "human level AGI" systems. These competency areas have been assembled somewhat opportunistically via a review of the cognitive and developmental psychology literature as well as the scope of the current AI field. We are not claiming this as a precise or exhaustive list of the competencies characterizing human-level general intelligence, and will be happy to accept additions to the list, or mergers of existing list items, etc. What we are advocating is not this specific list, but rather the approach of enumerating competency areas, and then generating tasks by combining competency areas with scenarios.

We also give, with each competency, an example task illustrating the competency. The tasks are expressed in the robot preschool context for concreteness, but they all apply to the virtual preschool as well. Of course, these are only examples, and ideally to teach an AGI in a structured way one would like to

- associate several tasks with each competency
- present each task in a graded way, with multiple subtasks of increasing complexity
- associate a quantitative metric with each task

However, the briefer treatment given here should suffice to give a sense for how the competencies manifest themselves practically in the AGI Preschool context.

1. Perception
    - **Vision**: image and scene analysis and understanding
        - *Example task:* When the teacher points to an object in the preschool, the robot should be able to identify the object and (if it's a multi-part object) its major parts. If it can't perform the identification initially, it can approach the object and manipulate it before making its identification.
    - **Hearing**: identifying the sounds associated with common objects; understanding which sounds come from which sources in a noisy environment
        - *Example task:* When the teacher covers the robot's eyes and then makes a noise with an object, the robot should be able to guess what the object is
    - **Touch**: identifying common objects and carrying out common actions using touch alone
        - *Example task:* With its eyes and ears covered, the robot should be able to identify some object by manipulating it; and carry out some simple behaviors (say, putting a block on a table) via touch alone
    - **Crossmodal**: Integrating information from various senses
        - *Example task:* Identifying an object in a noisy, dim environment via combining visual and auditory information

- **Proprioception**: Sensing and understanding what its body is doing
  - *Example task:* The teacher moves the robot's body into a certain configuration. The robot is asked to restore its body to an ordinary standing position, and then repeat the configuration that the teacher moved it into.

2. Actuation

- **Physical skills**: manipulating familiar and unfamiliar objects
  - *Example task:* Manipulate blocks based on imitating the teacher: e.g. pile two blocks atop each other, lay three blocks in a row, etc.
- **Tool use**, including the flexible use of ordinary objects as tools
  - *Example task:* Use a stick to poke a ball out of a corner, where the robot cannot directly reach
- **Navigation**, including in complex and dynamic environments
  - *Example task:* Find its own way to a named object or person through a crowded room with people walking in it and objects laying on the floor.

3. Memory

- **Declarative**: noticing, observing and recalling facts about its environment and experience
  - *Example task:* If certain people habitually carry certain objects, the robot should remember this (allowing it to know how to find the objects when the relevant people are present, even much later)
- **Behavioral**: remembering how to carry out actions
  - *Example task:* If the robot is taught some skill (say, to fetch a ball), it should remember this much later
- **Episodic**: remembering significant, potentially useful incidents from life history
  - *Example task:* Ask the robot about events that occurred at times when it got particularly much, or particularly little, reward for its actions; it should be able to answer simple questions about these, with significantly more accuracy than about events occurring at random times

4. Learning

- **Imitation**: Spontaneously adopt new behaviors that it sees others carrying out
  - *Example task:* Learn to build towers of blocks by watching people do it
- **Reinforcement**: Learn new behaviors from positive and/or negative reinforcement signals, delivered by teachers and/or the environment
  - *Example task:* Learn which box the red ball tends to be kept in, by repeatedly trying to find it and noticing where it is, and getting rewarded when it finds it correctly
- **Imitation/Reinforcement**
  - *Example task:* Learn to play "fetch", "tag" and "follow the leader" by watching people play it, and getting reinforced on correct behavior
- **Interactive Verbal Instruction**
  - *Example task:* Learn to build a particular structure of blocks faster based on a combination of imitation, reinforcement and verbal instruction, than by imitation and reinforcement without verbal instruction
- **Written Media**
  - *Example task:* Learn to build a structure of blocks by looking at a series of diagrams showing the structure in various stages of completion

- **Learning via Experimentation**
  - *Example task:* Ask the robot to slide blocks down a ramp held at different angles. Then ask it to make a block slide fast, and see if it has learned how to hold the ramp to make a block slide fast.

5. Reasoning

- **Deduction**, from uncertain premises observed in the world
  - *Example task:* If Ben more often picks up red balls than blue balls, and Ben is given a choice of a red block or blue block to pick up, which is he more likely to pick up?
- **Induction**, from uncertain premises observed in the world
  - *Example task:* If Ben comes into the lab every weekday morning, then is Ben likely to come to the lab today (a weekday) in the morning?
- **Abduction**, from uncertain premises observed in the world
  - *Example task:* If women more often give the robot food than men, and then someone of unidentified gender gives the robot food, is this person a man or a woman?
- **Causal reasoning**, from uncertain premises observed in the world
  - *Example task:* If the robot knows that knocking down Ben's tower of blocks makes him mad, then what will it say when asked if kicking the ball at Ben's tower of blocks will make Ben mad?
- **Physical reasoning**, based on observed "fuzzy rules" of naive physics
  - *Example task:* Given two balls (one rigid and one compressible) and two tunnels (one significantly wider than the balls, one slightly narrower than the balls), can the robot guess which balls will fit through which tunnels?
- **Associational reasoning**, based on observed spatiotemporal associations
  - *Example task:* If Ruiting is normally seen near Shuo, then if the robot knows where Shuo is, that is where it should look when asked to find Ruiting

6. Planning

- **Tactical**
  - *Example task:* The robot is asked to bring the red ball to the teacher, but the red ball is in the corner where the robot can't reach it without a tool like a stick. The robot knows a stick is in the cabinet so it goes to the cabinet and opens the door and gets the stick, and then uses the stick to get the red ball, and then brings the red ball to the teacher.
- **Strategic**
  - *Example task:* Suppose that Matt comes to the lab infrequently, but when he does come he is very happy to see new objects he hasn't seen before (and suppose the robot likes to see Matt happy). Then when the robot gets a new object Matt has not seen before, it should put it away in a drawer and be sure not to lose it or let anyone take it, so it can show Matt the object the next time Matt arrives.
- **Physical**
  - *Example task:* To pick up a cup with a handle which is lying on its side in a position where the handle can't be grabbed, the robot turns the cup in the right position and then picks up the cup by the handle
- **Social**
  - *Example task:* The robot is given a job of building a tower of blocks by the end of the day, and he knows Ben is the most likely person to help him, and he knows that Ben is more likely to say "yes" to helping him when Ben is alone. He also knows that Ben is less likely to say yes if he's asked too many times, because Ben doesn't like being nagged. So he waits to ask Ben till Ben is alone in the lab.

7. Attention

   - **Visual Attention** within its observations of its environment
     - *Example task:* The robot should be able to look at a scene (a configuration of objects in front of it in the preschool) and identify the key objects in the scene and their relationships.
   - **Social Attention**
     - *Example task:* The robot is having a conversation with Itamar, which is giving the robot reward (for instance, by teaching the robot useful information). Conversations with other individuals in the room have not been so rewarding recently. But Itamar keeps getting distracted during the conversation, by talking to other people, or playing with his cellphone. The robot needs to know to keep paying attention to Itamar even through the distractions.
   - **Behavioral Attention**
     - *Example task:* The robot is trying to navigate to the other side of a crowded room full of dynamic objects, and many interesting things keep happening around the room. The robot needs to largely ignore the interesting things and focus on the movements that are important for its navigation task.

8. Motivation

   - **Subgoal creation**, based on its preprogrammed goals and its reasoning and planning
     - *Example task:* Given the goal of pleasing Hugo, can the robot learn that telling Hugo facts it has learned but not told Hugo before, will tend to make Hugo happy?
   - **Affect-based motivation**
     - *Example task:* Given the goal of gratifying its curiosity, can the robot figure out that when someone it's never seen before has come into the preschool, it should watch them because they are more likely to do something new?
   - **Control of emotions**
     - *Example task:* When the robot is very curious about someone new, but is in the middle of learning something from its teacher (who it wants to please), can it control its curiosity and keep paying attention to the teacher?

9. Emotion

   - **Expressing Emotion**
     - *Example task:* Cassio steals the robot's toy, but Ben gives it back to the robot. The robot should appropriately display anger at Cassio, and gratitude to Ben.
   - **Understanding Emotion**
     - *Example task:* Cassio and the robot are both building towers of blocks. Ben points at Cassio's tower and expresses happiness. The robot should understand that Ben is happy with Cassio's tower.

10. Modeling Self and Other

    - **Self-Awareness**
      - *Example task:* When someone asks the robot to perform an act it can't do (say, reaching an object in a very high place), it should say so. When the robot is given the chance to get an equal reward for a task it can complete only occasionally, versus a task it finds easy, it should choose the easier one.
    - **Theory of Mind**

- *Example task:* While Cassio is in the room, Ben puts the red ball in the red box. Then Cassio leaves and Ben moves the red ball to the blue box. Cassio returns and Ben asks him to get the red ball. The robot is asked to go to the place Cassio is about to go.

- **Self-Control**
  - *Example task:* Nasty people come into the lab and knock down the robot's towers, and tell the robot he's a bad boy. The robot needs to set these experiences aside, and not let them impair its self-model significantly; it needs to keep on thinking it's a good robot, and keep building towers (that its teachers will reward it for).
- **Other-Awareness**
  - *Example task:* If Ben asks Cassio to carry out a task that the robot knows Cassio cannot do or does not like to do, the robot should be aware of this, and should bet that Cassio will not do it.
- **Empathy**
  - *Example task:* If Itamar is happy because Ben likes his tower of blocks, or upset because his tower of blocks is knocked down, the robot should express and display these same emotions

11. Social Interaction

- **Appropriate Social Behavior**
  - *Example task:* The robot should learn to clean up and put away its toys when it's done playing with them.
- **Social Communication**
  - *Example task:* The robot should greet new human entrants into the lab, but if it knows the new entrants very well and it's busy, it may eschew the greeting
- **Social Inference** about simple social relationships
  - *Example task:* The robot should infer that Cassio and Ben are friends because they often enter the lab together, and often talk to each other while they are there
- **Group Play** at loosely-organized activities
  - *Example task:* The robot should be able to participate in "informally kicking a ball around" with a few people, or in informally collaboratively building a structure with blocks

12. Communication

- **Gestural communication** to achieve goals and express emotions
  - *Example task:* If the robot is asked where the red ball is, it should be able to show by pointing its hand or finger
- **Verbal communication** using English in its life-context
  - *Example tasks:* Answering simple questions, responding to simple commands, describing its state and observations with simple statements
- **Pictorial Communication** regarding objects and scenes it is familiar with
  - *Example task:* The robot should be able to draw a crude picture of a certain tower of blocks, so that e.g the picture looks different for a very tall tower and a wide low one
- **Language acquisition**
  - *Example task:* The robot should be able to learn new words or names via people uttering the words while pointing at objects exemplifying the words or names
- **Cross-modal communication**
  - *Example task:* If told to "touch Bob's knee" but the robot doesn't know what a knee is, being shown a picture of a person and pointed out the knee in the picture should help it figure out how to touch Bob's knee

13. Quantitative

    - **Counting** sets of objects in its environment
      - *Example task:* The robot should be able to count small (homogeneous or heterogeneous) sets of objects
    - **Simple, grounded arithmetic** with small numbers
      - *Example task:* Learning simple facts about the sum of integers under 10 via teaching, reinforcement and imitation
    - **Comparison** of observed entities regarding quantitative properties
      - *Example task:* Ability to answer questions about which object or person is bigger or taller
    - **Measurement** using simple, appropriate tools
      - *Example task:* Use of a yardstick to measure how long something is

14. Building/Creation

    - **Physical**: creative constructive play with objects
      - *Example task:* Ability to construct novel, interesting structures from blocks
    - **Conceptual** invention: concept formation
      - *Example task:* Given a new category of objects introduced into the lab (e.g. hats, or pets), the robot should create a new internal concept for the new category, and be able to make judgments about these categories (e.g. if Ben particularly likes pets, it should notice this after it has identified "pets" as a category)
    - **Verbal** invention
      - *Example task:* Ability to coin a new word or phrase to describe a new object (e.g. the way Alex the parrot coined "bad cherry" to refer to a tomato)
    - **Social**
      - *Example task:* If the robot wants to play a certain activity (say, practicing soccer), it should be able to gather others around to play with it

Based on these competencies and associated tasks, it is not hard to articulate a specific roadmap for progress toward human-level AGI, inspired by human child development. This sort of roadmap does not give a highly rigorous, objective way of assessing the percentage of progress toward the end-goal of human-level AGI. However, it gives a much better sense of progress than one would have otherwise. For instance, if an AGI system performed well on diverse metrics corresponding to 50% of the competency areas listed above, one would seem justified in claiming to have made very substantial progress toward human-level AGI. If an AGI system performed well on diverse metrics corresponding to 90% of these competency areas, one would seem justified in claiming to be "almost there." Achieving, say, 25% of the metrics would give one a reasonable claim to "interesting AGI progress." This kind of qualitative assessment of progress is not the most one could hope for, but again, it is better than the progress indications one could get *without* this sort of roadmap.

10 A CogPrime Thought Experiment: Build Me Something I Haven't Seen Before

AGI design necessarily leads one into some rather abstract spaces – but being a human-like intelligence in the everyday world is a pretty concrete thing. If the CogPrime research program is successful, it will result not just in abstract ideas and equations, but rather in real AGI robots carrying out tasks in the everyday human world; and AGI agents in virtual worlds and online digital spaces conducting important business, doing science, entertaining and being entertained by us, and so forth. With this in mind, in this final chapter

we will bring the discussion closer to the concrete and everyday, and pursue a thought experiment of the form "How would a completed CogPrime system carry out this specific task?"

The task we will use for this thought-experiment is one we have sometimes used as a running example in our internal technical discussions within the OpenCog team, and have briefly mentioned above. We consider the case of a robotically or virtually embodied CogPrime system, operating in a preschool type environment, interacting with a human whom it already knows and given the task of "Build me something with blocks that I haven't seen before."

This target task is fairly simple, but it is complex enough to involve essentially every one of CogPrime 's processes, interacting in a unified way.

We consider the case of a simple interaction based on the above task where:

1. The human teacher tells the CogPrime agent "Build me something with blocks that I haven't seen before."
2. After a few false starts, the agent builds something it thinks is appropriate and says "Do you like it?"
3. The human teacher says "It's beautiful. What is it?"
4. The agent says "It's a car man" [and indeed, the construct has 4 wheels and a chassis vaguely like a car, but also a torso, arms and head vaguely like a person]

Of course, a complex system like CogPrime could carry out an interaction like this internally in many different ways, and what is roughly described here is just one among many possibilities.

In [GPGtOT13], this example is discussed more thoroughly, via enumerating a number of CogPrime processes and explaining some ways that each one may help CogPrime carry out the target task. Here, instead, we give more evocative "semi-narrative" conveying the dynamics that would occur in CogPrime while carrying out the target task, and mentioning how each of the enumerated cognitive processes as it arises in the narrative.

The reason we call this a semi-narrative rather than a narrative, is that there is no particular linear order to the processes occurring in each phase of the situation described here. CogPrime's internal cognitive processes do not occur in a linear narrative; rather, what we have is a complex network of interlocking events. But still, describing some of these events concretely in a manner correlated with the different stages of a simple interaction, may have some expository value.

10.1 Let the Semi-Narrative Begin...

The human teacher tells the CogPrime agent "Build me something with blocks that I haven't seen before."

Upon hearing this, the agent's cognitive cycles are dominated by language processing and retrieval from episodic and sensory memory.

The agent may decide to revive from disk the mind-states it went through when building human-pleasing structures from blocks before, so as to provide it with guidance

It will likely experience the emotion of happiness, because it anticipates the pleasure of getting rewarded for the task in future.

The ubergoal (pre-programmed, top-level goal) of pleasing the teacher gets active (gets funded significantly with STI currency), as there it becomes apparent there are fairly clear ways of fulfilling that goal (via the subgoal S of building blocks structures that will get positive response from the teacher). Other ubergoals like gaining knowledge are not funded as much with STI currency just now, as they are not immediately relevant.

Action selection, based on ImplicationLinks derived via PLN (between various possible activities and the subgoal S) causes it to start experimentally building some blocks structures. Past experience with building (turned into ImplicationLinks via mining the SystemActivityTable) tells it that it may want to build a little bit in its internal simulation world before building in the external world, causing STI currently to flow to the simulation MindAgent.

The Atom corresponding to the context blocks-building gets high STI and is pushed into the AttentionalFocus (the set of Atoms with the highest ShortTermImportance values), making it likely that many future

inferences will occur in this context. Other Atoms related to this one also get high STI (the ones in the blocks-building map, and others that are especially related to blocks-building in this particular context).

After a few false starts, the agent builds something it thinks is appropriate and says "Do you like it?"

Now that the agent has decided what to do to fulfill its well-funded goal, its cognitive cycles are dominated by action, perception and related memory access and concept creation.

An obvious subgoal is spawned: build a new structure now, and make this particular structure under construction appealing and novel to the teacher. This subgoal has a shorter time scale than the high level goal. The subgoal gets some currency from its supergoal using the mechanism of RFS spreading.

Action selection must tell it when to continue building the same structure and when to try a new one, as well as more micro level choices

Atoms related to the currently pursued blocks structure get high STI.

After a failed structure (a false start) is disassembled, the corresponding Atoms lose STI dramatically (leaving AF) but may still have significant LTI, so they can be recalled later as appropriate. They may also have VLTI so they will be saved to disk later on if other things push them out of RAM due to getting higher LTI.

Meanwhile everything that's experienced from the external world goes into the ExperienceDB.

Atoms representing different parts of aspects of the same blocks structure will get Hebbia Links between them, which will guide future reasoning and importance spreading.

Importance spreading helps the system go from an idea for something to build (say, a rock or a car) to the specific plans and ideas about how to build it, via increasing the STI of the Atoms that will be involved in these plans and ideas.

If something apparently good is done in building a blocks structure, then other processes and actions that helped lead to or support that good thing, get passed some STI from the Atoms representing the good thing, and also may get linked to the Goal Atom representing good in this context. This leads to reinforcement learning.

The agent may play with building structures and then seeing what they most look like, thus exercising abstract object recognition (that uses procedures learned by MOSES or hillclimbing, or uncertain relations learned by inference, to guess what object category a given observed collection of percepts most likely falls into).

Since the agent has been asked to come up with something surprising, it knows it should probably try to formulate some new concepts. Because it has learned in the past, via SystemActivityTable mining, that often newly formed concepts are surprising to others. So, more STI currency is given to concept formation MindAgents, such as the ConceptualBlending MindAgent (which, along with a lot of stuff that gets thrown out or stored for later use, comes up with car-man).

When the notion of car is brought to mind, the distributed map of nodes corresponding to car get high STI. When car-man is formed, it is reasoned about (producing new Atoms), but it also serves as a nexus of importance-spreading, causing the creation of a distributed car-man map.

If the goal of making an arm for a man-car occurs, then goal-driven schema learning may be done to learn a procedure for arm-making (where the actual learning is done by MOSES or hill-climbing).

If the agent is building a man-car, it may have man-building and car-building schema in its ActiveSchemaPool at the same time, and SchemaActivation may spread back and forth between the different modules of these two schema.

If the agent wants to build a horse, but has never seen a horse made of blocks (only various pictures and movies of horses), it may uses MOSES or hillclimbing internally to solve the problem of creating a horse-recognizer or a horse-generator which embodies appropriate abstract properties of horses. Here as in all cases of procedure learning, a complexity penalty rewards simpler programs, from among all programs that approximately fulfill the goals of the learning process.

If a procedure being executed has some abstract parts, then these may be executed by inferential procedure evaluation (which makes the abstract parts concrete on the fly in the course of execution).

To guess the fitness of a procedure for doing something (say, building an arm or recognizing a horse), inference or simulation may be used, as well as direct evaluation in the world.

Deductive, inductive and abductive PLN inference may be used in figuring out what a blocks structure will look or act like like before building it (it's tall and thin so it may fall down; it won't be bilaterally

symmetric so it won't look much like a person; etc.)

Backward-chaining inference control will help figure out how to assemble something matching a certain specification e.g. how to build a chassis based on knowledge of what a chassis looks like. Forward chaining inference (critically including intensional relationships) will be used to estimate the properties that the teacher will perceive a given specific structure to have. Spatial and temporal algebra will be used extensively in this reasoning, within the PLN framework.

Coordinating different parts of the body say an arm and a hand will involve importance spreading (both up and down) within the hierarchical action network, and from this network to the hierarchical perception network and the heterarchical cognitive network.

In looking up Atoms in the AtomSpace, some have truth values whose confidences have decayed significantly (e.g. those regarding the teacher's tastes), whereas others have confidences that have hardly decayed at all (e.g. those regarding general physical properties of blocks).

Finding previous blocks structures similar to the current one (useful for guiding building by analogy to past experience) may be done rapidly by searching the system's internal dimensional-embedding space.

As the building process occurs, patterns mined via past experience (tall things often fall down) are used within various cognitive processes (reasoning, procedure learning, concept creation, etc.); and new pattern mining also occurs based on the new observations made as different structures are build and experimented with and destroyed.

Simulation of teacher reactions, based on inference from prior examples, helps with the evaluation of possible structures, and also of procedures for creating structures.

As the agent does all this, it experiences the emotion of curiosity (likely among other emotions), because as it builds each new structure it has questions about what it will look like and how the teacher would react to it.

The human teacher says "It's beautiful. What is it?" The agent says "It's a car man"

Now that the building is done and the teacher says something, the agent's cognitive cycles are dominated by language understanding and generation. The Atom representing the context of talking to the teacher gets high STI, and is used as the context for many ensuing inferences.

Comprehension of "it" uses anaphor resolution based on a combination of ECAN and PLN inference based on a combination of previously interpreted language and observation of the external world situation.

The agent experiences the emotion of happiness because the teacher has called its creation beautiful, which is recognizes as a positive evaluation so the agent knows one of its ubergoals ("please the teacher") has been significantly fulfilled.

The goal of pleasing the teacher causes the system to want to answer the question. So the QuestionAnswering DialogueController schema gets paid a lot and gets put into the ActiveSchemaPool. In reaction to the question asked, this DC chooses a semantic graph to speak, then invokes NL generation to say it.

NL generation chooses the most compact expression that seems to adequately convey the intended meaning, so it decides on "car man" as the best simple verbalization to match the newly created conceptual blend that it thinks effectively describes the newly created blocks structure.

The positive feedback from the user leads to reinforcement of the Atoms and processes that led to the construction of the blocks structure that has been judged beautiful (via importance spreading and SystemActivityTable mining).

10.2 Conclusion

The simple situation considered in this section is complex enough to involve nearly all the different cognitive processes in the CogPrime system – and many interactions between these processes. This fact illustrates one of the main difficulties of designing, building and testing an artificial mind like CogPrime – until nearly all of the system is built and made to operate in an integrated way, it's hard to do any meaningful test of the system. Testing PLN or MOSES or conceptual blending in isolation may be interesting computer science, but it doesn't tell you much about CogPrime as a design for a thinking machine.

According to the CogPrime approach, getting a simple child-like interaction like "build me something with blocks that I haven't seen before" to work properly requires a holistic, integrated cognitive system. Once one has built a system capable of this sort of simple interaction then, according to the theory underlying CogPrime, one is not that far from a system with adult human-level intelligence. Of course there will be a

lot of work to do to get from a child-level system to an adult-level system – it won't necessarily unfold as "automatically" as seems to happen with a human child, because CogPrime lacks the suite of developmental processes and mechanisms that the young human brain has. But still, a child CogPrime mind capable of doing the things outlined in this chapter will have all the basic components and interactions in place, all the ones that are needed for a much more advanced artificial mind.

Of course, one could concoct a narrow-AI system carrying out the specific activities described in this section, much more simply than one could build a CogPrime system capable of doing these activities. But that's not the point – the point of this section has been not to explain how to achieve some particular narrow set of activities "by any means necessary", but rather to explain how these activities might be achieved within the CogPrime framework, which has been designed with much more generality in mind.

It would be worthwhile to elaborate a number of other situations similar to the one described in this chapter, and to work through the various cognitive processes and structures in CogPrime carefully in the context of each of these situations. In fact this sort of exercise has frequently been carried out informally in the context of developing CogPrime. But this paper is already long enough, so we will end here, and leave the rest for future works – emphasizing that it is via intimate interplay between concrete considerations like the ones presented in this section, and general algorithmic and conceptual considerations, that we have the greatest hope of creating advanced AGI. The value of this sort of interplay actually follows from the theory of real-world general intelligence indicated above. Thoroughly general intelligence is only possible given unrealistic computational resources, so real-world general intelligence is about achieving high generality given limited resources relative to the specific classes of environments relevant to a given agent. Specific situations like building surprising things with blocks are particularly important insofar as they embody broader information about the classes of environments relevant to broadly *human-like* general intelligence.

No doubt, once a CogPrime system is completed, the specifics of its handling of the situation described here will differ somewhat from the treatment presented in this chapter. Furthermore, the final CogPrime system may differ algorithmically and structurally in some respects from the specifics outlined here – it would be surprising if the process of building, testing and interacting with CogPrime didn't teach us some new things about various of the topics covered. But our conjecture is that, if sufficient effort is deployed appropriately, then a system much like the CogPrime system here will be able to handle the situation described in this section in a roughly similar manner to the one described in this chapter – and that this will serve as a natural precursor to much more dramatic AGI achievements.

11 Broader Issues

Current, practical AGI development is at a quite early stage – we are still struggling to get our AGI systems to deal with basic situations like creative blocks play, which human children and even apes handle easily. This is not surprising, as when one takes a design like CogPrime and gradually spells it out into a specific software system, one obtains a large and complex system, which requires at least dozens of expert human-years to refine, implement and test. Much simpler kinds of software, such as word processors, games or operating systems, often require dozens to hundreds of expert human years for their realization.

Even at this early stage, however, it is worthwhile to pay attention to the broader issues related to AGI development, such as the potential for escalation of AGI beyond the human level, and the ethical implications of human level or more advanced AGI. Given the reality of exponential technological acceleration [Kur06], it is quite possible for an area of technology to progress from the early to advanced stages in a brief number of years.

11.1 Ethical AGI

Creating an AGI with guaranteeably ethical behavior seems an infeasible task; but of course, no human is guaranteeably ethical either, and in fact it seems almost guaranteed that in any moderately large group of humans there are going to be some with strong propensities for extremely unethical behaviors, according to any of the standard human ethical codes. One of our motivations in developing CogPrime has been the belief that *an AGI system, if supplied with a commonsensically ethical goal system and an intentional component based largely on rigorous uncertain inference, should be able to reliably achieve a much higher level of commonsensically ethical behavior than any human being.*

Our explorations in the detailed design of CogPrime's goal system have done nothing to degrade this belief. While we have not yet developed any CogPrime system to the point where experimenting with its ethics is meaningful, based on our understanding of the current design it seems to us that

- a typical CogPrime system will display a much more consistent and less conflicted and confused motivational system than any human being, due to its explicit orientation toward carrying out actions that (based on its knowledge) rationally seem most likely to lead to achievement of its goals
- if a CogPrime system is given goals that are consistent with commonsensical human ethics (say, articulated in natural language), and then educated in an ethics-friendly environment such as a virtual or physical school, then it is reasonable to expect the CogPrime system will ultimately develop an advanced (human adult level or beyond) form of commmonsensical human ethics

Human ethics is itself wracked with inconsistencies, so one cannot expect a rationality-based AGI system to precisely mirror the ethics of any particular human individual or cultural system. But given the degree to which general intelligence represents adaptation to its environment, and interpretation of natural language depends on life history and context, it seems very likely to us that a CogPrime system, if supplied with a human-commonsense-ethics based goal system and then raised by compassionate and intelligent humans in a school-type environment, would arrive at its own variant of human-commonsense-ethics. The AGI system's ethics would then interact with human ethical systems in complex ways, leading to ongoing evolution of both systems and the development of new cultural and ethical patterns. Predicting the future is difficult even in the absence of radical advanced technologies, but our intuition is that this path has the potential to lead to beneficial outcomes for both human and machine intelligence.

11.2 Toward Superhuman General Intelligence

Human-level AGI is a difficult goal, relative to the current state of scientific understanding and engineering capability, and most of our work on CogPrime has been focused on our ideas about how to achieve it. However, we also intuitively suspect the CogPrime architecture has the ultimate potential to push beyond the human level in many ways. As part of this suspicion we advance the claim that *once sufficiently advanced, a CogPrime system should be able to radically self-improve via a variety of methods, including supercompilation and automated theorem-proving.*

Supercompilation allows procedures to be automatically replaced with equivalent but massively more time-efficient procedures. This is particularly valuable in that it allows AI algorithms to learn new procedures without much heed to their efficiency, since supercompilation can always improve the efficiency afterwards. So it is a real boon to automated program learning.

Theorem-proving is difficult for current narrow-AI systems, but for an AGI system with a deep understanding of the context in which each theorem exists, it should be much easier than for human mathematicians. So we envision that ultimately an AGI system will be able to design itself new algorithms and data structures via proving theorems about which ones will best help it achieve its goals in which situations, based on mathematical models of itself and its environment. Once this stage is achieved, it seems that machine intelligence may begin to vastly outdo human intelligence, leading in directions we cannot now envision.

While such projections may seem science-fictional, we note that the CogPrime architecture explicitly supports such steps. If human-level AGI is achieved within the CogPrime framework, it seems quite feasible that profoundly self-modifying behavior could be achieved fairly shortly thereafter. For instance, one could take a human-level CogPrime system and teach it computer science and mathematics, so that it fully understood the reasoning underlying its own design, and the whole mathematics curriculum leading up the the algorithms underpinning its cognitive processes.

11.3 Conclusion

What we have sought to do in this lengthy review paper is, mainly,

- to articulate a theoretical perspective on general intelligence, according to which the creation of a human-level AGI doesn't require anything *that* extraordinary, but "merely" an appropriate combination

of closely interoperating algorithms operating on an appropriate multi-type memory system, utilized to enable a system in an appropriate body and environment to figure out how to achieve its given goals

- to describe a software design (CogPrime) that, according to this somewhat mundane but theoretically quite well grounded vision of general intelligence, appears likely (according to a combination of rigorous and heuristic arguments) to be able to lead to human-level AGI using feasible computational resources
- to describe some of the preliminary lessons we've learned via implementing and experimenting with aspects of the CogPrime design, in the OpenCog system

We wish to stress that *not all of our arguments and ideas need to be 100% correct in order for the project to succeed*. The quest to create AGI is a mix of theory, engineering, and scientific and unscientific experimentation. If the current CogPrime design turns out to have significant shortcomings, yet still brings us a significant percentage of the way toward human-level AGI, the results obtained along the path will very likely give us clues about how to tweak the design to more effectively get the rest of the way there. And the OpenCog platform is extremely flexible and extensible, rather than being tied to the particular details of the CogPrime design. While we do have faith that the CogPrime design as described here has human-level AGI potential, we are also pleased to have a development strategy and implementation platform that will allow us to modify and improve the design in accordance with whatever suggestions are made by our ongoing experimentation.

Many great achievements in history have seemed more magical before their first achievement than afterwards. Powered flight and spaceflight are the most obvious examples, but there are many others such as mobile telephony, prosthetic limbs, electronically deliverable books, robotic factory workers, and so on. We now even have wireless transmission of power (one can recharge cellphones via wifi), though not yet as ambitiously as Tesla envisioned. We very strongly suspect that human-level AGI is in the same category as these various examples: an exciting and amazing achievement, which however is achievable via systematic and careful application of fairly mundane principles. We believe computationally feasible human-level intelligence is both *complicated* (involving many interoperating parts, each sophisticated in their own right) and *complex* (in the sense of involving many emergent dynamics and structures whose details are not easily predictable based on the parts of the system) ... but that neither the complication nor the complexity is an obstacle to engineering human-level AGI.

In our view, what is needed to create human-level AGI is not a new scientific breakthrough, nor a miracle, but "merely" a sustained effort over a number of years by a moderate-sized team of appropriately-trained professionals, completing the implementation of an adequate design – such as the one described in [GPGtOT13] and roughly sketched here – and then parenting and educating the resulting implemented system. CogPrime is by no means the only possible path to human-level AGI, but we believe it is considerably more fully thought-through and fleshed-out than any available alternatives. Actually, we would love to see CogPrime and a dozen alternatives simultaneously pursued – this may seem ambitious, but it would cost a fraction of the money currently spent on other sorts of science or engineering, let alone the money spent on warfare or decorative luxury items. We strongly suspect that, in hindsight, our human and digital descendants will feel amazed that their predecessors allocated so few financial and attentional resources to the creation of powerful AGI, and consequently took *so long* to achieve such a fundamentally straightforward thing.

References

[AAB+11] Sam Adams, Itamar Arel, Joscha Bach, Robert Coop, Rod Furlan, Ben Goertzel, J. Storrs Hall, Alexei Samsonovich, Matthias Scheutz, Matthew Schlesinger, Stuart C Shapiro, and John Sowa. Mapping the landscape of human-level artificial general intelligence. *Artificial intelligence Magazine*, 2011.

[AABL02] Nancy Alvarado, Sam S. Adams, Steve Burbeck, and Craig Latta. Beyond the turing test: Performance metrics for evaluating a computer simulation of the human mind. *Development and Learning, International Conf. on*, 0, 2002.

[And96] John R. Anderson. *The Architecture of Cognition*. Lawrence Erlbaum Associates, 1996.

[ARC09] I. Arel, D. Rose, and R. Coop. Destin: A scalable deep learning architecture with application to high-dimensional robust pattern recognition. *Proc. AAAI Workshop on Biologically Inspired Cognitive Architectures*, 2009.

[Bac09] Joscha Bach. *Principles of Synthetic Intelligence*. Oxford University Press, 2009.

[Bat79] Gregory Bateson. *Mind and Nature: A Necessary Unity*. New York: Ballantine, 1979.

[BF71] J. D. Bransford and J. Franks. The abstraction of linguistic ideas. *Cognitive Psychology*, 2:331–350, 1971.

[BF09] Bernard Baars and Stan Franklin. Consciousness is computational: The lida model of global workspace theory. *International Journal of Machine Consciousness.*, 2009.

[Bol98] B. Bollobas. *Modern Graph Theory*. Springer, 1998.

[Cal96] William Calvin. *The Cerebral Code*. MIT Press, 1996.

[Cha09] Mark Changizi. *The Vision Revolution*. BenBella Books, 2009.

[Den91] Daniel Dennett. *Brainstorms*. Cambridge MA: MIT Press, 1991.

[FT02] G. Fauconnier and M. Turner. *The Way We Think: Conceptual Blending and the Mind's Hidden Complexities*. Basic, 2002.

[GASP08] Ben Goertzel, Onar Aam, Tony Smith, and Kent Palmer. Mirror neurons, mirrorhouses, and the algebraic structure of the self. *Cybernetics and Human Knowing,*, 2008.

[GB09] Ben Goertzel and Stephan Vladimir Bugaj. Agi preschool. In *Proc. of the Second Conf. on Artificial General Intelligence*. Atlantis Press, 2009.

[GdG08] Ben Goertzel and Hugo de Garis. Xia-man: An extensible, integrative architecture for intelligent humanoid robotics. pages 86–90, 2008.

[GEA08] Ben Goertzel and Cassio Pennachin Et Al. An integrative methodology for teaching embodied non-linguistic agents, applied to virtual animals in second life. In *Proc.of the First Conf. on AGI*. IOS Press, 2008.

[GI11] B. Goertzel and M. Iklé. Steps toward a geometry of mind. In J Schmidhuber and K Thorisson, editors, *Proceedings of AGI-11*. Springer, 2011.

[GIGH08] B. Goertzel, M. Ikle, I. Goertzel, and A. Heljakka. *Probabilistic Logic Networks*. Springer, 2008.

[GIW12] Ben Goertzel, Matt Ikle', and Jared Wigmore. The architecture of human-like general intelligence. In *Foundations of Artificial General Intelligence*, 2012.

[GMIH08] B. Goertzel, I. Goertzel M. Ikle, and A. Heljakka. *Probabilistic Logic Networks*. Springer, 2008.

[Goe94] Ben Goertzel. *Chaotic Logic*. Plenum, 1994.

[Goe01] Ben Goertzel. *Creating Internet Intelligence*. Plenum Press, 2001.

[Goe06] Ben Goertzel. *The Hidden Pattern*. Brown Walker, 2006.

[Goe08] Ben Goertzel. A pragmatic path toward endowing virtually-embodied ais with human-level linguistic capability. IEEE World Congress on Computational Intelligence (WCCI), 2008.

[Goe09a] Ben Goertzel. Cognitive synergy: A universal principle of feasible general intelligence? 2009.

[Goe09b] Ben Goertzel. The embodied communication prior. In *Proceedings of ICCI-09, Hong Kong*, 2009.

[Goe09c] Ben Goertzel. Opencog prime: A cognitive synergy based architecture for embodied artificial general intelligence. In *ICCI 2009, Hong Kong*, 2009.

[Goe10a] Ben Goertzel. Opencogprime wikibook. *OpenCog Wiki Site*, 2010. `http://wiki.opencog.org/w/OpenCogPrime:WikiBook`.

[Goe10b] Ben Goertzel. Toward a formal definition of real-world general intelligence. In *Proceedings of AGI-10*, 2010.

[Goe11] Ben Goertzel. The mind-world correspondence principle. *Dynamical Psychology*, 2011. `http://wp.dynapsyc.org/`.

[Goe12] Ben Goertzel. Perception processing for general intelligence: Bridging the symbolic/subsymbolic gap. *Proceedings of AGI-12*, 2012.

[GP08] Ben Goertzel and Cassio Pennachin. How might probabilistic reasoning emerge from the brain? In *Proceedings of AGI 2008*, 2008.

[GPC+11] Ben Goertzel, Joel Pitt, Zhenhua Cai, Jared Wigmore, Deheng Huang, Nil Geisweiller, Ruiting Lian, and Gino Yu. Integrative general intelligence for controlling game ai in a minecraft-like environment. In *Proc. of BICA 2011*, 2011.

[GPGtOT13] B. Goertzel, Cassio Pennachin, Nil Geisweiller, and the OpenCog Team. *Building Better Minds: An Architecture for Artificial General Intelligence*. In preparation, 2013.

[GPI+10] Ben Goertzel, Joel Pitt, Matthew Ikle, Cassio Pennachin, and Rui Liu. Glocal memory: a design principle for artificial brains and minds. *Neurocomputing*, April 2010.

[GPPG06] Ben Goertzel, Hugo Pinto, Cassio Pennachin, and Izabela Freire Goertzel. Using dependency parsing and probabilistic inference to extract relationships between genes, proteins and malignancies implicit among multiple biomedical research abstracts. In *Proc. of Bio-NLP 2006*, 2006.

[GW11] Ben Goertzel and Jared Wigmore. Cognitive synergy is tricky. *Chinese Journal of. Mind and Computation*, 2011.

[HG08] David Hart and Ben Goertzel. Opencog: A software framework for integrative artificial general intelligence. In *AGI*, volume 171 of *Frontiers in Artificial Intelligence and Applications*, pages 468–472. IOS Press, 2008.

[Hof79] Douglas Hofstadter. *Godel, Escher, Bach: An Eternal Golden Braid*. Basic, 1979.

[Hof96] Douglas Hofstadter. *Fluid Concepts and Creative Analogies*. Basic Books, 1996.

[Hut96] Edwin Hutchins. *Cognition in the Wild*. MIT Press, 1996.

[Hut05] Marcus Hutter. *Universal Artificial Intelligence: Sequential Decisions based on Algorithmic Probability*. Springer, 2005.

[Kur06] Ray Kurzweil. *The Singularity is Near*. 2006.

[KWRK05] R W Kamphaus, A P Winsor, E W Rowe, and S Kim. A history of intelligence test interpretation. In *Contemporary intellectual assessment: Theories, tests, and issues, Ed. by D.P. Flanagan and P.L. Harrison*, page 2338, 2005.

[Lai12] John E Laird. *The Soar Cognitive Architecture*. MIT Press, 2012.

[LH07] Shane Legg and Marcus Hutter. A collection of definitions of intelligence. In *Advances in Artificial General Intelligence*. IOS, 2007.

[Loo06] Moshe Looks. *Competent Program Evolution.* PhD Thesis, Computer Science Department, Washington University, 2006.

[Pei34] C. Peirce. *Collected papers: Volume V. Pragmatism and pragmaticism.* Harvard University Press. Cambridge MA., 1934.

[RM95] H. L. Roediger and K. B. McDermott. Creating false memories: Remembering words not presented in lists. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 21:803–814, 1995.

[Ros88] Israel Rosenfield. *The Invention of Memory: A New View of the Brain.* Basic Books, 1988.

[SZ04] R. Sun and X. Zhang. Top-down versus bottom-up learning in cognitive skill acquisition. *Cognitive Systems Research*, 5, 2004.

[TC05] Endel Tulving and R. Craik. *The Oxford Handbook of Memory.* Oxford U. Press, 2005.

[Tur50] Alan Turing. Computing machinery and intelligence. *Mind*, 59, 1950.

[Wan06] Pei Wang. *Rigid Flexibility: The Logic of Intelligence.* Springer, 2006.

[Who64] Benjamin Lee Whorf. *Language, Thought and Reality.* 1964.

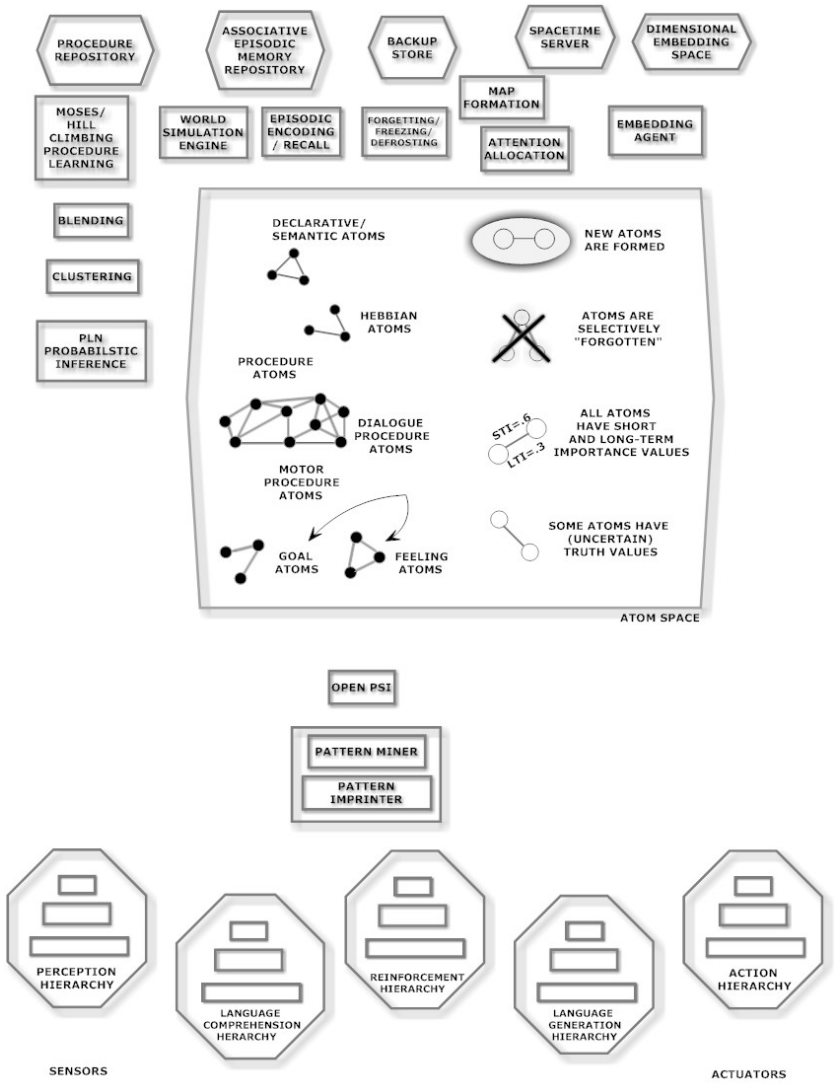

Figure 3: **Key Explicitly Implemented Processes of CogPrime**. The large box at the center is the Atomspace, the system's central store of various forms of (long-term and working) memory, which contains a weighted labeled hypergraph whose nodes and links are "Atoms" of various sorts. The hexagonal boxes at the bottom denote various hierarchies devoted to recognition and generation of patterns: perception, action and linguistic. Intervening between these recognition/generation hierarchies and the Atomspace, we have a pattern mining/imprinting component (that recognizes patterns in the hierarchies and passes them to the Atomspace; and imprints patterns from the Atomspace on the hierarchies); and also OpenPsi, a special dynamical framework for choosing actions based on motivations. Above the Atomspace we have a host of cognitive processes, which act on the Atomspace, some continually and some only as context dictates, carrying out various sorts of learning and reasoning (pertinent to various sorts of memory) that help the system fulfill its goal sand motivations.

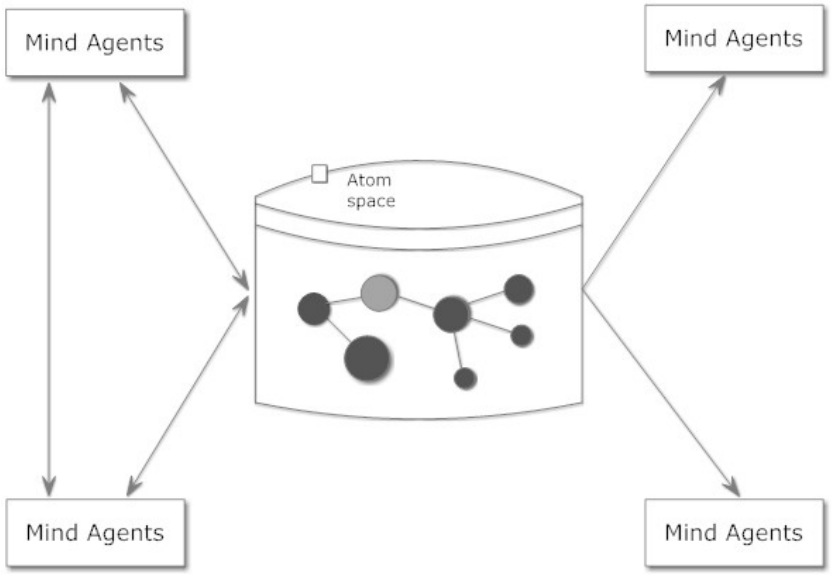

Figure 4: **MindAgents and AtomSpace in OpenCog.** This is a conceptual depiction of one way cognitive processes may interact in OpenCog – they may be wrapped in MindAgent objects, which interact via cooperatively acting on the AtomSpace.

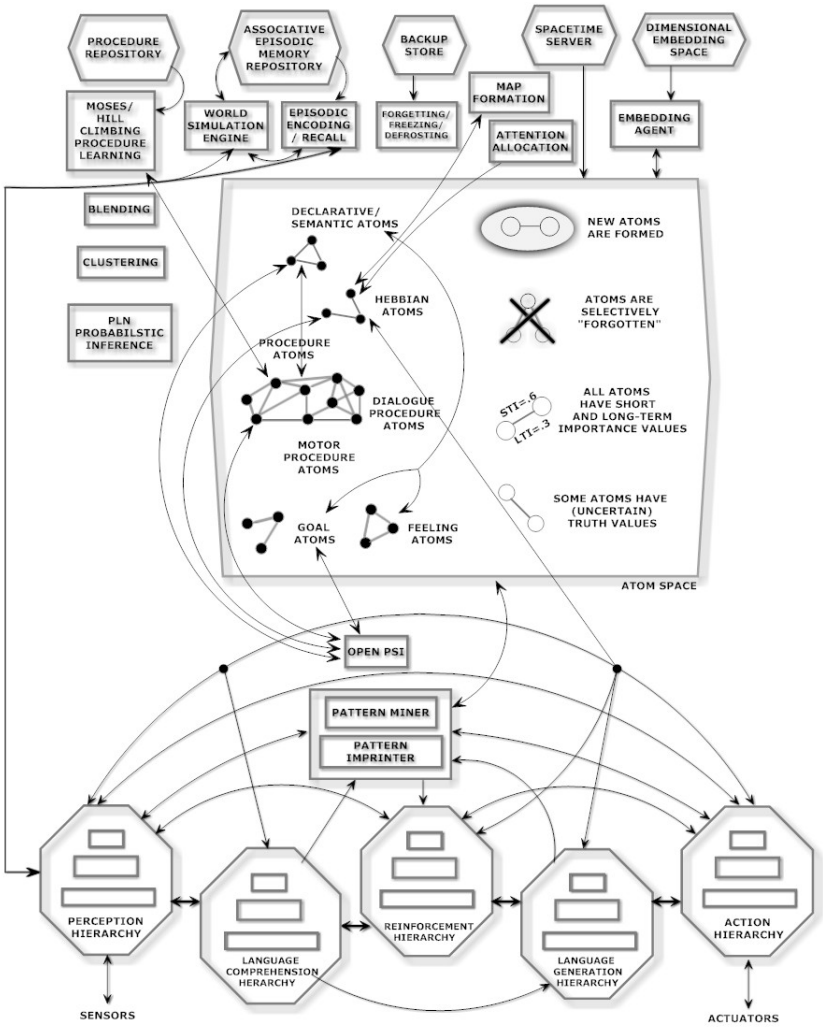

Figure 5: **Links Between Cognitive Processes and the Atomspace**. The cognitive processes depicted all act on the Atomspace, in the sense that they operate by observing certain Atoms in the Atomspace and then modifying (or in rare cases deleting) them, and potentially adding new Atoms as well. Atoms represent all forms of knowledge, but some forms of knowledge are additionally represented by external data stores connected to the Atomspace, such as the Procedure Repository; these are also shown as linked to the Atomspace.

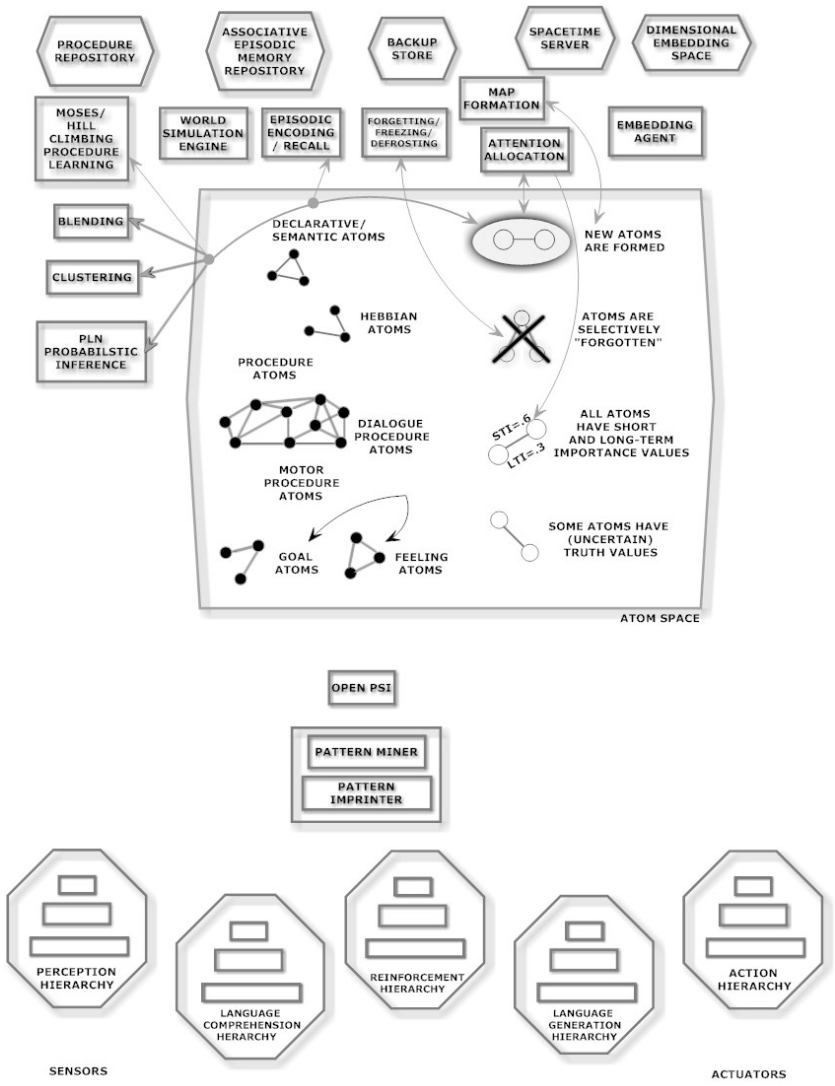

Figure 6: **Invocation of Atom Operations By Cognitive Processes**. This diagram depicts some of the Atom modification, creation and deletion operations carried out by the abstract cognitive processes in the CogPrime architecture.

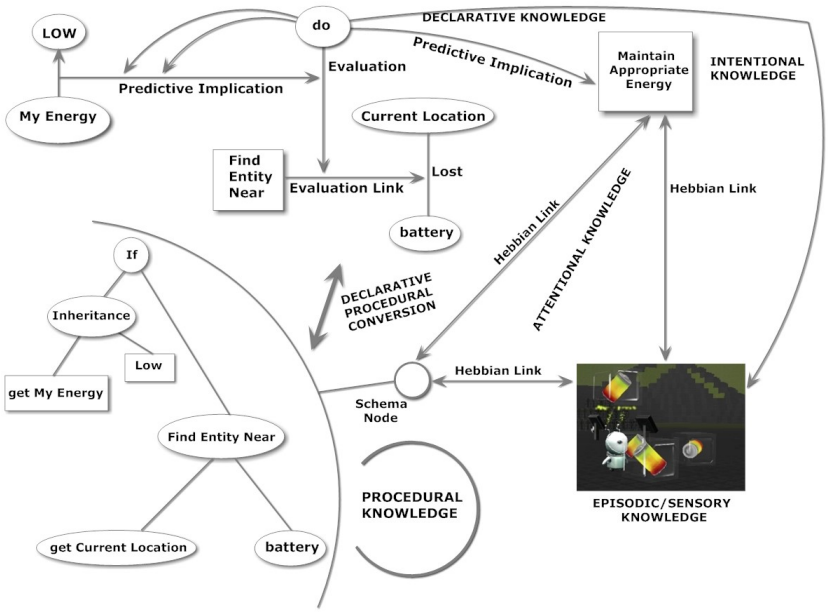

Figure 7: **Relationship Between Multiple Memory Types**. The bottom left corner shows a program tree, constituting procedural knowledge. The upper left shows declarative nodes and links in the Atomspace. The upper right corner shows a relevant system goal. The lower right corner contains an image symbolizing relevant episodic and sensory knowledge. All the various types of knowledge link to each other and can be approximatively converted to each other.

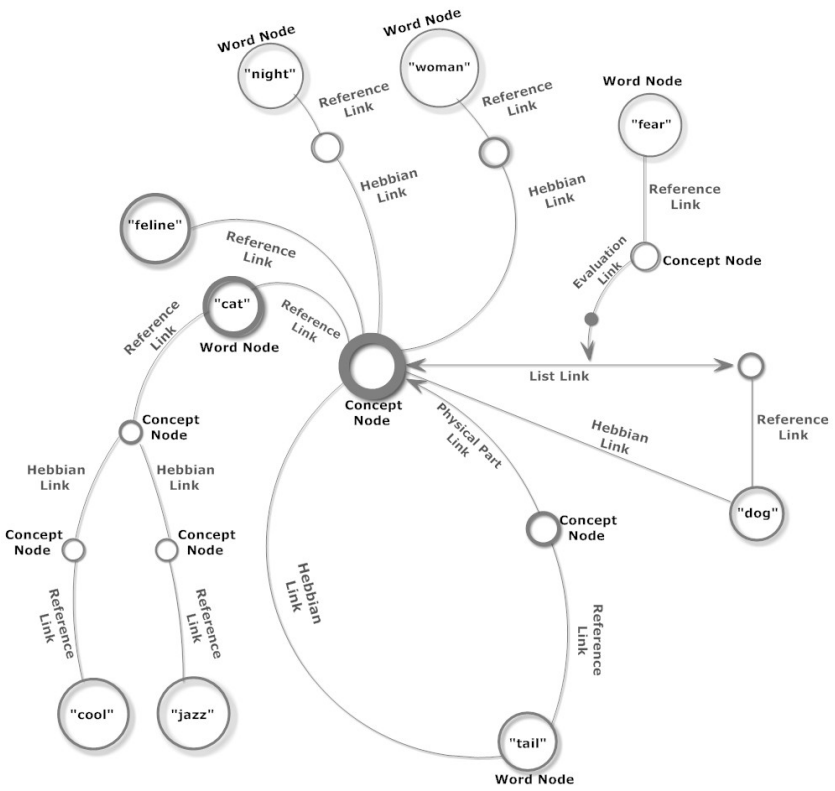

Figure 8: **Example of Explicit Knowledge in the Atomspace**. One simple example of explicitly represented knowledge in the Atomspace is linguistic knowledge, such as words and the concepts directly linked to them. Not all of a CogPrime system's concepts correlate to words, but some do.

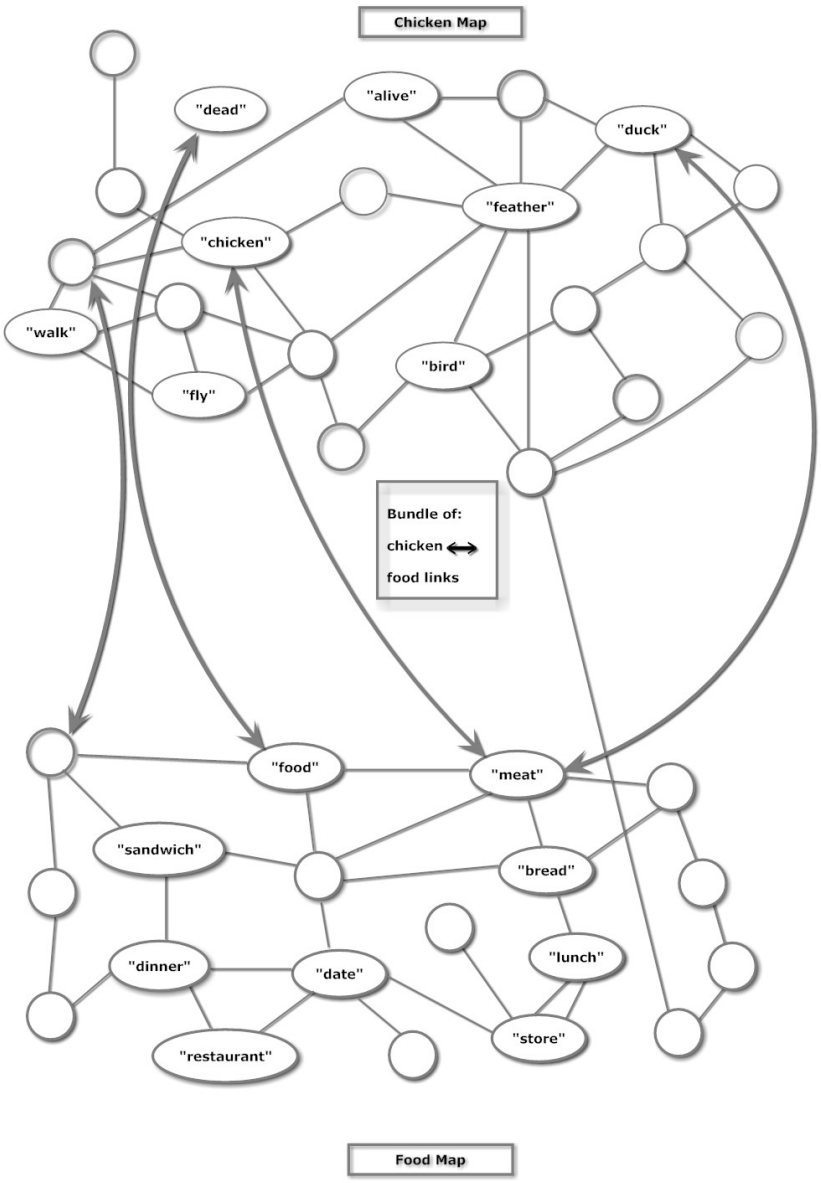

Figure 9: **Example of Implicit Knowledge in the Atomspace**. A simple example of implicit knowledge in the Atomspace. The "chicken" and "food" concepts are represented by "maps" of ConceptNodes interconnected by HebbianLinks, where the latter tend to form between ConceptNodes that are often simultaneously important. The bundle of links between nodes in the chicken map and nodes in the food map, represents an "implicit, emergent link" between the two concept maps. This diagram also illustrates "glocal" knowledge representation, in that the chicken and food concepts are each represented by individual nodes, but also by distributed maps. The "chicken" ConceptNode, when important, will tend to make the rest of the map important – and vice versa. Part of the overall chicken concept possessed by the system is expressed by the explicit links coming out of the chicken ConceptNode, and part is represented only by the distributed chicken map as a whole.