

Chapter 2

A Brief Overview of CogPrime

2.1 Introduction

Just as there are many different approaches to human flight—airplanes, helicopters, balloons, spacecraft, and doubtless many methods no person has thought of yet—similarly, there are likely many different approaches to advanced artificial general intelligence. All the different approaches to flight exploit the same core principles of aerodynamics in different ways; and similarly, the various different approaches to AGI will exploit the same core principles of general intelligence in different ways.

The thorough presentation of the CogPrime design is the job of Part 2 of this book—where, not only are the algorithms and structures involved in CogPrime reviewed in more detail, but their relationship to the theoretical ideas underlying CogPrime is pursued more deeply. The job of this chapter is a smaller one: to give a high-level overview of some key aspects the CogPrime architecture at a mostly nontechnical level, so as to enable you to approach Part 2 with a little more idea of what to expect. The remainder of Part 1, following this chapter, will present various theoretical notions enabling the particulars, intent and consequences of the CogPrime design to be more thoroughly understood.

2.2 High-Level Architecture of CogPrime

Figures 2.1, 2.2, 2.4 and 2.5 depict the high-level architecture of CogPrime, which involves the use of multiple cognitive processes associated with multiple types of memory to enable an intelligent agent to execute the procedures that it believes have the best probability of working toward its goals in its current context. In a robot preschool context, for example, the top-level goals will be simple things such as pleasing the teacher, learning new information and skills, and protecting the robot's body. Figure 2.3 shows part of the architecture via which cognitive processes interact with each other, via commonly acting on the AtomSpace knowledge repository.

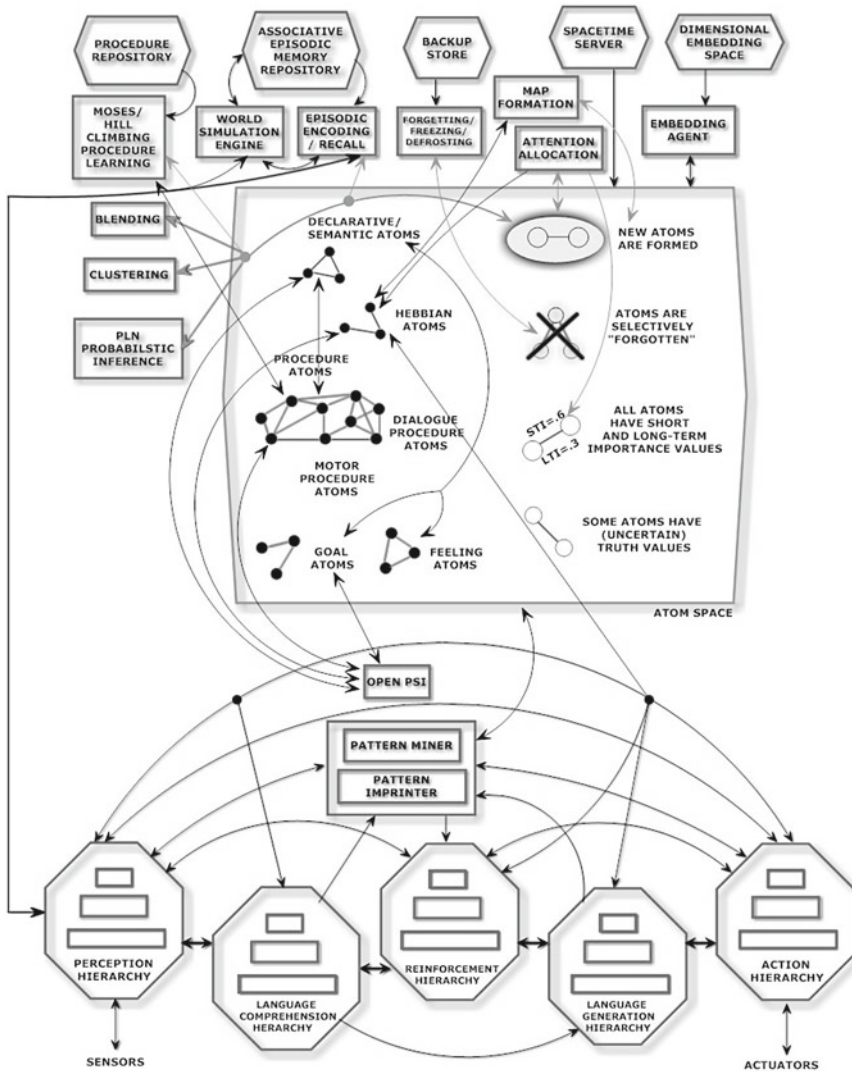


Fig. 2.1 High-level architecture of CogPrime. This is a conceptual depiction, not a detailed flow-chart (which would be too complex for a single image). Figures 2.2, 2.4 and 2.5 highlight specific aspects of this diagram

Comparing these diagrams to the integrative human cognitive architecture diagrams to be given in Chap. 7, one sees 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. Table 2.1 articulates the key connections between the components of the CogPrime diagram and those of the integrative diagram, thus indicating the general cognitive functions instantiated by each of the CogPrime components.

2.3 Current and Prior Applications of OpenCog

Before digging deeper into the theory, and elaborating some of the dynamics underlying the above diagrams, we pause to briefly discuss some of the practicalities of work done with the OpenCog system currently implementing 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. Most relevantly to the present work, it 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 Fig. 2.6 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
- spontaneous exploration of their world, remembering their experiences and using them to bias future learning and linguistic interaction.

One current OpenCog initiative 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 it’s 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

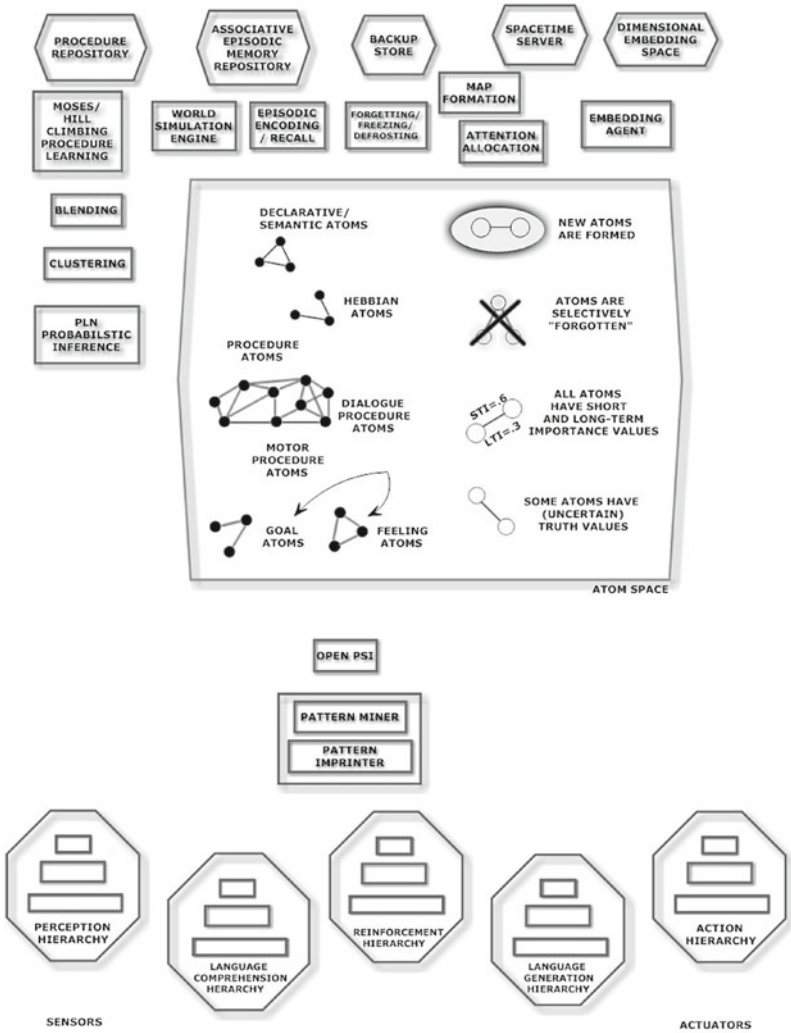


Fig. 2.2 Key explicitly implemented processes of CogPrime. The large box at the center is the Atomspace, the system’s central store of various forms of (long-term and working) memory, which contains a weighted labeled hypergraph whose nodes and links are “Atoms” of various sorts. The hexagonal boxes 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 goals and motivations

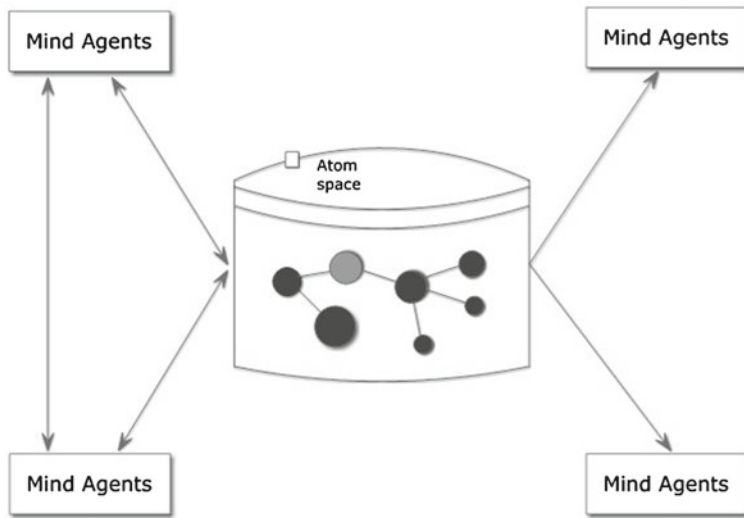


Fig. 2.3 MindAgents and AtomSpace in OpenCog. This is a conceptual depiction of one way cognitive processes may interact in OpenCog—they may be wrapped in MindAgent objects, which interact via cooperatively acting on the AtomSpace

relatively simple to make an AI system do things like this in a trivial and 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.

Specific, particularly simple tasks that are the focus of this project team's 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.

2.3.1 Transitioning from Virtual Agents to a Physical Robot

Preliminary experiments have also been conducted using OpenCog to control a Nao robot as well as a virtual dog [GdG08]. This involves hybridizing OpenCog with a

Table 2.1 Connections between the CogPrime architecture diagram and the integrative architecture diagram

CogPrime component	Int. diag. sub-diagram	Int. diag. component
Procedure repository	Long-term memory	Procedural
Procedure repository	Working memory	Active procedural
Associative episodic memory	Long-term memory	Episodic
Associative episodic memory	Working memory	Transient episodic
Backup store	Long-term memory	No correlate: a function not necessarily possessed by the human mind
Spacetime server	Long-term memory	Declarative and sensorimotor
Dimensional embedding space	No clear correlate: a tool for helping multiple types of LTM	
Dimensional embedding agent	No clear correlate	
Blending	Long-term and working memory	Concept formation
Clustering	Long-term and working memory	Concept formation
PLN probabilistic inference	Long-term and working memory	Reasoning and plan learning/optimization
MOSES/Hillclimbing	Long-term and working memory	Procedure learning
World simulation	Long-term and working memory	Simulation
Episodic encoding/recall	Long-term g memory	Story-telling
Episodic encoding/recall	Working memory	Consolidation
Forgetting/freezing/defrosting	Long-term and working memory	No correlate: a function not necessarily possessed by the human mind
Map formation	Long-term memory	Concept formation and pattern mining
Attention allocation	Long-term and working memory	Hebbian/attentional learning
Attention allocation	High-level mind architecture	Reinforcement
Attention allocation	Working memory	Perceptual associative memory and local association
AtomSpace	High-level mind architecture	No clear correlate: a general tool for representing memory including long-term and working, plus some of perception and action
AtomSpace	Working memory	Global workspace (the high-STI portion of AtomSpace) and other workspaces
Declarative atoms	Long-term and working memory	Declarative and sensorimotor

(continued)

Table 2.1 (continued)

CogPrime component	Int. diag. sub-diagram	Int. diag. component
Procedure atoms	Long-term and working memory	Procedural
Hebbian atoms	Long-term and working memory	Attentional
Goal atoms	Long-term and working memory	Intentional
Feeling atoms	Long-term and working memory	Spanning declarative, intentional and sensorimotor
OpenPsi	High-level mind architecture	Motivation/action selection
OpenPsi	Working memory	Action selection
Pattern miner	High-level mind architecture	Arrows between perception and working and long-term memory
Pattern miner	Working memory	Arrows between sensory memory and perceptual associative and transient episodic memory
Pattern imprinter	Working memory	Arrows between action selection and sensorimotor memory, and between the latter and perception/action subsystems
Pattern imprinter	High-level mind architecture	Arrows pointing to action subsystem from working and long-term memories
Perception hierarchy	High-level mind architecture	Perception subsystems
Perception hierarchy	Working memory	Perception/action subsystems and sensory and sensorimotor memory
Language comprehension hierarchy	Language	Comprehension hierarchy
Language generation hierarchy	Language	Generation hierarchy
Reinforcement hierarchy	High-level mind architecture	Reinforcement
Reinforcement hierarchy	Action	Reinforcement hierarchy
Action hierarchy	Action	Collection of specialized action hierarchies

There is a row for each component in the CogPrime architecture diagram, which tells the corresponding sub-diagrams and components of the integrative architecture diagram given in Chap. 7. Note that the description “Long Term and Working Memory” indicates occurrence in two separate sub diagrams of the integrative diagram, “Long Term Memory” and “Working Memory”

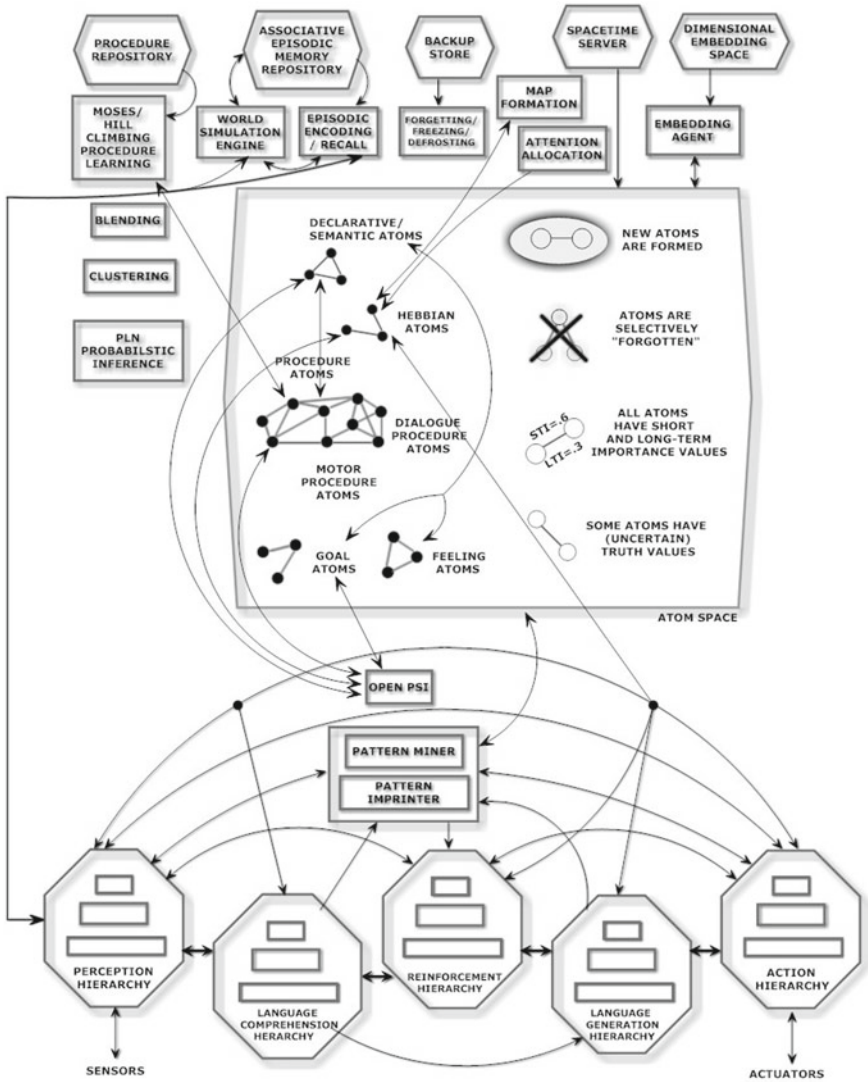


Fig. 2.4 Links between cognitive processes and the Atomspace. The cognitive processes depicted all act on the Atomspace, in the sense that they operate by observing certain Atoms in the Atomspace and then modifying (or in rare cases deleting) them, and potentially adding new Atoms as well. Atoms represent 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

separate (but interlinked) subsystem handling low-level perception and action. In the experiments done so far, this has been accomplished in an extremely simplistic way. How to do this right is a topic treated in detail in Chap. 8 of Part 2.

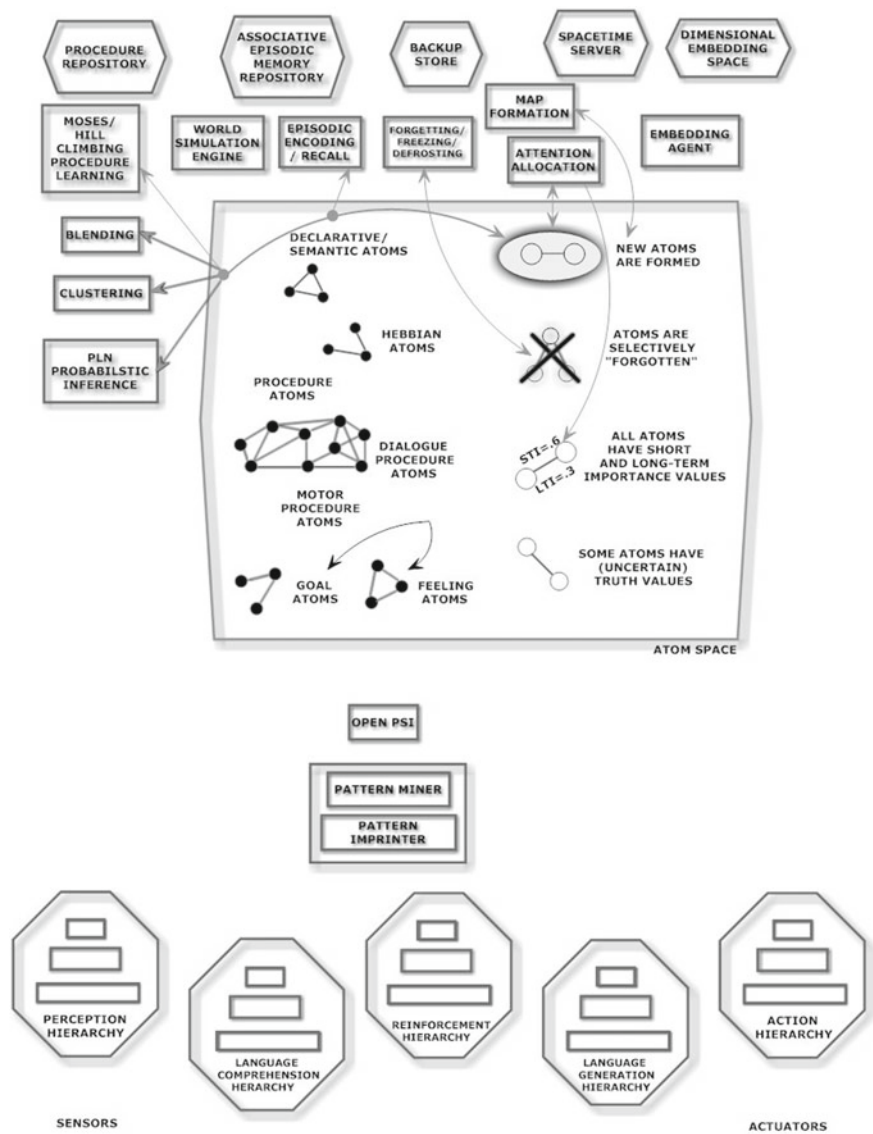


Fig. 2.5 Invocation of Atom operations by cognitive processes. This diagram depicts some of the Atom modification, creation and deletion operations carried out by the abstract cognitive processes in the CogPrime architecture

We suspect that reasonable level of capability will be achievable by simply interposing DeSTIN, as described in Chaps. 8–10 of Part 2 (or some other similar system), as a perception/action “black box” between OpenCog and a robot. Some preliminary experiments in this direction have already been carried out, connecting the



Fig. 2.6 Screenshot of OpenCog-controlled virtual dog

OpenPetBrain to a Nao robot using simpler, less capable software than DeSTIN in the intermediary role (off-the-shelf speech-to-text, text-to-speech and visual object recognition software).

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 will achieve this via the creation and real-time utilization of links between the nodes in CogPrime’s and DeSTIN’s internal networks (a topic to be explored in more depth later in this chapter).

2.4 Memory Types and Associated Cognitive Processes in CogPrime

Now we return to the basic description 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 2.2 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 2.7 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 patterns, 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 Chap. 8, efficiency is not a side-issue but rather the essence of real-world AGI (since as Hutter (2005) 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**.

Table 2.2 Memory types and cognitive processes in CogPrime

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	DeSTIN, and corresponding structures in the atomspace	Association, attention allocation, pattern creation, credit assignment

The third column indicates the general cognitive function that each specific cognitive process carries out, according to the patternist theory of cognition

2.4.1 Cognitive Synergy in PLN

To put a little meat on the bones of the “cognitive synergy” idea, discussed briefly above and more extensively in latter chapters, we now elaborate a little on the role it plays in the interaction between procedural and declarative learning.

While the probabilistic evolutionary learning algorithm 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], and are summarized in Chap. 16 (Part 2); here we will eschew most details and focus mainly on pointing out how PLN seeks to achieve efficient inference control via integration with other cognitive processes.

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

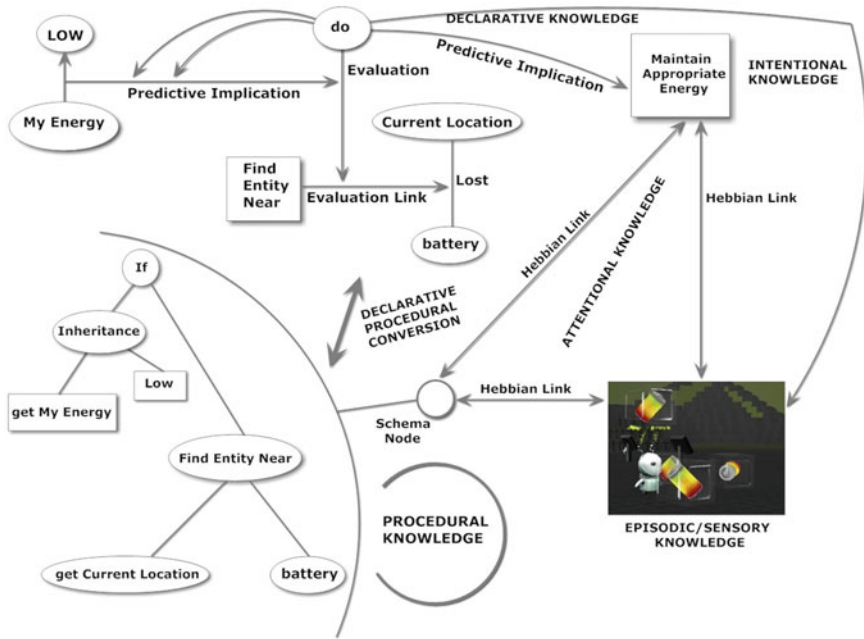


Fig. 2.7 Relationship between multiple memory types. The bottom left corner shows a program tree, constituting procedural knowledge. The upper left shows declarative nodes and links in the Atomspace. The upper right corner shows a relevant system goal. The lower right corner contains an image symbolizing relevant episodic and sensory knowledge. All the various types of knowledge link to each other and can be approximatively converted to each other

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 $(A_1, \dots, A_n)$ of (*inference rule, stored knowledge*) pairs that might be used to produce L
2. Using analogical reasoning to prior inferences, assign each A_i a probability of success

- If some of the A_i are estimated to have reasonable probability of success at generating reasonably confident estimates of L 's truth value, then invoke Step 1 with A_i in place of L (at this point the inference process becomes recursive)
- If none of the A_i looks sufficiently likely to succeed, then inference has “gotten stuck” and another cognitive process should be invoked, e.g.
 - **Concept creation** may be used to infer new concepts related to A and B , and then Step 1 may be revisited, in the hope of finding a new, more promising A_i involving one of the new concepts
 - **MOSES** may be invoked with one of several special goals, e.g. the goal of finding a procedure P so that $P(X)$ predicts whether $X \rightarrow B$. If MOSES finds such a procedure P then this can be converted to declarative knowledge understandable by PLN and Step 1 may be revisited....
 - **Simulations** may be run in CogPrime's internal simulation engine, so as to observe the truth value of $A \rightarrow B$ in the simulations; and then Step 1 may be revisited....

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

2.5 Goal-Oriented Dynamics in CogPrime

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

$$\text{Context} \wedge \text{Procedure} \rightarrow \text{Goal} < p >$$

(summarized $C \wedge P \rightarrow G$). 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 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 (as reviewed in Chap. 6; 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 have already been mentioned in Chap. 6.

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 will be reviewed in Chap. 6 below, 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 Fig. 2.8 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 Fig. 2.9 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

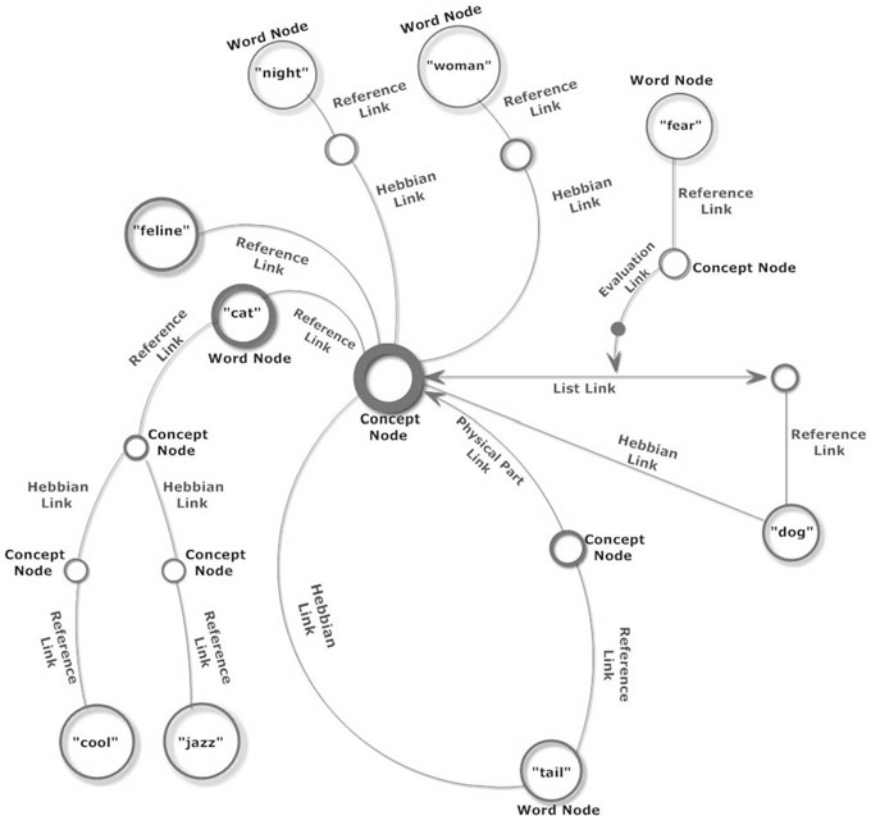


Fig. 2.8 Example of explicit knowledge in the Atomspace. One simple example of explicitly represented knowledge in the Atomspace is linguistic knowledge, such as words and the concepts directly linked to them. Not all of a CogPrime system's concepts correlate to words, but some do

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.

2.6 Analysis and Synthesis Processes in CogPrime

We now return to CogPrime's fundamental cognitive dynamics, using examples from the "virtual dog" application to motivate the discussion.

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:

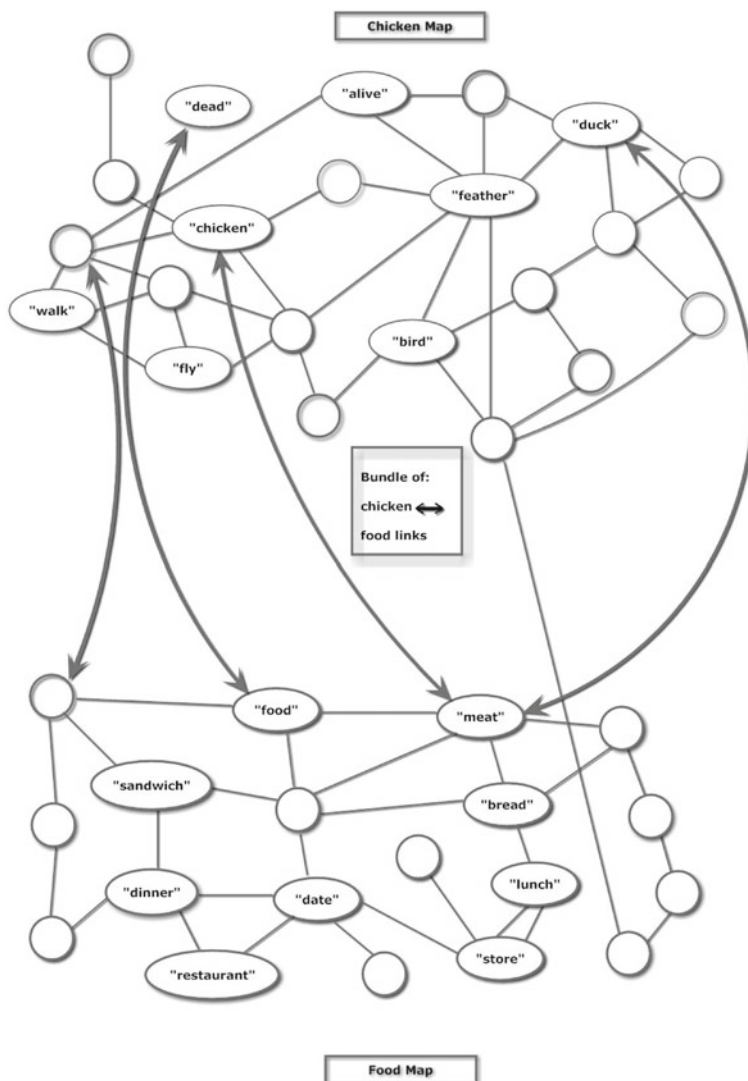


Fig. 2.9 Example of implicit knowledge in the Atomspace. A simple example of implicit knowledge in the Atomspace. The “chicken” and “food” concepts are represented by “maps” of ConceptNodes interconnected by HebbianLinks, where the latter tend to form between ConceptNodes that are often simultaneously 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

- **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.

More specifically, 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.

2.7 Conclusion

To thoroughly describe a comprehensive, integrative AGI architecture in a brief chapter would be an impossible task; all we have attempted here is a brief overview, to be elaborated on in Part 2 of this book. We do not expect this brief summary to be enough to convince the skeptical reader that the approach described here has a reasonable odds of success at achieving its stated goals. However, we hope to have given the reader at least a rough idea of *what sort of AGI design* we are advocating, and *why and in what sense we believe it can lead to advanced artificial general intelligence*. For more details on the structure, dynamics and underlying concepts of CogPrime, the reader is encouraged to proceed to Part 2—after completing Part 1, of course. Please be patient—building a thinking machine is a big topic, and we have a lot to say about it!

Engineering General Intelligence, Part 1

A Path to Advanced AGI via Embodied Learning and
Cognitive Synergy

Goertzel, B.; Pennachin, C.; Geisweiller, N.

2014, XXII, 409 p. 105 illus., 32 illus. in color., Hardcover

ISBN: 978-94-6239-026-3

A product of Atlantis Press