A non-routine problem solving mechanism for a general cognitive agent architecture

Aregahgen Negatu, Stan Franklin, and Lee McCauley

Department of Computer Science and the Institute for Intelligent Systems
The University of Memphis
Memphis, TN 38152, USA
{asnegatu|franklin}@memphis.edu,tmccauly@cs.memphis.edu

Abstract

One aspect of human intelligence is its ability to achieve goals by devising unexpected and even creative solutions to problems that have never before been encountered. This ability of exploring and constructing solutions to non-routine problems is central to the development of our sciences and technologies. Replicating Non-Routine Problem Solving (NRPS) capability in agent architecture would allow software agents and/or robots to deal more intelligently with highly complex and dynamically changing environments, and to cope with situations unforeseen by their designers. This chapter describes an NRPS mechanism in the cognitive architecture framework called Intelligent Distribution Agent (IDA) and its Learning incarnation LIDA. LIDA is a hybrid architecture that integrates different mechanisms for modules and their processes such as perception, emotion, memories (sensory, perceptual, episodic, procedural, working), action selection, expectation, learning, deliberation, problem solving, metacognition, and selective attention (or functional consciousness). Relevant to the NRPS mechanism, we will briefly discuss LIDA's perception, selective attention, expectation, procedural memory, and action selection mechanisms.

Our general approach is that an NRPS mechanism should involve the process of recruiting and activating all the available knowledge pieces and processes so that a search for a novel solution takes place over the entire solution space of the agent. L/IDA's procedural memory stores, besides the lower-level entities that we call behaviors, high-level procedural constructs called behavior streams or goal hierarchies (hierarchical partially-ordered action plans.) If a behavior stream becomes relevant based on the content of selective attention, a copy of it then gets instantiated with its variables bound, and becomes part of the dynamics in the action selection module. While active in the action selection system it competes for the control of the behavior of the agent over multiple cognitive cycles and executes an associated task. LIDA's cognitive cycle is an iterative, continually active process that brings about the interplay among the various components of the architecture, resulting in an action being selected and executed. Particularly, we view a non-routine problem solving mechanism as a special goal hierarchy whose task it is to generate a new behavior stream that handles a novel situation that could not be handled by available routine solutions. Non-routine problem solving in LIDA is a deliberative process over multiple cognitive cycles. We will describe the details of the interaction of the components in the architecture, and the control of the special non-routine problem solving goal hierarchy system over the solution search process.

1. Introduction

Advances in machine intelligent problem solving are possible by understanding and modeling cognitive functions and their integration in humans. An autonomous agent (Franklin & Graesser, 1997), whether it is human, animal or artificial, must continual decide on its next action and to do so, it must frequently sense its environment and select the appropriate response that satisfy or leads towards satisfying one or more of its primary drives. As agents negotiate environments, they encounter multitude of challenges or problems. Looking at differently, an autonomous agent engages in a continuous process of solving problems – while a problem situation is defined by its current environmental and internal states along with its current goals that could satisfy at least on one its agenda or drives; finding a solution (producing the appropriate action in the given situation) is a deliberative process that could involve multiple mental modules. The level of familiarity to the problem situation determines whether a problem is routine or non-routine and the amount of cognitive load in the deliberation. A routine problem solving is a process of applying an established procedure as a solution to a problem given that there is past experience of using this solution to the problem. A non-routine problem solving is a process of devising a new procedure that solves a problem in a way that has not been solved previously. The later is our main focus in this chapter.

We humans, as agents with the highest cognitive capacity, have the ability to devise unexpected, and often clever, solutions to problems we've never before encountered. Sometimes they are even creative. This ability to solve non-routine problems has played a central role in the development of our sciences and technologies. It would be useful to replicate this ability in software agents and robots, both for its practical value and for the light it would shed on human problem solving. On the practical side, agent architecture capable of non-routine problem solving would allow software agents and/or robots to deal more intelligently with highly complex and dynamically changing environments, and to cope with situations unforeseen by their designers. Applications might be to unmanned vehicles for exploratory or military uses, as well as industrial control systems. The tasks of many human information agents could be automated (Franklin, 2001). On the science side, each design decision taken in pursuit of such an agent architecture translates immediately into a hypothesis, hopefully testable, for cognitive scientists and neuroscientists (Franklin, 1997, 2000b; Franklin & Graesser, 2001; Baars & Franklin, 2003; Franklin, Baars, Ramamurthy, & Ventura, 2005).

Baars (1988, 1997) in his Global Workspace Theory (GWT) presented strong psychological and neuroscience evidences that show the important role of consciousness in integrating the many specialized cognitive processes so that humans can deal with novelty in general and non-routine problem solving in particular. Mithen (1996, 1999) also affirms, in presenting his notion of *cognitive fluidity* in the evolution of mind, that modern human's highest form of mental activity takes place as a result of cognitive fluidity (consciousness) – the working together or integration of the various mental modules and a fluid flow of knowledge and idea among them. Consciousness's role in fluid flow or broadcast of information enable the modules to influence each other (cooperation and competition) and this in turn results a boundless capacity for learning, imagination, creativity, and problem solving. Mithen explicitly mentions the need to evolve to conscious mind for a capacity to handle a non-routine problem solving such as tool making (e.g. handaxe), in which unexpected contingencies arise and plans need to be continually modified.

Many others also argue that human intelligence will not be where it is now without evolving consciousness to handle novelty (Baars, 1988; Arp, 2007; Bogdan, 1994; Cosmides & Tooby, 1992; Gardner, 1993; Humphrey, 1992; Pinker, 1997). Searle (1992) points out that our much greater flexibility, sensitivity, and creativity are derived from the evolutionary advantages of attaining consciousness. Crick (1994) also asserts that without consciousness, you can deal only with familiar, rather routine situations or respond to with very limited information in novel situations (p. 20).

Thus there is ample reason to pursue agent architecture with capability for non-routine problem solving. But can such pursuits succeed? We argue here that there is a good chance that it can. The major source of our optimism is the software agent technology of Learning Intelligent Distribution Agent (LIDA) - an autonomous agent that aspires to model several facets of human (and animal) cognition. LIDA is the partially conceptual, learning extension, of the original IDA system implemented computationally as a software agent (D'Mello et al., 2006). The original IDA system was designed as an autonomous agent, and performed personnel work for the US Navy in a human-like fashion (Franklin, 2001). Although the design of IDA was inspired by several theories of human and animal cognition, it did not learn. The LIDA system adds three fundamental forms of learning to IDA: perceptual, procedural, and episodic learning.

Over the years several AI researchers and cognitive scientists have developed cognitive models designed around some unified theory of cognition (Newell, 1990). Some of these well known models include SOAR (Laird, Newell, & Rosenbloom, 1987), ACT-R (Lebiere & Anderson, 1993), and Clarion (Sun, 1997). Most of these are based on some extension of Post production systems (Post, 1943). In contrast, The LIDA model is a comprehensive, conceptual and computational model covering a large portion of human cognition. The model is primarily based on Baars' Global Workspace theory (GWT) (1988) and it implements and fleshes out a number of mostly psychological and neuropsychological cognition theories. These include: situated or embodied cognition (Varela, Thompson, & Roach 1991, Glenberg, 1997), Barsalou's theory of perceptual symbol systems (1999), working memory (Baddeley & Hitch, 1974), Glenberg's theory (1997) of the importance of affordances to understanding, and Sloman's architecture for a human-like agent (1999). There are architecture that aspire to model consciousness and LIDA's comparison with some of them is made somewhere else (Franklin et. al., 2007).

To implement LIDA's cognitive model, its computational architecture employs several modules that are designed using computational mechanisms drawn from the "new AI." These include variants of the Copycat Architecture (Hofstadter and Mitchell 1995, Marshall 2002), Sparse Distributed Memory (Kanerva, 1988; Rao and Olac, 1998), the Schema Mechanism (Drescher 1991, Chaput et al. 2003), the Behavior Net (Maes 1989), and the Subsumption Architecture (Brooks 1991).

We argue that basing non-routine problem solving and other components of intelligence behavior on functional aspects of human cognition may serve dual purposes by yielding both engineering and scientific gains. We expect engineering improvements because we are basing our computational mechanism on the best known example of intelligence, i.e. humans. Scientific gains can be achieved by using computer systems to test and perhaps augment psychological theories of non-routine problem solving (NRPS) and other mental decision making processes.

One potential pitfall of relying on psychological theories is that they typically model only small pieces of cognition. In contrast, by its very nature the control system of any autonomous agent or cognitive robot must be fully integrated. That is, it must chose its actions based on real world sensation and perception along with incoming endogenous stimuli utilizing *all* needed internal processes. Once again the use of the LIDA system, as integrative general intelligence architecture, helps to overcome this problem, a major contribution in explaining a plausible cognitive approach for non-routine problems solving. As the focus of this chapter, we will discuss a non-routine problems solving mechanism in the LIDA agent framework, which incorporates much of what seems to be necessary for non-routine problem solving including functional consciousness.

In pursuing such a non-routine problem solving architecture, we'll need help from many sources. Our basic approach is on what Psychologist Arthur Glenberg refers to *meshing* - a process of finding unexpected and sometimes clever solutions to non-routine problems (as cited in; Glenberg & Robertson, 2000). Meshing is typically accomplished in humans by putting together bits and pieces of knowledge and techniques that have been stored and, perhaps, used in the past to help solve other problems. Finding a solution often begins with identifying these bits and pieces, then typically continuing to find ways to mesh them. We will use these observations as a guideline. Consciousness provides the means to extensively mobilize the unconscious bits and pieces of knowledge that could contribute towards constructing a solution.

This chapter will have the following outline. In the next section, we will discuss LIDA, our agent architecture, and its relevant components to support non-routine problem solving as well as its cognitive processing cycle. In section three, we will present our issues that need to be addressed in non-routine problem solving. Particularly, we discuss the approach for detection of non-routine problem situation. Then, in section four we discuss the main body of this chapter, a mechanism for non-routine problem solving in our cognitive agent architecture – LIDA. We conclude the chapter with a conclusion that has brief discussion on related works and summarization of this chapter.

2. Architectural Support for Non-Routine Problem Solving

The LIDA is hybrid architecture (partly symbolic and partly connectionist) with all symbols being grounded in the physical world in the sense of Brooks (1986). The fundamental computational mechanism of the LIDA system is the *codelet* (Hofstadter & Mitchell, 1994), a small piece of code executing as an independent thread that is specialized for some relatively simple task. There are many types of specialized codelets; for instance, some are specialized to identify particular perceptual feature (perceptual codelets), some others pay attention to a particular situation (attention codelets), others are specialized to execute low-level procedural actions (behavior codelets) and so on. LIDA architecture integrates different mechanisms for several cognitive modules and their processes including perception, emotion, memories (sensory, perceptual, episodic, procedural, working), action selection, expectation, learning, deliberation, problem solving, metacognition, and selective attention (or functional consciousness). Relevant to the NRPS mechanism, we will briefly discuss LIDA's perceptual associative memory, workspace, selective attention, expectation, procedural memory, and action selection mechanisms.

2.1 Perceptual Associative Memory

The perceptual knowledge-base takes the form of a semantic net with activation (called the slipnet) motivated by Hofstadter and Mitchell's Copycat architecture (1994). Nodes of the slipnet constitute the agent's perceptual symbols (Barsalou, 1999), representing individuals, categories and simple relations. The perceptual symbols are grounded in the real world by their ultimate connections to various primitive feature detectors having their receptive fields among the sensory receptors. An incoming stimulus, say a visual image, is descended upon by a hoard of perceptual codelets. Perceptual codelets respond to specific features from the various sensory streams and perform perceptual tasks such as recognition and identification. Each of these codelets is looking for some particular feature (a certain color, an edge at a particular angle, etc) or more complex features (a T junction, a red line). Upon finding a feature of interest to it, the codelet will activate an appropriate node or nodes in the slipnet. Activation is passed. The network will eventually stabilize. Nodes with activations over threshold, along with their links, are taken to provide the constructed meaning of the stimulus, the percept (see Figure 1).

2.2 Workspace

LIDA's workspace is analogous to the preconscious buffers of human working memory. Perceptual codelets write to the workspace as do other, more internal codelets. Attention codelets watch what is written in the workspace in order to react to it. Items in the workspace decay over time, and may be overwritten. Another pivotal role of the workspace is the building of temporary structures over multiple cognitive cycles (see below). Perceptual symbols from the slipnet are assimilated into existing relational and situational templates while preserving spatial and temporal relations between the symbols. The structures in the workspace also decay rapidly.

2.3 Selective Attention

Selective attention (functional consciousness) in LIDA is an implementation of Global Workspace Theory (Baars, 1988) with hosts of attention codelets, each playing the role of a daemon, watching for an appropriate condition under which to act. Each attention codelet watches for some particular situation that might call for selective attention (i.e. novelty, changes, etc). Upon encountering such a situation, the attention codelet is associated with a few nodes (from the slipnet) carrying a description of the situation. A coalition of codelets (collection of related codelets) is thus formed. During any given cycle one of these coalitions with the highest average activation is considered relevant and broadcasts its information to every other codelet (Baars, 1988). This broadcast is used to recruit schemes (see below) and perform various types of learning (D'Mello, et al., 2006).

2.4 Procedural Memory

Procedural memory in LIDA is a modified and simplified form of Drescher's schema mechanism (1991), the scheme net. The scheme net is a directed graph whose nodes are (action) schemes and whose links represent the 'derived from' relation. Built-in primitive (empty) schemes directly controlling effectors are analogous to motor cell assemblies controlling muscle groups in humans. A scheme consists of an action, together with its context and its result (see Figure 1). The context and results of the schemes are represented by perceptual symbols (Barsalou, 1999) for objects, categories, and relations in perceptual associative memory. In order for a scheme to act, it first needs to be instantiated and then selected for execution in accordance with the action selection mechanism (discussed next).The action of an instantiated scheme consists of one or more behavior codelets that execute the actions in parallel.

2.5 Action Selection

The LIDA architecture employs an enhancement of Maes' behavior net (1989) for high-level action selection in the service of drives (primary and internal motivators) (Cañamero, 1997). The behavior net is a digraph (directed graph) composed of behavior codelets (a single action), behaviors (multiple behavior codelets operating in parallel), and behavior streams (multiple behaviors operating in some partial order) and their various links. These three entities all share the same representation in procedural memory (i.e., a scheme). As in connectionist models, this digraph spreads activation. The activation comes from three sources: from pre-existing activation stored in the behaviors, from the environment, and from drives. To be acted upon, a behavior must be executable (preconditions satisfied), must have activation over threshold, and must have the highest such activation.

LIDA's action selection mechanism incorporates five major enhancements over Maes' behavior net: (i) *Variables* – While Maes' behavior net operates on the basis of boolean propositions only, LIDA's mechanism supports variables that get bound during the instantiation of procedural schemes; (ii) *Restricted search space* – During the action selection phase Maes' mechanism performs a global search over all the available competency modules while the enhanced behavior net restricts its search to relevant (instantiated) goal hierarchies, which are a subset of the available competencies; (iii) *Failure handling* - Maes' mechanism assumes that the result of a selected action is deterministic in that every action produces its expected outcome. Therefore, this mechanism is unable to handle execution failures which frequently occur in any real system. On the other hand LIDA's enhanced behavior net is endowed with a degree of fault tolerance via its expectation mechanism; (iv) *Priority control* – Maes' mechanism modulates the priorities of competing goals by building *static* causal links among competence modules while LIDA's mechanism provides parametric control to *dynamically* change goal priorities at run time; (v) *Planning and subgoaling* – Maes' mechanism does not support classic AI planning and subgoaling but LIDA's mechanism, as a collection of goal structures, supports both (see Negatu, 2010; Negatu & Franklin, 2002).

Expectation Codelet

In the LIDA architecture all high-level constructs such as behaviors are underlain by specialized processors – codelets. Particularly, one or behavior codelets (executing in parallel) and at least one expectation codelets underlie each behavior. One of the functions of behavior codelets is to realize a low level primitive action that is equivalent to a neuronal group that control primitive motor activity. Expectation codelets, as anticipatory behavior controllers, are important Role in the action selection system. Their controlling functions are monitoring behavioral effects, evaluating outcomes and reporting possible failures during the execution of a behavior.

Expectation codelets always attempt to provide feedback, which is based on the comparison of actual internal sensory consequences (including proprioception, tactile, etc.) and sensed environmental effects of behavioral action with the desired effects. The feedback compilation process may take one or more cognitive cycles (see below). When behavioral action fails, an expectation codelet tries to bring the detected failure to consciousness. The function of watching the events that are associated with the action of its behavior makes the expectation codelet a special case of an attention codelet. Expectation codelets have role in automatization (Negatu, 2006), non-routine problem solving, procedural learning (D'Mello, et al., 2006). They also have functional equivalence to preafference (Freeman, 1995) process in the brain and preparatory attention (LaBerge, 1995.)

Goal hierarchy and Subgoaling

The subgoaling and goal hierarchy facilitation in the behavior streams is particularly important to the non-routine problem solving mechanism that we address in this chapter. According to global workspace theory a goal context (or an intention) is a non-qualitative mental representation about one's own future actions that can dominate central limited capacity or consciousness; it constrains conscious goal images without itself being conscious. Goal contexts perform the following major functions. (1) They help to evoke, shape and produce actions. (2) They can be part of hierarchical structures that allow the agent to deal with tasks that extend over more than one conscious event. (3) They represent future states of the agent, serving to constrain the processes that can help reach those future states. (4) They can be part of hierarchical structures that can constrain a stream of "consciousness." In this section we discuss the high-level implementation of IDA's action selection module that comprises its goal context hierarchy.

Figure 1 illustrates an example of such a goal context hierarchy. Drives are at the top of the hierarchy, where behavior streams serve one or more of these drives directly or indirectly. Behavior stream 1 satisfies the drive directly, while stream 2 satisfies it indirectly through behavior stream 1. Such structures are not explicitly programmed; they are constructed at run time in an instantiated behavior network system. The figure shows a goal node – which is a special case of behavior node without explicit action. They simply state or infer when a particular state of affairs is desired and/or achieved.

The style of organization of smaller behavior streams into larger goal hierarchies (Figure 1) provides an effective sub-goaling mechanism for our action selection system. The sub-goaling mechanism brings a regressive reasoning capability in the decision making or action selection process. Small behavior streams, which are well defined partial order plans, are solutions to simple routine tasks. Such small behavior streams can be brought together to handle larger routine and non-routine problems. Referring to figure 1, a deliberative process with backward reasoning can recruit the appropriate behavior stream that satisfies and bind the unbound subgoal of behavior stream 1 at its behavior B3. The sub-goaling process is the one that plays the role of 'bringing together' (or the construction of goal hierarchy). That is, sub-goaling allows a goal-directed problem solving.

2.6 Cognitive Cycle

Autonomous agent (Franklin & Graesser, 1997) cops with its changing environment by its continuous, cyclic chore of "sense-decide-act." LIDA's cognitive cycle (Franklin et al., 2005) is the cycle of refined cognitive steps (starting after sensation and ending with action) that bring about the appropriate decision for specific situation. As Franklin and Baars (2009) put it "A cognitive cycle can be thought of as a moment of cognition - a cognitive moment; higher-level cognitive processes are composed of many of these cognitive cycles, each a cognitive atom." This metaphor is to say that the steps in a cognitive cycle correspond to the various sub-atoms in an atom. While there is evidence to suggest that in humans about five cognitive cycles happen per second. Since the LIDA architecture is composed of several specialized mechanisms a continual process that causes the functional interaction among the various components is essential. The cognitive cycle as such is an iterative, cyclical, continually active process that brings about the interplay among the various components of the architecture. The nine steps of cognitive cycle are shown in figure 2 and described below.

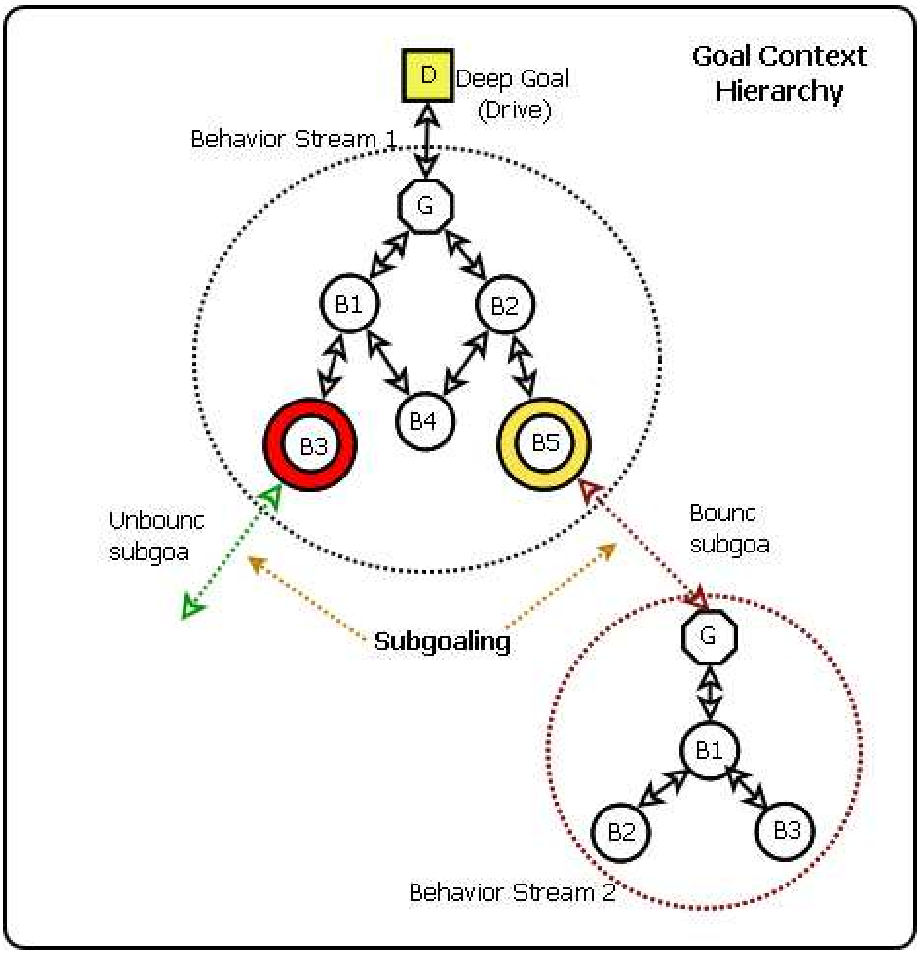

Figure 1. An example of a goal context hierarchy with multiple be3havior streams working together. Stream 1 handles its problem only after streams 2 satisfies its subgoal (at B5) and after another stream binds and satisfies the unbound subgoal (at B3).

1) **Perception**. Sensory stimuli, external or internal, are received and interpreted by perception producing the beginnings of meaning.

2) **Percept to preconscious buffer**. The percept, including some of the data plus the meaning, as well as possible relational structures, is stored in the preconscious buffers of LIDA's working memory (workspace). Temporary structures are built.

3) **Local associations**. Using the incoming percept and the residual contents of working memory, including emotional content, as cues, local associations are automatically retrieved from transient episodic memory and from declarative memory, and stored in long-term working memory.

4) **Competition for consciousness**. Attention codelets view long-term working memory, and bring novel, relevant, urgent, or insistent events to consciousness.

5) **Conscious broadcast**. A coalition of codelets, typically an attention codelet and its covey of related information codelets carrying content, gains access to the global workspace and has its contents broadcast. In humans, this broadcast is hypothesized to correspond to phenomenal consciousness.

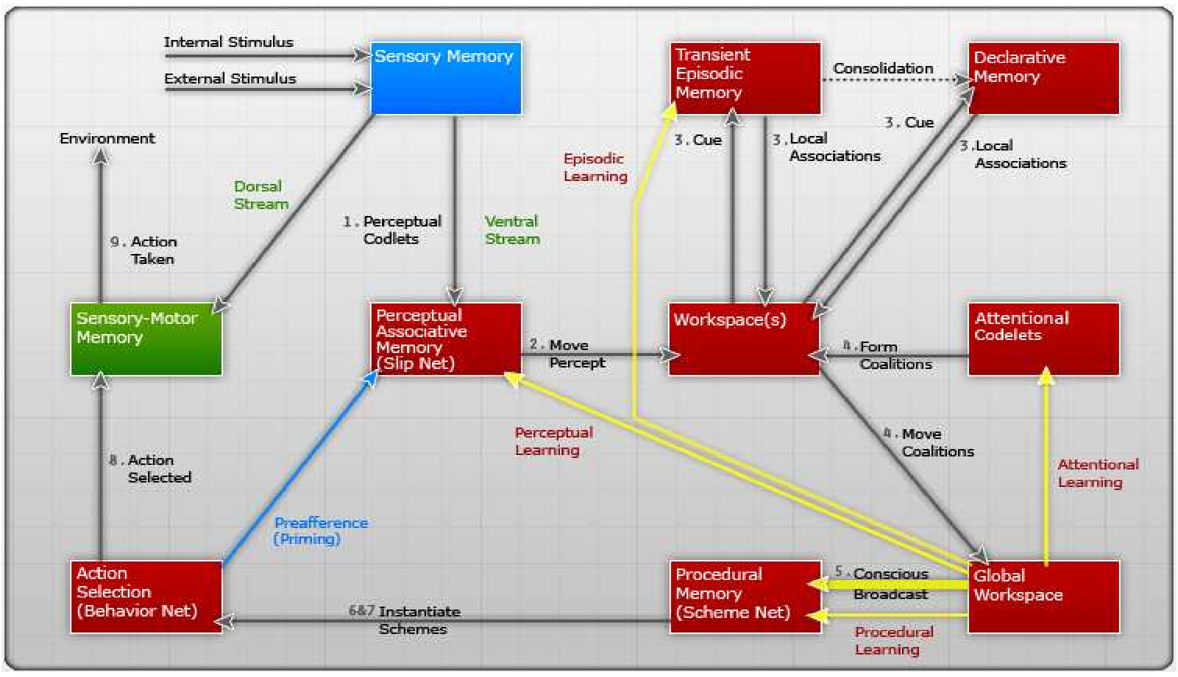

Figure 2. LIDA's Cognitive cycle

6) **Recruitment of resources**. Relevant schemes respond to the conscious broadcast. These are typically schemes (underlain by behavior codelets) whose context is relevant to information in the conscious broadcast. Thus consciousness solves the relevancy problem in recruiting resources.

7) **Setting goal context hierarchy**. The recruited schemes use the contents of consciousness, including feelings/emotions, to instantiate new goal context hierarchies (copies of themselves) into the behavior net (action selection system), bind their variables, and increase their activation. Other, environmental, conditions determine which of the earlier behaviors (goal contexts) also receive variable binding and/or additional activation.

8) **Action chosen**. The behavior net chooses a single behavior (scheme, goal context), from a just instantiated behavior stream or possibly from a previously active stream. Each selection of a behavior includes the generation of an expectation codelet (see the next step).

9) **Action taken**. The execution of a behavior (goal context) results in the behavior codelets performing their specialized tasks, having external or internal consequences, or both. LIDA is taking an action. The acting codelets also include at least one expectation codelet whose task it is to monitor the action, bringing to consciousness any failure in the expected results.

As shown in figure 2, multiple learning mechanisms are initiated following the broadcast of conscious content. In perceptual associative memory learning of new entities and associations, and the reinforcement of old ones occur, events are encoded in the Transient Episodic Memory, and new schemes may be learned old schemes are reinforced in Procedural Memory; in all of the learning processes, the conscious content determines what to be learned.

Several high-level tasks and cognitive processes operate over multiple cognitive cycles. Particularly, deliberative processes are in the continuous interplay and mutual influence of behaviors streams (goal context hierarchies) and selective attention (consciousness). Non-routine problem solving in LIDA is such a deliberative process spanning over multiple cognitive cycles. We will describe the details of the interaction of the components in the architecture, and the control of the special non-routine problem solving goal hierarchy system over the solution search process.

3. Approach for Non-Routine Problem Solving

In this section we will restate what the problem is and how we approach the modeling and design of a non-routine problem solving module in the LIDA architectural framework.

3.1 Routine and Non-Routine Problems

Routine problems are the ones that have readily available *applicable solutions*. Non-routine problems are novel problem situations that arise from *unavailability of routine solutions* that can handle them. More specifically, non-routine problems cannot be categorized as any of the routine classes of problems, and thus none of the available behavior streams could be applicable.

In general, a problem solving process may include the capabilities of noting differences, similarities and, to some extent, analogies between a non-routine problem and well-established routine problems, and the resulting information is used to construct new solutions by modifying existing solutions. A mechanism may need to breakdown complex structures of procedural knowledge to lower-level components, then using the available knowledge to construct new goal structures as solutions. The breakdown of structures can continue to the simplest or collection of base units that Edelman (1987) called primary repertoires. Problem solving, whether routine or non-routine and like traditional AI planning, is always specified with goal and the initial states

There are two main issues to be considered in handling a non-routine problem situation. The first involves the recognition of a novel situation and the relevant information contained in the pieces of percepts. While this is, indeed, an important and challenging issue, it is not the sense of nonroutine problem solving that we are addressing here. We consider this first issue one of perceptual learning, rather than nonroutine problem solving. The second issue is to determine how to respond to the situation in an appropriate manner. This dynamic adaptation of procedural knowledge will be addressed here as the sense of nonroutine problem solving that we discuss here.

3.2 Detecting Non-Routine Problems

In LIDA, understanding or give meaning to the current situation happens at the end of the third step of the cognitive cycle. That is, incoming stimuli activate low-level feature detectors in Sensory Memory. The output is sent to Perceptual Associative Memory where higher-level feature detectors feed in to more abstract entities such as objects, categories, actions, events, etc. The resulting percept is sent to the Workspace where it cues both Transient Episodic Memory and Declarative Memory producing local associations. These local associations are combined with the percept to generate a current situational model, the agent understands of what's going on right now.

In the "consciousness" phase parts of the situational model picked by attention codelets and their coalition is moved to Global Workspace, in which competition takes place to select the most salient, the most relevant, the most important, the most urgent coalition whose contents become the content of consciousness that are broadcast globally.

Using the conscious contents, possible action schemes (mainly behavior streams) are recruited from Procedural Memory. A copy of each such is instantiated with its variables bound and sent to Action Selection system (behavior net), where it competes to be the behavior selected for this cognitive cycle. The selected behavior triggers Sensory-Motor Memory to produce a suitable algorithm for the execution of the behavior. Its execution completes the cognitive cycle.

In LIDA, there are two scenarios in which a non-routine problem can arise. 1) If LIDA encounters an unexpected type of situation that could not exactly be categorized in any one of the known classes of problems; it is a *non-routine problem situation* since there will not be known behavior streams to handle the situation. 2) In an active behavior stream (dominant goal hierarchy), an executing behavior (dominant goal context) may fail to achieve its expected outcome, which happens if LIDA encounters perturbation in its environment, a variation in its domain, or if any one of the acting behavior codelets malfunctions. This situation crops up if there is no alternative routine solution that corrects the *novel error situation*. In both scenarios, the non-routine problems arise from instances of *execution error*, which eventually is reflected in the workspace.

Considering the first scenario, once the understood novel situation is written to the workspace, there may not be an attention codelet that will respond to it. This indicates that non-routine problem solving will require a special attention codelet whose task is to respond to unknown situations. But how is such a codelet to recognize such novel problem situations without knowing about all the routine occurrences? It seems to be similar to the problem faced by the immune system and may well require the same kind of solution. Just as a human would, IDA learns new procedural methods incrementally based on related, known concepts. This is more commonly described as "learning at the fringe" (Doignon & Falmagne, 1985). Let's see how this could be done. After a reasonably short time of a novel problem situation is written in the workspace, a special, nonroutine problem solving attention, codelet notices that no other attention codelet is attempting to deal with this new situation and competes to bring the information to consciousness. Assuming that the nonroutine problem solving attention codelet wins the competition for consciousness, its content of non-routine situation will be broadcast global and this global reach allows recruiting the appropriate mental resources that handle the situation.

How does LIDA detect non-routine problems that arise from execution errors in running behavior streams? Each behavior, besides behavior codelets, has one or more expectation codelets under it. When a behavior starts execution and dispatches its behavior codelets for action, at the same time it activates each of its expectation codelets. The role of an expectation codelet includes watching, evaluating and, perhaps, reporting on the outcome of the action. The function of watching events that are associated with the action of its behavior makes an expectation codelet a particular kind of attention codelet. The monitored changes resulting from a behavioral action are characterized and formulated as an outcome or result. If the actual outcome is different from the expected outcome, the expectation codelet detects an error situation. The execution error, as a non-routine problem, is described as an unfulfilled part of the outcome (goal), the fulfilled part of the outcome (state to be protected) if there is any, and the contextual state at the point of failure. The contextual state includes the perceptual context from working memory, long-term memory and the internal state in the behavior net. The expectation codelet that detects an execution error moves to the Global Workspace and competes to bring the error information to consciousness.

While there are no behavior codelets that directly respond to the broadcast non-routine problem situation, various behavior codelets that recognize pieces of the information will respond to varying degrees, depending on how much of the information they are familiar with. These initial respondents will become the base or the seed from which the search for possible solutions is initiated. With the sequence of cognitive cycles, the search space is constrained by what is available in the conscious content. In addition, their activation levels at the time, corresponding to each behavior codelet's assessment of its applicability, are used to further bias the search for a solution. The core of this chapter is to present a meshing (Glenberg, 1997) mechanism that builds new goal structures (behavior streams) that could handle a novel situation.

Following global workspace theory, we conjecture that creative solving of non-routine problems by humans is accomplished using consciousness in the role of a recruiter of relevant resources. We further conjecture that such creative problem solving also requires a meshing mechanism capable of cutting and pasting various goal contexts along with their associated processors. Finally, we conjecture that some sort of mechanism (processor(s)) is needed for recognizing such problems and training attention on them.

Translating all this into computational terms, we conjecture that LIDA could be provided with attention codelets that recognize non-routine problems and bring them to *consciousness* and with a sufficiently capable cut and paste mechanism for behavior streams and scripts, so that it will be capable of providing clever, possibly creative, solutions to non-routine problems from its domain.

In general, LIDA’s perception and the memory modules may need to be involved in the recognition of nonroutine situation and in the process of constructing creative solutions. But the scope of the research here is limited to the integration of non-routine problem solving mechanisms with the action selection and “consciousness” modules. In the coming sections of this chapter we will present the details of a mechanism that could build a novel solution to a problem, which is recognized to be nonroutine in a given situation.

4. Non-Routine Problem Solving Mechanism

When we humans are faced with a problem to solve, we often create in our mind different strategies or possible solutions. We imagine the effects of executing each strategy or trial solution without actually doing so. This is similar to a kind of internal virtual reality or a simulated run. Eventually, we decide upon one strategy or trial solution, and try solving the problem using it. This process is called deliberation (Sloman 1999). The non routine problem solving is a deliberative process to device solution for novel problems. As Glenberg (1997) suggests such process calls for meshing – use of chunks of prior knowledge towards obtaining solutions to novel problems.

In this section we will present the mechanism that searches a solution (behavior stream) out of existing schemes and doing so using consciousness to guide the search over multiple cognitive cycles.

4.1 Basis for the mechanism

We would like to point out two driving principles for building LIDA’s non-routine problem solving mechanism. The first is that *complex processes are built from simple and primitive processes*. The complexity arises from the structure that governs the interaction of the primitive processes in time and space. The second is that *creativity is an important feature in handling non-routine problems*; we believe that creativity is the process of using simple units to build larger and more complex structures such as behaviors and behavior stream templates (procedural schemes). An important capability of non-routine problem solving is to create new behavior codelets using refined domain knowledge, but this level of creativity needs further investigation and we do not address it here. That is, we limit the primary repertoire to the level of codelets, which can be considered as neuronal groups (Edelman, 1987).

An attention or expectation codelet that detected a non-routine problem may win the competition to “consciousness,” and the problem information content is broadcast. Then behavior codelets find themselves relevant to the pieces of the broadcast information and instantiate the associated behavior streams. Particularly, there is a special *non-routine problem solving (NRPS) behavior codelet* that becomes relevant to all broadcasts related to all non-routine problem situations. The NRPS behavior codelet activates the NRPS module to facilitate the deliberative process of searching solution(s).

Non-routine situations should be recognized and brought to “consciousness” by nonroutine attention codelets. Our approach to managing nonroutine problems is functionally equivalent to what Shallice (1988) call a Supervisory Attention System (SAS), which is associated with the prefrontal cortex. SAS could correct errors and determine novel action plans, and it has a number of functions (Shallice, 1988). (i) SAS Monitor: This can be considered as a nonroutine situation detection system. As is the case in IDA, SAS Monitor is connected to the long-term working memory. Novelty detection, which may take multiple cognitive cycles, notices departures from what is expected while looking at long-term working memory (LTWM) (Ericsson & Kintsch, 1995). Looking is done by cuing the LTWM with the content of current percepts and retrieved content from different memories, which in turn are cued by the current situation. In IDA, this role should be addressed in the perception and the different memory modules; (ii) SAS Modulator: The SAS must activate relevant actions and goal structures while attenuating irrelevant actions, particularly to try out available solutions to the novel situation. In IDA, this functionality is realized by behavior codelets that respond to pieces of information from nonroutine problem handling related broadcasts and which instantiate relevant behavior streams. This is similar to adaptation by assimilation (Piaget, 1983) in the sense of using existing solutions to handle novel situations; (iii) SAS Generator: SAS is used to generate novel strategies (such as altering goal structures or action plans) to solve nonroutine problems. Our research contribution in this chapter is to present a design for a mechanism that realizes the function of the SAS generator.

Under LIDA framework, a Non-Routine Problem Solving (NRPS) mechanism should be a consciously mediated reasoning system that breaks down and combines procedural knowledge pieces to produce new behavior streams. An AI planner is a type of problem solver and a deliberative action selection system. But, in our system deliberative action requires "consciousness," which is not assumed in AI planners. Particularly, we propose a non-routine problem solving module of IDA as a special planning (regressive and dynamic) behavior stream (goal structure), called *NRPS behavior stream*. As any other behavior stream, the NRPS behavior stream gets instantiated and executed in multiple cognitive cycles. But its particular task is to construct a new behavior stream as a solution for a given non-routine problem situation. As such, NRPS behavior stream can be called a *Meta behavior stream* since it has knowledge about other behavior streams.

4.2 The Mechanism: NRPS Behavior Stream

To incorporate a non-routine problem solving mechanism as a special goal structure, we assume two constraints about schemes in the procedural memory. (i) Each behavior codelet (primitive action) is associated with a corresponding expectation codelet. (ii) Based on the broadcast content, a recruited behavior codelet instantiates itself and relevant behavior streams in one of the two modes: (a) Execution mode – so that the instantiations become part of the action selection dynamics of an agent; that is, when a behavior codelet executes, its associated expectation codelet knows how to monitor the execution effects and how to report the outcome. (b) Non-routine problem solving or regressive mode – instantiations become part of the pool of actions that is available for the solution searching process.

The first constraint allows the NRPS behavior stream be able to cut and paste behavior codelets and coupled expectation codelets as atomic units easily. A behavior codelet and its corresponding expectation codelet are reorganized under different behavior as a unit during the solution search process.

The second constrained differentiates the role behavior codelets play after instantiation. According to Baars (1988) intention is future-directed, nonqualitative mental representation of one's own action with a potential to dominate selective attention or to become conscious content. In the regressive reasoning process during non-routine problem solving, behavior codelets could be called intention codelets (behavior codelets with goal image or intention mode). Intention codelets are a special case attention codelets as they compete for consciousness with the goal image of future state as a content, which in turn enables recruitment of processors and subgoals (representing procedural constructs) that help to achieve the future goal state. Similarly, in non-routine problem solving or regressive mode, behavior nodes, goals nodes are underlain by intention codelets and expectations codelets. Intention codelets, representing scheme, are part of IDA's primary repertoire and a special case attention codelet will compete to bring the corresponding goal images to consciousness. The NRPS mechanism, during its regressive solution search process, involves intention codelets corresponding the top goal and subgoals. Our approach *hypothesizes that there is an intention codelet, as a primary repertoire, corresponding to and representing every scheme node (goal state)* including for those at the highest level of abstraction.

The NRPS behavior stream is shown in figure 3. It shows the instantiated copy of the stream following the understanding of the problem situation, selective attention (conscious broadcast), the recruitment of schemes as relevant resources from procedural memory, and their instantiation and variable binding as they become active in the action selection system. The NRPS mechanism is based on a partial-order planning (POP) system (Sacerdoti 1977; Tate 1990; Wilkins 1990) - and the service of "consciousness" to recruit relevant resources in its solution searching process. Variations of partial-order planners allow rich domain knowledge representation and are used in real world domains.

O-Plan (Tate, Drabble, & Kirby, 1994) and Sipe-2 (Wilkins, 1990; Wilkins et al., 1995) are domain independent partial-order planners that have been used in a number of real world domains. Such planners allow plan repairing and interaction with humans, which are features particularly important in IDA's domain. Erol, Handler, and Nau (1994) presented a formal definition of partial-order planning and proved that it is sound and complete. Wilkins (2001) discussed the comparative advantages of partial-order planners.

Standard POP systems usually take initial state, goal state, and all available actions of an agent and return a plan if one is found. For our NRPS mechanism, the POP system should be modified so that it could deal with a limited set of actions to start planning and to allow "conscious" deliberation if an impasse or a contentious point is reached in the planning process. An impasse is reached if a planner cannot find an action (in the current set of actions) that could satisfy an open subgoal, which we call an impasse subgoal (or contentious subgoal).

The NRPS behavior stream uses a modified POP system (the detailed algorithm can be found Negatu, 2010) that takes a starting-plan, current-pool-of-actions or possible steps and a possible impasse subgoal and then returns resulting-plan, impasse-subgoal, and result-explanation, which provides human readable information on the result. A "consciously" mediated planning process continues until a consistent plan is obtained as a solution (no impasse-subgoal) or no solution is found (the impasse-subgoal exists and no new relevant action could clear the impasse-subgoal.)

As we discussed before, the non-routine problem situation becomes conscious content once the coalition of the NRPS attention codelet wins the competition in the Global Workspace and its content (problem description) is globally broadcasted. This broadcast recruits or primes the NRPS behavior stream (in the procedural memory) and instantiate it for action in the Action Selection (behavior net) system. Once instantiated, it guides a multi cognitive cycle solution search. Next, we will discuss the role of each behavior in the NRPS behavior stream (figure 3).

i. Initiate Problem Solving

This behavior is the one that sets the problem-solving context. Its variables are bound by the underlying NRPS behavior codelet that responds to a broadcasted non-routine problem description (initial state and goal state). Along with the instantiation of the NRPS behavior stream, relevant schemes (mostly behavior codelets) also respond to the broadcast of the same non-routine problem description. These instantiated and bound schemes form the initial set of actions (operators) available to the planner. When this behavior executes, it asserts the readiness of the initial state, goal state, and the initial actions (operators) to the other behaviors in the NRPS behavior stream. As such, it sets the context for solving the problem in hand.

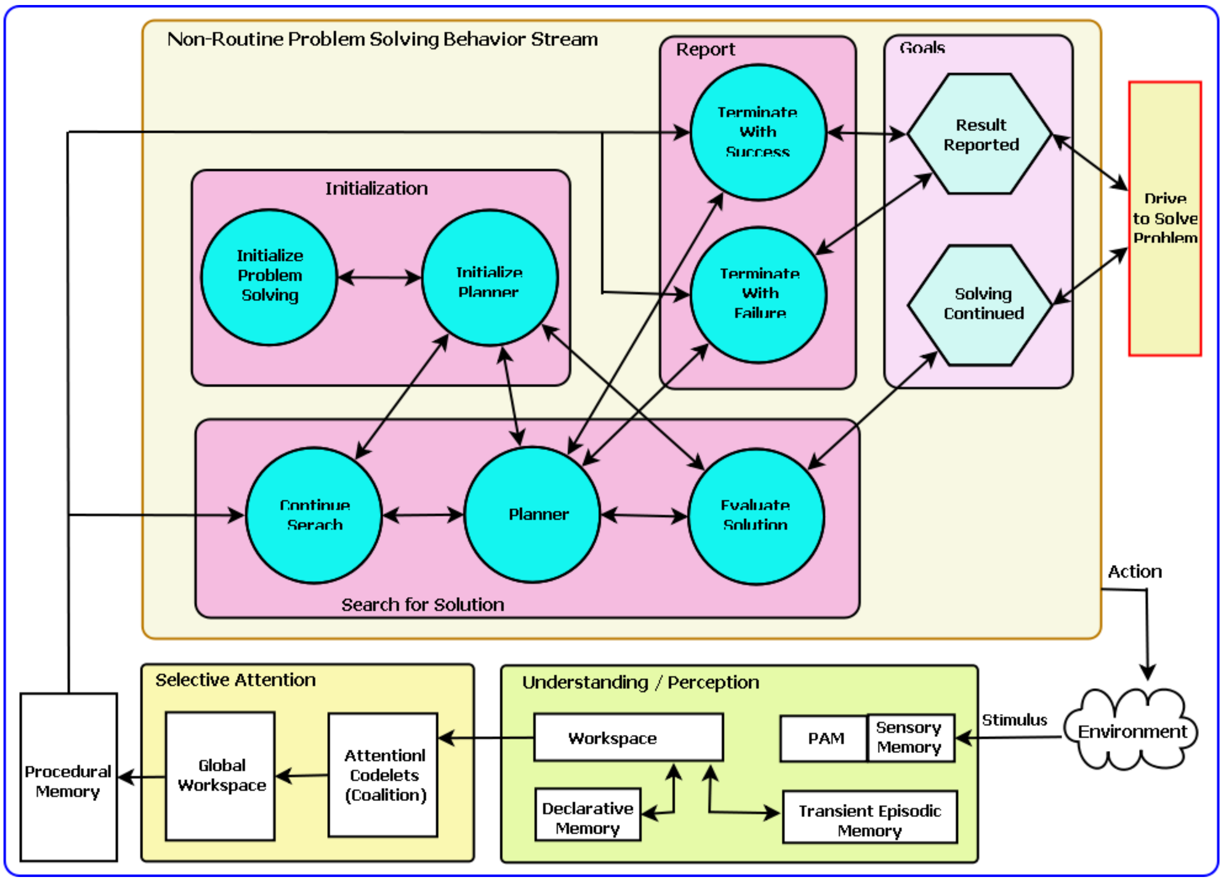

Figure 3: Non-routine problem solving (NRPS) module is a special behavior stream (goal-context hierarchy), which guides a deliberative process for problem solving over multiple cognitive cycles.

ii. Initialize Planner

Using the problem context set by the *Initiate Problem Solving* behavior, this behavior compiles the planner initialization parameters. Particularly, on its execution it does the following: (i) Using the initial and goal states, it creates the initial empty plan (in which the *Start* dummy plan action has literals in the initial state of the problem as its effects, and the *Finish* dummy plan action has literals in the goal state of the problem as its preconditions). (ii) Nondeterministically, it chooses an open subgoal (one of the literals from the goal state of the problem) and sets it as an impasse or contentious subgoal, which may or may not be an actual impasse but needs to be set for the proper initialization of the planner. (iii) It initializes an empty list, which is used to store "consciously" deliberated impasse subgoals. Each detected impasse subgoal is registered in this list.

iii. Planner

This behavior executes to build a new plan based on the previous plan and the current pool of actions. At the end of the planning process, this behavior asserts an impasse subgoal (set to null if there is none), result-explanation, and the new plan. For the details of the planning algorithm, refer to Negatu (2010).

iv. Evaluate Solution

This behavior evaluates the solution search state. Particularly, it checks whether the search for a solution should continue, whether a solution has been found, or if the search has failed to find a solution. The checking process is simple and based on the asserted value of the impasse subgoal and whether this impasse has been deliberated in the earlier cycles or not. If no impasse subgoal left to be resolved, then the evaluation is a *success in finding a solution* with the current plan being the solution to the problem. If an impasse subgoal exists and is in the list of deliberated impasse subgoals, then it has been dealt before and therefore the evaluation is a *failure to find a solution* to the problem. The result-explanation gives human readable information feedback to a human on how the failure arose. If an impasse subgoal exists, but is not in the list of deliberated impasse subgoals, then the evaluation is to *continue the search for a solution*.

The actions of behavior codelets under this behavior produce internal state change that asserts one of the three evaluation statuses (*success in finding a solution, continue the search for a solution, and failure to find a solution*) and possibly the impasse subgoal. After the internal state change is perceived, the evaluation content as a percept and whatever it cues from memory is copied to the Global Workspace. Relevant attention codelets, which watch for the solution evaluation in Global Workspace, try to bring the information to "consciousness." Upon winning the competition, the content of the attention codelet and its coalition is broadcast. Relevant NRPS related behavior codelets respond to the broadcast and bind the evaluation content to the appropriate behaviors in the NRPS behavior stream. If an impasse subgoal exists, its broadcast causes behavior codelets or schemes that could satisfy the subgoal or with the subgoal in their effect /result list to respond. Thus, they become the new additions or recruits in the set of actions available to the *Planner* behavior for its next cycle of search for a solution. That is, consciousness is used as filter and recruiter in the solution searching process and this is, we think, the main advantage of this mechanism over other planning systems.

v. Continue Search

This behavior is underlain by a behavior codelet that responds to a broadcast of an evaluation that asserts *continue the search for solution*, which has been understood as such and reflected in the Global Workspace as a result of the action of *Evaluate Solution* behavior. After its variables are bound by its underlain behavior codelet and upon its execution, this behavior compiles a new set of actions, which consists of the union of the existing actions and the newly arrived actions. This behavior, on its execution, asserts the preconditions that allow the planner behavior to continue the search with the possible addition of newly recruited actions.

vi. Terminate with Failure

This behavior is underlain by a behavior codelet that responds to a broadcast of an evaluation with content of *failure to find solution* to the problem. After its variables are bound by its underling behavior codelet, this behavior executes to compose an extended explanation on the failure situation and to terminate or give up on the solution search process for the problem. The failure explanation could help as input for human experts so that they can provide additional knowledge to the agent. Although it is beyond the scope of our discussion here, actual explanation composition could be done by another behavior stream.

vii. Terminate with Success

This behavior is underlain by a behavior codelet that responds to a broadcast of an evaluation with the content of *success in finding a solution* to the problem. The last plan produced by the planner is a solution. Upon its execution, this behavior converts the plan into a corresponding behavior stream template or scheme. The conversion happens based on two facts. (a) The solution as a complete plan is a total order of actions (behavior codelets and associated expectation codelets that can be applied sequentially to an initial state to produce a goal state. (b) The solution as a totally ordered plan could be specified by a set of partially ordered complete plans (behavior streams). For the current version of LIDA's action selection mechanism, the conversion also should satisfy a critical criterion - behavior codelets that underlie a behavior should be independent or should be able to execute in parallel or in any order without affecting each other. In planning terms, two actions are independent of each other; if any change of order happens between the actions, the plan remains consistent – no cycle is introduced in the ordering constraints of the plan and no conflicts arise in any one of the causal links of the plan.

The simplest conversion from the plan solution to a behavior stream template is to have each action (behavior codelet and expectation codelet) underlie a behavior. Such is a correct solution and may be the only valid way to have a conversion, but it would be the worst case scenario particularly in execution efficiency. We suggest the following heuristics for a better conversion to be applied in that order whenever doing so keeps the partial plan consistent: (a) Map a set of behavior codelets to an existing behavior. (b) Replace independently executable behavior codelets in the plan solution with a new behavior, and to do so by limiting the maximum number of behavior codelets that can underlie new behaviors. (c) Each action (behavior codelet and coupled expectation codelet), which is not converted in the previous two heuristics, is mapped to underlie a behavior by itself. Reusing knowledge pieces is an important feature of agents; the first heuristics allows the reusing of existing structures in the behavior network. Having multiple behavior codelets that underlie a single behavior improves performance since the selection of a behavior in the behavior network dynamics leads to the parallel execution of the underlying codelets.

The actual solution to the non-routine problem at hand is a novel behavior stream template or scheme with the possibility of a new goal structure built using only existing behaviors, or only new behaviors, or a mix of new and old behaviors. The *Terminate with Success* behavior effects the storing of the new behavior stream template in the procedural memory – its action change internal state that to be understood as the *availability of a solution to the non-routine problem* at hand and consequently the success information becomes content in the workspace. An appropriate attention codelet, upon winning the competition in Global Workspace, the broadcast of its content (success of finding the solution) causes the incorporation of the new scheme into the procedural memory.

Behavior steams are goal structures with hierarchy. Although the POP algorithm are known to produce flat plans, the free interconnection of schemes (behaviors and behaviors streams) based on their causal links and the heuristics of constructing new schemes by using already existing schemes as components maintain the hierarchical nature of behavior streams and the reusability of schemes. This was possible since schemes with different levels of abstraction are treated the same way – schemes.

NPRS Behavior Stream at Work

We have discussed details of the role of each behavior in the NRPS behavior stream in the solution search process for a given non-routine problem. Here, we point out the flow of information processing in the instantiation and execution of the NRPS behavior stream.

To start with, a non-routine problem situation is perceived and associated with what could be remembered; then the problem description is eventually stored in the workspace. LIDA does this in its understanding phase – the first three steps of its cognitive cycle (CC-1 to CC-3). An attention codelet that watches for a non-routine problem situation in the workspace becomes active, with its coalition of information codelets, and the coalition competes in the Global Work space to get access to "consciousness" (CC-4). If and when the competition is won, the coalition gains access to the global workspace and the problem description is broadcasted (CC-5). The *NRPS behavior codelet* responds to the broadcast of the non-routine problem description. Other behavior codelets may find themselves relevant to pieces of information in the same broadcast and can recruit resources that could help to construct a solution for the problem (CC-6). The responding behavior codelets bind their variables with the relevant information in the broadcast. The NRPS behavior codelet instantiates and binds the *NRPS behavior stream* (CC-7). The other recruited behavior codelets (in NRPS or planning mode) with the behavior streams they prime become part of the initial set of actions (operators) for the problem solving process.

Once the NRPS behavior stream is instantiated, it is part of the dynamics in the Action Selection system or behavior network; then the dynamics there choose a behaviors for control of action (CC-8) and get executed (CC-9) one at a time and over multiple cognitive cycles. The NRPS behavior stream guides the search for a solution to the problem with the help of "conscious" mediation when an impasse arises. The *Initiate Problem Solving* and the *Initialize Planner* behaviors are executed only to initialize the problem solving process. While the *Planner* behavior has the task of planning in the solution search process, the *Evaluate Solution* behavior has the task of evaluating the output of the *Planner* behavior and its execution produce change in internal state that effect the writing of the understood evaluation in the work space (CC-1 to CC-9). The standard "conscious" mediation process (CC-4 & CC-5) feeds back the evaluation content and possibly recruits additional resources (schemes).

If the evaluation is to continue the problem solving process, a relevant behavior codelet binds (or rebinds) and activates the *Continue Search* behavior, which has the task of setting up and asserting the parameters required by the *Planner* behavior. As long as the evaluation is to continue the search and it being the dominant goal context (with one of its behaviors is a winner in the behavior net dynamics), the NRPS behavior stream executes in a cycle: execution of *Continue Search* behavior is followed by execution of *Planner* behavior, followed by execution of *Evaluate Solution* behavior, followed by "conscious" mediation, followed again by execution of *Continue Search* behavior.

The *NRPS behavior stream* terminates with failure or success. To do so, a "consciously" mediated result processing, which involves all or most of the steps of LIDA's cognitive cycle, takes place. The process involves many codelets that are particularly designed to the nonroutine problem solving task. Besides behavior codelets that underlie the NRPS behavior stream, there are many attention and information codelets with the role of watching for or recognizing workspace content that relates to nonroutine problem solving. Among those are attention and information codelets that watch for a result status of failure or a success in the solution search process. If consciously mediated evaluation content asserts failure in finding a solution to the problem the result reporting action is executed by the *Terminate with Failure* behavior with the task of writing an explanation of the failure situation. If the consciously mediated evaluation content asserts success in finding a solution to the problem, the result reporting action is executed by the *Terminate with Success* behavior with the task of saving the solution as a behavior stream template in the procedural memory or scheme net and writing the availability of a solution to the stated problem in the workspace. The termination state written in the workspace could be brought to consciousness by relevant attention codelets. Consequently, failure states could prompt the agent to explain the problem to humans and success states will allow the agent to apply the new solution.

5. Conclusion

Nonroutine problem solving is the process of devising appropriate stream of actions as a response to novel or unexpected situations. Devising solutions is a type of learning or adaptation, which requires paying focus of attention or involvement of consciousness in handling the situation (Baars, 1997). Norman and Shallice (1986) and Shallice (1988) have proposed a functional model for a control of both routine (automatic) and non-routine behavior. This functional model is broadly equivalent to LIDA's decision making mechanism. In the Shallice's mechanism, non-routine problem is an attention-based learning managed by a mechanism called Supervisory Attention System (SAS), which is mapped to the prefrontal cortex (Shallice, 1988). Focusing on the deliberate action of the mechanism, working memory provides the attention system to access the current intentions, goals, and relevant past episodes. The SAS enables error correction, troubleshooting, handling non-routine problems using supervisory sub-functions such as SAS Monitor, SAS Modulator, and SAS Generator. The *SAS Monitor* is a non-routine problem situation or novelty detector. It is primarily connected to working memory. Non-routine situation detection is departure or mismatch of known contents from what is perceived in the world, intended action, outcome of executed action. These are related to what is stored in episodic, procedural, and working memory pieces. The monitoring sub-function is a trigger mechanism to the other supervisory sub-functions. Although we do not address the function of the SAS Monitor here, we think that this function in LIDA is connected to internal perception and the different memory types the agent may have. The *SAS Modulator* function attenuates the activation of inappropriate actions and enhances the activation of alternate but attended goal structures. Shallice suggests few possible modulatory responses which we do not cover here. The SAS Modulator function is a type of applying existing solutions or behavior streams to new problems – which is adaptation by assimilation (Piaget, 1983). The *SAS Generator* is triggered if the modulatory response by the SAS Modulator does not succeed; which asserts the unavailability of recognized strategy to the situation in hand. The SAS Generator must be able to produce a novel strategy (new goal hierarchy or partial plan) that could handle the novel or non-routine problem situation.

As the case of Global Workspace Theory, the SAS framework suggests learning is triggered and happens with the investment of attentional resources or involvement of consciousness. Our presentation in this chapter addresses the role of the SAS Generator – devising new behavior streams as a response to non-routine problems. While SAS Generator (Shallice, 1988) is a functional model to device solutions to non-routine problems, the presented mechanism here is a more concrete and detailed specification of the function of Shallice’s SAS Generator; the specification is also unique in the sense that it tries to realize the function as per Baar’s Global Workspace Theory in general and as the integrative cognitive agent framework of LIDA in particular.

To manage a nonroutine problem, first, the problem should be recognized, and then, a novel action plan (if there is one to find) should be produced as a solution to the problem. We presented a mechanism that addresses the later with the help of conscious mediation. The hypothesis is that *consciousness* provides an effective resource recruitment mechanism in the search process for a novel solution. This non-routine problem solver can be seen from the point of view of the human developmental process. We are born with basic primitive capabilities, which could correspond to the codelets in our mechanism or fine-grained level of competencies. We inherit some behaviors and develop other behaviors as experience in the environment progresses. The new behaviors are developed out of the established behaviors and the underling primitive capabilities. To start with, most problems are non-routine. Progressively, we accumulate routine problem solving capabilities via different ways: by instructions from other humans, by direct feedback from the interactions in the environment, by imitation, etc. In our mechanism, routine problem solving capabilities correspond to ready-made behavior streams (partial plans) in the procedural memory or scheme database (in the plan space). The behavior streams, the behaviors in them and the underlying behavior codelets are building blocks to generate solutions to non-routine problems. As experience grows, the non-routine problem becomes routine. If a routine task that uses a routine solution (behavior stream) is performed repetitively (or over learned), the routine task may become an automatic task; this is an automatization process, which is presented somewhere else (Negatu et al., in review; Negatu, 2010). Automatization can be used as a foundation to chunk over learned sequences of actions into a larger unit, such as a behavior. The presented nonroutine problem solver is a type of adaptation or learning system that follows the Edlman's (1987) Neuronal Group Selection mechanism, which itself is inspired by Piaget's (1952) theory of cognitive development.

In general, planners produce a sequence of actions. In our case, the non-routine problem solving module, with a planner at the center, is used to produce a behavior stream template (or task network) that could be instantiated to perform actions in a certain sequence. The actual sequencing of the actions and conflict resolutions happen in the dynamics of the behavior net where instantiated streams reside. So, in LIDA, the NRPS stream and the behavior net contribute in plan generation, and the behavior net does the plan execution. Both the plan generation and the plan execution are edited by consciousness. Non-routine problem solving, as a deliberative process in IDA, is consciously mediated, and this enables IDA to interact with humans, using the service of its perception module. It is nearly impossible to preprogram the complete domain knowledge; so, for LIDA and for any other agent system, an effective interaction with humans is important so that humans input their knowledge at contentious decisions.

We hypothesize that non-routine problem solving in the LIDA architecture is a construction of a new goal hierarchy with a control of a unique behavior stream, itself a goal hierarchy, operating over multiple cognitive cycles, with the shaping of partial plans of action at each cycle. We believe that integration of the AI planning system and LIDA's consciousness process, along with the human interaction capability in LIDA, contribute towards the development of effective, non-routine problem solving systems.

References

Arp, R. (2007). Scenario visualization: An evolutionary account of vision-related creative problem solving. Cambridge, MA: MIT Press.

Baars, B.J. (1988). A Cognitive Theory of Consciousness. Cambridge: Cambridge University Press.

Baars, B.J. (1997). In the Theory of Consciousness. Oxford: Oxford University Press.

Baars, B. J., & Franklin, S. (2003). How conscious experience and working memory interact. Trends in Cognitive Science, 7, 166-172.

Baars, Bernard J and Stan Franklin. (2009). Consciousness is computational: The LIDA model of global workspace theory. International Journal of Machine Consciousness.

Baddeley, A D and G J Hitch. 1974. Working memory. In The psychology of learning and motivation, ed. G A. Bower:47–89. New York: Academic Press.

Barsalou, L. W. (1999). Perceptual symbol systems. Behavioral and Brain Sciences 22:577-609.

Bogdan, R. (1994). Grounds for cognition: How goal-guided behavior shapes the mind. Hillsdale, CA: LEA Publishers.

Brooks, R. A. 1986. A robust layered control system for a mobile robot. IEEE Journal of Robotics and Automation. RA-2:14-23.

Brooks, R.A. 1991. Intelligence without representation. Artificial Intelligence 47: 139-159.

Cañamero, D. (1997). Modelling Motivations and Emotions as a Basis for Intelligent Behavior. In Proceedings of the First International Symposium on Autonomous Agents, AA'97, Marina del Rey, CA, February 5-8, The ACM Press.

Chaput, Harold H., Benjamin Kuipers, and Risto Miikkulainen. 2003. Constructivist learning: A neural implementation of the schema mechanism. In Proceedings of WSOM '03: Workshop for Self-Organizing Maps. Kitakyushu, Japan.

Cosmides, L., & Tooby, J. (1992) The psychological foundations of culture. In J. Barkow, L. Cosmides, & J. Tooby (Eds.), The adapted mind (pp. 19-136). New York: Oxford University Press.

Crick, F. (1994). The astonishing hypothesis. New York: Simon & Schuster.

D'Mello, S. K., Franklin, S., Ramamurthy, U., & Baars, B. J. (2006). A Cognitive Science Based Machine Learning Architecture. AAAI Spring Symposia Technical Series, Stanford CA, USA. Technical Report SS-06-02 (pp. 40-45). AAAI Pres.

Doignon, J.-P., & Falmagne, J.-C. (1985). Spaces for the assessment of knowledge. International Journal of Man-Machine Studies, 23, 175-196.

Drescher, G. (1991). Made Up Minds: A Constructivist Approach to Artificial Intelligence, Cambridge, MA: MIT Press.

Edelman, G. M. (1987). Neural Darwinism: the theory of neuronal group selection. New York: Basic Books.

Erol, K.; Hendler, J.; & Nau, D.S. (1994). UMCP: A Sound and Complete Procedure for Hierarchical Task-Network Planning. In Proc. AIPS. Morgan Kaufman.

Franklin, S. (1997). Autonomous Agents as Embodied AI. Cybernetics and Systems 28:499–520.

Franklin, S. (2001b). An Agent Architecture Potentially Capable of Robust Autonomy. AAAI Spring Symposium on Robust Autonomy; American Association for Artificial Intelligence; Stanford, CA; March.

Franklin , S. 2001. Automating Human Information Agents. In Practical Applications of Intelligent Agents, ed. Z. Chen, and L. C. Jain. Berlin : Springer-Verlag.

Franklin, S., Baars, B.J., Ramamurthy, U., & Ventura, M. (2005). The Role of Consciousness in Memory" in Brains, Minds and Media, Vol.1, 2005 (bmm150) (urn:nbn:de:0009-3-1505).

Franklin, S., & Graesser, A.C. (1997). Is it an Agent, or just a Program?: A Taxonomy for Autonomous Agents. In Intelligent Agents III. Berlin: Springer Verlag.

Franklin, S., & A. Graesser. (2001). Modelling Cognition with Software Agents. In CogSci2001: Proceedings of the 23rd Annual Conference of the Cognitive Science Society, ed. J. D. Moore, and K. Stenning. Mahwah, NJ: Lawrence Erlbaum Associates; August 1-4, 2001.

Franklin, Stan, Uma Ramamurthy, Sidney K. D'Mello, Lee McCauley, Aregahegn Negatu, Rodrigo Silva L., and Vivek Datla. (2007). LIDA: A computational model of global workspace theory and developmental learning. In AAAI Fall Symposium on AI and Consciousness: Theoretical Foundations and Current Approaches. Arlington, VA: AAAI.

Freeman, W.J. (1995). Societies of Brains. A Study in the Neuroscience of Love and Hate. Hillsdale NJ: Lawrence Erlbaum.

Gardner, H. (1993). Multiple intelligences: The theory in practice. New York: Basic Books.

Glenberg, A. (1997). What memory is for. Behavioral and Brain Sciences, 20, 1-19.

Hofstadter, R. D., & Mitchell, M. (1994). The Copycat Project: A model of mental fluidity and analogy-making. In Advances in connectionist and neural computation theory, Vol. 2: Analogical connections, eds. K. J. Holyoak & J. A. Barnden. Norwood N.J.: Ablex.

Humphrey, N. (1992). A history of the mind: Evolution of the birth of consciousness. New York: Simon & Schuster.

Kanerva, P. (1988). Sparse Distributed Memory. Cambridge MA: The MIT Press.

LaBerge, D. 1995. Attentional processing: the brain's art of mindfulness. Harvard University Press, 1995.

Laird, E. J., Newell, A. & Rosenbloom, P. S. (1987). SOAR: An Architecture for General Intelligence. Artificial Intelligence 33:1–64.

Lebiere, C., and Anderson, J. R. (1993). A connectionist implementation of the ACT-R production system. In *Proceedings of the Fifteenth Annual Meeting of the Cognitive Science Society,* pp. 635-640. Hillsdale, NJ: Erlbaum.

Maes, P. (1989). How to do the right thing. Connection Science, 1, 291-323.

Marshall, J. 2002. Metacat: A self-watching cognitive architecture for analogy-making. In 24th Annual Conference of the Cognitive Science Society:631-636.

Negatu, S. Aregahegn (2010). Decision Making System for Cognitive Machines, Berlin, LAP Lambert Academic Publishing.

Negatu, A. S., & S. Franklin. (2002). An action selection mechanism for 'conscious' software agents. Cognitive Science Quarterly 2:363-386.

Negatu, A. S., T. L. McCauley, & S. Franklin. (in review). *Automatization for Software Agents.*

Newell, A. (1990) Unified Theories of Cognition. Cambridge, MA: Harvard University Press.

Norman, D. A. and Shallice, T. (1986). Attention to action: Willed and automatic control of behaviour. In Davidson, R. J., Schwartz, G. E., and Shapiro, D., editors, Consciousness and Self-Regulation: Advances in Research and Theory. Plenum Press.

Piaget, Jean. (1952). The Origins of Intelligence in Children. New York: International University Press.

Piaget, Jean. (1983). Piaget's theory. In P. Mussen (ed.). Handbook of Child Psychology. 4th edition. Vol. 1. New York: Wiley.

Pinker, S. (1997). How the mind works. New York: W. W. Norton & Company.

Post, Emil Leon. (1943) Formal Reductions of the General Combinatorial Decision Problem, *American Journal of Mathematics* **65** (2): 197-215, 1943.

Rao, Rajesh P.N and Olac, Fuentes. (1998). Hierarchical learning of navigational behaviors in an autonomous robot using a predictive sparse distributed memory. Machine Learning 31: 87-113.

Sacerdoti, E.D. (1977). A Structure for Plans and Behavior, Elsevier-North Holland.

Searle, J. (1992). The rediscovery of the mind. Cambridge, MA: MIT Press.

Shallice, T. (1988). From Neuropsychology to Mental Structure. Cambridge: Cambridge University Press.

Sloman, A. (1999). What Sort of Architecture is Required for a Human-like Agent? In Foundations of Rational Agency, ed. M. Wooldridge, and A. Rao. Dordrecht, Netherlands: Kluwer Academic Publishers.

Sun, R. (1997). Learning, action, and consciousness: A hybrid approach towards modeling consciousness. Neural Networks, 10(7), 1317–1331.

Tate, A. (1990). Generating Project Networks. In Alen, J.; Hendler, J.; and Tate, A., editors 1990, Readings in Planning. Morgan Kaufman. 291-296.

Tate, A., Drabble, B., & Kirby, R. B. (1994). O-Plan2: an open architecture for command, planning, and control. In Fox, M, and Zweben, M., editors, Intelligent Scheduling. Morgan Kaufman, San Francisco, CA. 213-239.

Varela, F. J., E. Thompson, and E. Rosch. 1991. The Embodied Mind. Cambridge, MA: MIT Press.

Wilkins, D. E. (1990). Domain-independent Planning: Representation and Plan generation. In Alen, J.; Hendler, J.; and Tate, A., editors 1990, Readings in Planning. Morgan Kaufman. 319-335.

Wilkins, E. D. (2001). A Call for Knowledge-based Planning. AI Magazine, Spring 2001, Vol. 22, No. 1.

Wilkins, D.E., Myers, K.L., Lowrance, J.D., & Wesley, L.P. (1995). Planning and reacting in uncertain and dynamic environments. Journal of Experimental and Theoretical AI. &(1): 197-227.