7-21-2011

When To Pay Attention?: Asynchrony Requires A Trigger

Siminder Kaur

WHEN TO PAY ATTENTION?: ASYNCHRONY REQUIRES A TRIGGER

by

Siminder Kaur

A Thesis

Submitted in Partial Fulfillment of the

Requirements for the Degree of

Master of Science

Major: Computer Science

The University of Memphis

August 2011

Acknowledgments

This thesis would not have been possible without the guidance and the help of several individuals who in one way or another contributed and extended their valuable assistance in the preparation and completion of this study.

First and foremost, my utmost gratitude to my advisor, Dr. Stan Franklin, whose sincerity and encouragement I will never forget. Dr. Franklin has been my mentor as I hurdle all the obstacles in the completion this research work. Also many thanks are extended to my teachers here at the University of Memphis and special thanks to my committee members Dr. King-Ip Lin, and Dr. Vinhthuy Phan, for their guidance and continuous support. I would like to thank the members of the Cognitive Computing Research Group for their friendship and valuable feedback during the course of this study.

And lastly, I would like to extend thanks to my family, and my dear husband Vaneet for encouraging and supporting me throughout the years in graduate school. I especially want to thank Vaneet for his constant support during the final stages of my study, without which it would have been next to impossible to complete my thesis.

ABSTRACT

Kaur, Siminder. M.S. The University of Memphis. August 2011. When to pay attention?: Asynchrony requires a trigger. Major Professor: Dr. Stan Franklin.

LIDA is an integrated, comprehensive control architecture for autonomous agents capable of making complex decisions in dynamic environments. LIDA's computational architecture consists of various modules that each operate asynchronously. However, the Global Workspace (GW) module, responsible for conscious attention, transfers information serially. There can be several computational techniques used to initiate the broadcast of content from the GW. The present study evaluated four techniques called "triggers" to start the GW broadcast under four different conditions. A parameter search was performed to find the optimum trigger parameters, followed by an investigation to identify the trigger(s) which were most useful for an efficient autonomous agent operating in an ALife domain. The Individual Activation Above Threshold (IAAT) trigger performed the best among all four triggers when used individually, whereas, the IAAT and No Broadcast Occurring (NBO) trigger combination outperformed all the other triggers individually, or in combination.

TABLE OF CONTENTS

List of Figures

1 Introduction

An autonomous agent is a system that is embedded in, and is a part of, an environment, that senses its environment and acts on it, over time, in pursuit of its own agenda, so that its actions affect what it senses in the future (Franklin & Graesser, 1997). LIDA (Learning with Intelligent Distribution Agent) is a control architecture for an autonomous agent capable of making complex decisions in a dynamic environment. The LIDA model, primarily based on the Global Workspace Theory (GWT) (Baars, 1988), is an integrated, comprehensive and highly complex model that covers a large segment of human cognition. The underlying LIDA cognitive architecture is both conceptual and computational, and is compatible with various findings in cognitive science and neuroscience.

LIDA is an extension of Intelligent Distribution Agent, an implemented and running software agent that finds new billets for US sailors at the end of their current tour of duty (Franklin, Kelemen, & McCauley, 1998). Every autonomous agent must sense its environment and act on it. This process is continuously repeated in the form of a sense-process-act cycle called the *cognitive cycle*. LIDA theory hypothesizes that humans and autonomous agents go through these complex cognitive cycles involving perception, a number of memory systems, attention and action selection and sample the world at an asynchronous rate of five to ten times a second (Madl, Baars, & Franklin, 2011).

In general, any software architecture that consists of a number of modules can either have synchronous or an asynchronous processing mechanism. Synchronous processing takes place when transmission of data between different modules is governed by some

external clock, whereas asynchronous communication is the transmission of data between two modules that are not synchronized with one another via a clocking mechanism or other techniques. Basically, the sender can transmit data at any time, and the receiver must be ready to accept information as and when it arrives. Architectures having a synchronous processing mechanism are computationally easier to implement and debug. But the major drawback of such a mechanism is that it cannot be effectively used to model human cognition which is mostly asynchronous (Zeki, 1998). Also, in synchronous processing, a receiver cannot start its internal processing until it receives data from the sender, which can hamper the overall efficiency of a system when compared to asynchronous processing. In asynchronous architectures, the modules can operate independently of the other modules and the interaction between modules primarily happens only for passing information. In other words, all the modules of the system work in parallel to each other, and operations in one module can be performed even when data is being transferred into this module by some other module. Asynchronous control has many advantages over a synchronous control mechanism. The main advantage of asynchronous control is that it can be used to model cognition, and an architecture based on this mechanism can be biologically plausible. Further, some autonomous agents cannot possibly operate without asynchrony, e.g., an agent having multiple sensors and different schedules for sampling its environment would need to perceive incoming data as and when it comes. Also, asynchronous data transfer mechanism enhances performance of the system by allowing modules to overlap processing with input/output operations. However, asynchronous data transfer introduces

additional software complexity and hence the implementation of such systems is difficult as compared to its counterpart.

LIDA's computational architecture consists of various modules that operate asynchronously between each other. Every module has its own internal processing and agenda. Modules receive input from other modules but their internal processing is not triggered by that input, rather each module runs continuously at its own specified frequency. However, two modules in LIDA transfer information serially, one of them being Global Workspace (GW) which is responsible for conscious attention in agents. Conscious attention, traditionally viewed as a serial stream, integrates different sources of information, but broadcasts only one content at any given moment. If messages were broadcasted in parallel, they would tend to overwrite one another, making it difficult to understand. Further, broadcasts are made to small processes that cannot handle a lot of information at once, so we have to limit the broadcasts in terms of size as well. GWT offers an explanation for consciousness being serial in nature rather than parallel, which is common in the rest of the nervous system (Franklin et al., 2007). The LIDA model implements this seriality of consciousness in its GW module within the *cognitive cycle*, which is described in detail later. Apart from the attention/consciousness mechanism in LIDA, one more process serial in nature, dealing with the same question of when to transmit some content is the action selection mechanism. An agent can choose only one action at any given time.

There can be several computational techniques to transmit content serially in a system, and the transmission can occur at different *time points* or conditions. *Time points* here do not necessarily mean different points in time but can be based on several other

factors. Content needs to be transferred at such a rate that it can be received and interpreted correctly by the receiver. This can help the agent in taking right decisions at the right time. So, the problem of finding out when to transmit some content in an asynchronous architecture seems to be a general computational problem. In LIDA, the GW module broadcasts the conscious content to all other modules serially. This broadcast can be initiated using any of several computational techniques called "triggers". We consider four triggers that can start the GW broadcast at four perhaps different time points. Since it is yet not clear which triggers are used by brains in their functioning, and if we want to use the LIDA model purely for replicating neurobiological experiments, we may not need all four triggers. However, if we want to use the LIDA model for developing practical agents that operate in artificial environments, we may very well need all four triggers so that the agent can take the right decisions at the right time. There is also a possibility that in order to get better *performance*[1], fewer than four triggers will be required. Thus, the problem of "when to pay attention" or in other words "when to transfer information serially in an asynchronous architecture" becomes really important for any cognitive/software architecture. In order to develop smart robots having human-like intelligence, some other cognitive architectures like ACT-R and SOAR will also need an attention mechanism, and will have to deal with the same problem of triggers. The study of triggers for GW can also give ideas to use in LIDA's action selection mechanism, where we have the same issue of when to choose an action. The present study investigates which trigger(s) will be the most useful for an efficient autonomous agent, and will also find its (their) corresponding internal parameters.

[1] Performance of the agent will be defined later in the chapters.

The remaining chapters are organized as follows: In chapter 2, we explain the LIDA architecture and describe the LIDA cognitive cycle in detail. Chapter 3 describes the LIDA-controlled artificial life (ALife) agent, and its environment that was used to run simulations based on the LIDA computational model. The chapter also describes in detail the implementation of the GW triggers. Chapters 4 and 5 provide details of the study experiments and the results of these experiments along with the discussion of the results. Chapter 6 highlights the conclusions and other relevant findings of the study.

2 LIDA and its Cognitive Cycle

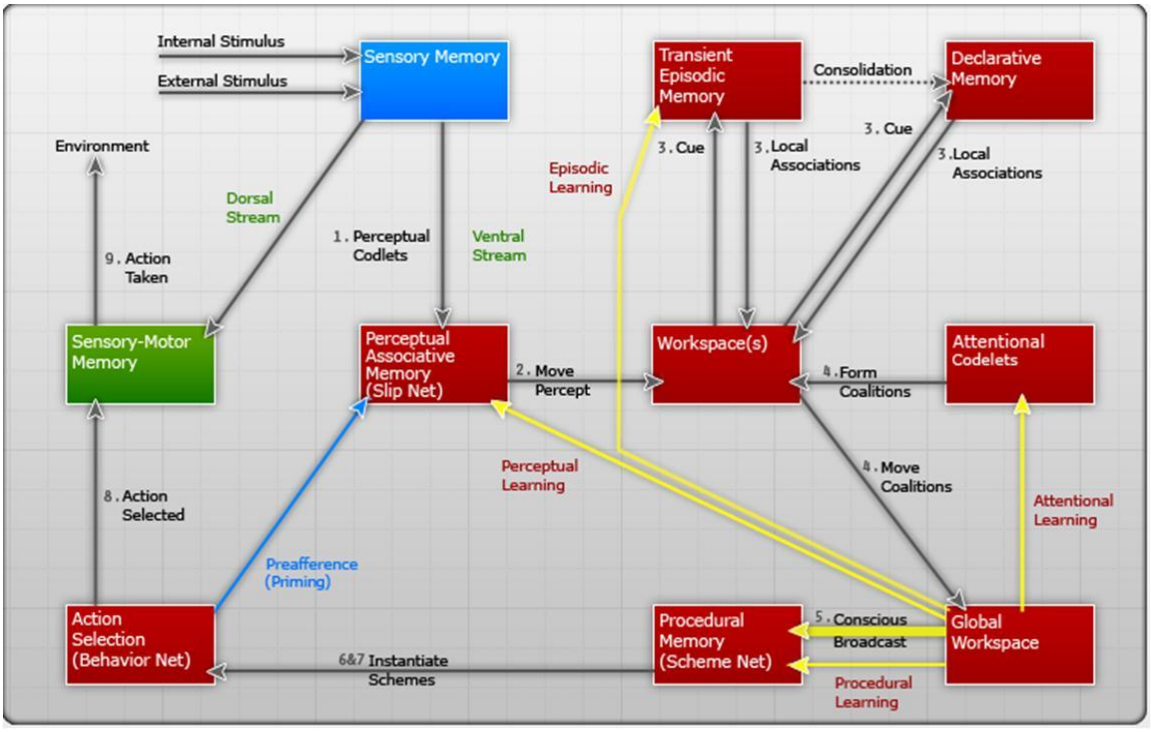

Figure 1. The LIDA architecture and its cognitive cycle

A diagram of the LIDA architecture and its cognitive cycle, derived from the LIDA cognitive model is shown in Figure 1. All autonomous agents, whether humans, animals, or artificial software agents must frequently sample their environment, make sense of it, and perform suitable actions on it. This process is continually repeated over time, and the iterations are called cognitive cycles. LIDA functions by means of flexible, serial but overlapping cycles of activity at the rate of 5 to 10 cycles per second. The term "cognitive cycle" is used because it states the functional roles of various memories, consciousness and decision-making in cognition, according to the global workspace theory (Baars, 1988). The LIDA cognitive cycle can be divided into three phases: the

understanding phase, the attention phase and the action phase. We will now present the LIDA cognitive cycle and its three phases, step by step in detail:

Step 1. The understanding phase of the cognitive cycle starts when a LIDA controlled agent senses its environment, both internal and external, through its various sensors and this raw sensory data is then interpreted by the Perceptual Associative Memory (PAM) where recognition and categorization takes place. Sensory Memory (SM) provides a workspace with decay time in milliseconds for potential interpretations of the sensory data. Specialized processes called feature detectors look into this workspace and find bits of information that may be meaningful and relevant to them, and activate appropriate nodes in PAM. A node can represent a concept, a category, a situation, a feeling, an individual, an event etc. and one node may be connected to another through a link/relation. In LIDA, PAM is implemented as a slipnet which is an activation passing semantic net (Hofstadter & Mitchell, 1995). The interpretations of concepts are assigned in every cognitive cycle, however, interpretations for complex features can accumulate over several cycles.

Step 2. In this step, nodes in PAM whose activation is beyond a certain threshold, form the current percept. The percept, which includes some of the data along with the meanings, is then moved to short-term preconscious Workspace or LIDA's Working Memory. This is done asynchronously i.e. the percept is moved to workspace as and when PAM nodes exceed the threshold.

Step 3. Transient Episodic Memory (TEM) in LIDA is used to store the *what*, *when* and *where* of events. It is a short-term memory with a decay rate measured in hours or perhaps a day. Declarative Memory (DM) is a long-term episodic memory that

contains autobiographical memory of events and also the semantic memory of facts. The events in DM may decay very slowly, if at all. Both these memories help the agent to recall events from its past and use the information in its current decision making. In this step, using the incoming percept and residual content of the Workspace, both episodic memories are cued to retrieve local associations. This information is then combined with the incoming percept and the residual content in Workspace from previous cognitive cycles that hasn't yet decayed, to make sense of the agent's current situation in its world. The Current Situation Model (CSM) is updated in the Workspace. It represents agent's understanding of what is going on at the current moment.

Step 4. Codelets in LIDA are small pieces of code specialized in their individual tasks, with each codelet running independently as a thread. Essentially, they act as daemons that run in the background and keep watching for conditions/situations to occur so that they can act in response. In this step, attention codelets begin the attention phase in LIDA by looking into the Workspace for any relevant, urgent or novel events that can be brought into consciousness. These events can be sensory images, feelings, emotions, memories, desires, goals, intentions, actions, etc. For example, a codelet might try to look for a node that represents "pain" in CSM. An attention codelet has three attributes: 1) Concern representing a situation/condition in CSM which can trigger the codelet to act; 2) Base-level activation representing the codelet's usefulness in bringing some information to consciousness; and 3) Current activation that measures how relevant the attention codelet is to agent's current situation. If an attention codelet finds its concern in the CSM, it creates a *coalition* in the Global Workspace that contains the codelet's *concern* and its related information. Each attention codelet runs as a background process

at a constant frequency. Codelets, on finding their respective *concerns*, create *coalitions* in GW asynchronously. Hence, at each instant of time, there may be multiple *coalitions* being added in GW by different attention codelets.

Step 5. In this step, the agent decides what aspect of its current situation is in most need of its attention. All the coalitions (each trying to bring something to consciousness) in GW compete among each other, and the coalition, which is the most relevant and urgent is selected as the winner of the competition. The winning coalition's contents become the content of consciousness, which is then broadcast globally across all modules. The broadcast, which initiates the action selection phase in LIDA, is hypothesized to be a necessary condition for conscious attention (Baars, 1988). This conscious broadcast serves to recruit other processes which can be used to choose an action to deal with the current situation or learn from it. This competition for attention can be started using any of the several triggers in GW. Conscious broadcast, received by all the other modules of LIDA leads to various kinds of learning. New entities are formed and old entities are reinforced in PAM with the help of Perceptual Learning. Events from the broadcast are used to store new memories in TEM.

Step 6. Procedural Memory (PM) is used to store various procedures/skills that an agent can choose to deal with some situation. These procedural skills are shaped by reinforcement learning. These procedures are called schemes in PM, where each scheme has a context, an action and an expected result. In LIDA, PM is implemented as a simplified Scheme Net (Drescher, 1991) which stores a self-managed collection of action schemes. In step 6, if the conscious broadcast contains content that matches the context of one or more possible action schemes, PM recruits these schemes. A copy of each

recruited scheme is then instantiated by binding its variables and is sent to the Action Selection (AS) module.

Step 7. The instantiated schemes coming from PM constitute behaviors in AS where they compete to get selected for execution. The Action Selection module is implemented as a Behavior Net (Maes, 1989) which chooses a single behavior from the set of competing behaviors. Each behavior has an activation denoting how useful the behavior is in the current situation. Behavior Net may choose an immediate incoming behavior or behavior from a previously active stream. The decision as to which behavior will be selected is influenced by current situation, motivation of the agent (implemented as feelings and emotions in LIDA), the agent's internal/external environment, and by the activations of the various behaviors. The chosen action/behavior is sent to Sensory Motor Memory (SMM) for execution.

Step 8. The incoming behavior is executed in this final step of the cognitive cycle. SMM receives the selected behavior and chooses a suitable algorithm for its execution. As a consequence of the execution, the behavior's underlying behavior codelets perform their specialized tasks, which can result in external or internal consequences. At least one of these behavior codelets is an expectation codelet, which monitors the action execution, bringing to consciousness any information bearing on the expected results. This step completes the cognitive cycle.

3 Design and Implementation

Artificial Life Environment

An Artificial Life (ALife) environment is used for testing the usefulness of various triggers. The ALife environment consists of a LIDA-controlled agent in a 5x5 grid world, wherein its objective is to survive by nourishing itself and avoiding predators and other potential dangers present in the environment. Since ALife is a discrete world, locations are idealized and simplified as single cells on the grid. A cell in this two-dimensional grid can contain any or all of the following objects: the agent, food, predator and a pit (Figure 2). The environment has multiple food items, pits and predators. The agent's goal is to avoid or kill the predators, avoid the pits, and grab food items when available for its survival. The agent can attempt to shoot a predator with the limited number of arrows it has if the predator is in its vision.

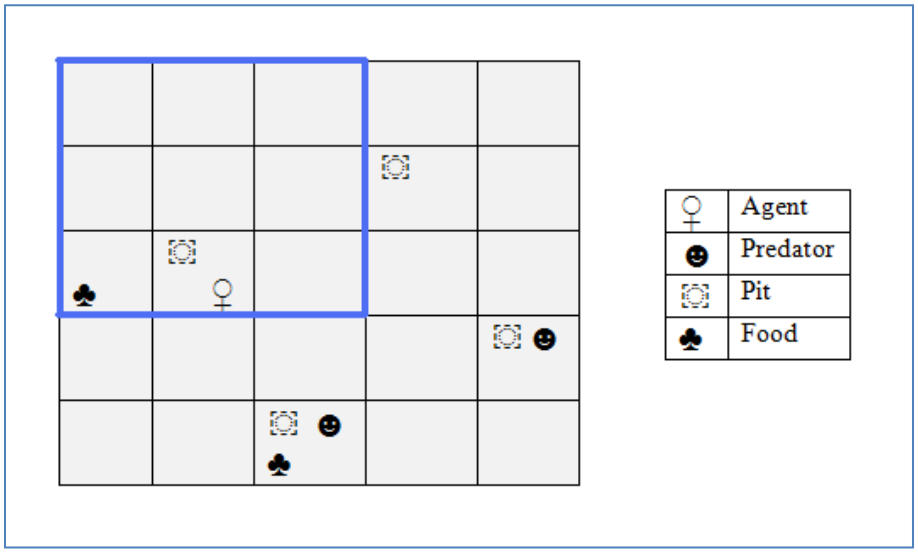

Figure 2. ALife Environment with agent's sense window highlighted

The environment, a 5x5 grid, is not fully visible to the agent. The agent has a 3x3 "sense window" by which it can see what is on its left, right and front, as shown in Figure 2. In this ALife environment, the agent can sense food, predator, pit and the grid boundary walls, if these objects are present in any of the cells in its sense window. Table 1 describes the actions that the agent can take in order to survive in this ALife environment.

Table 1

Actions in the ALife World

Action Name	**Action Description**
GO_FORWARD	Instruct the agent to attempt to move one cell forward
RIGHT_TURN	Instruct the agent to attempt to rotate right
LEFT_TURN	Instruct the agent to attempt to rotate left
GRAB	Instruct the agent to attempt to eat the food that is in the current cell
SHOOT	Instruct the agent to attempt to shoot at the predator in its line of vision
NO_OP	Instruct the agent to attempt to do nothing

A performance measure represents how well an agent behaves in its environment and there can be various ways to measure the performance. We can either use a subjective opinion, where the agent is questioned how satisfied it is with its own performance, or an objective performance measure can be used, which is primarily driven by the objectives of the agent in a particular environment. We considered an objective performance measure because the LIDA agent used in this study was not built in a way by which it can provide a subjective opinion. The agent's performance can be measured objectively using various criterions. One way to measure performance of the agent is to

see how long the agent took to meet its objective. Or, we could measure the number of actions that were taken by the agent to meet its objective, thereby telling us if the agent took irrelevant actions in meeting its objective. An intelligent agent should reach its objective in as few steps as possible. However, the performance of various triggers was compared by measuring the health of the agent, which is a comprehensive performance measure. The agent's health was based on a scoring system which takes into account whether the agent met its objectives in the stipulated amount of time and also considers the number of actions that the agent had to take in order to meet its objective. The scoring system is used to evaluate the agent's health as a result of various events that can happen in ALife environment. Events, for example, can include grabbing food, killing the predator, falling into a pit or being attacked by the predator.

The health scoring system used in this study was adapted from a popular gaming environment Wumpus World's scoring system (Russell & Norvig, 2003). Wumpus World is similar to our ALife environment in which the agent's task is to find hidden gold in a cave with dangerous places and return back to its starting location. The scoring system described in Russell and Norvig was used for the Wumpus World domain and it scored the various events drastically. This scoring mechanism was not suitable for an ALife domain which we used in the present study, hence, we had to adapt the scoring system for this domain. In Wumpus World, the agent loses same amount of health if it enters a pit or if it is eaten by a Wumpus, making pits and the Wumpus equally dangerous in the environment, whereas in ALife environment, pits were considered less dangerous than the predators, so agent's health was decreased by a smaller amount when it walked into a pit. Also, in Wumpus World, the event of agent killing the Wumpus is not scored.

However, in ALife, one of the objectives of the agent is to kill a predator if the predator is in agent's line of sight and the agent has arrows to shoot. This objective is important especially in those cases where the food is blocked by a predator and cannot be reached using any other way except by killing the predator blocking it. So we considered it important to increase the agent's health if the agent meets its objective in the same way it meets the objective of feeding itself.

The present study investigates when an agent should pay attention. In our ALife implementation, the agent takes a NO_OP action if it is not able to decide what it should do next. An intelligent agent will not take a NO_OP action unless it is absolutely necessary. A NO_OP action instructs the agent to do nothing so it will waste time in doing nothing. If the agent does not pay attention to something critical in suitable time, it may suffer. Hence, we considered it important to penalize the agent for taking a NO_OP action. Table 2 presents the comprehensive scoring system that was used in this study.

Table 2

Scoring System Used as the Performance Measure

Event	Health score points
Agent moves to a cell having predator	-50
Agent moves to a cell having a pit	-15
Agent shoots an arrow at the predator	-5
Agent grabs food	+50
Agent kills the predator	+50
Agent takes any action like moving forward or turning left	-1
Agent takes a NO_OP action	-5

Agent Implementation

A LIDA controlled agent is crafted to act in the ALife environment. The objects that exist in this ALife environment are simple and discrete structures represented by different characters in the software implementation. A simple perceptual capability is implemented for the agent where a single node in PAM represents each kind of ALife object. For example there is a node each for "food", "predator", "agent", etc. in PAM. Apart from the object nodes, egocentric location nodes like "front-center", "left", "far-right" represent the location of these objects in the environment with respect to the agent. However, PAM does not contain allocentric representations, i.e., information about the location of one object with respect to other objects in the environment. Feeling nodes like "pleasure" and "pain" linked to the object nodes are used to represent feelings, e.g., "food is pleasurable" and "pit is painful". Primitive feature detectors look for objects like predator, food, pit, wall etc. in the current sense window of the agent along with the location of these objects, and instantiate appropriate nodes in PAM. For example, if a feature detector locates a predator in a cell which is to the left of the agent, it will instantiate nodes "predator", "left" and connect these with a link in PAM. A separate feature detector looks to see if there are any objects in a cell in the immediate front of the agent i.e. whether the front cell is a safe location for the agent to move into. On finding that the front cell is safe, it instantiates an "empty" node linked to "front-center" node in PAM. Percept from PAM then moves to workspace, where the current situation model is updated. Transient Episodic Memory and Declarative Memory are not required and hence not used in this implementation.

The attention phase of the cognitive cycle starts when various attention codelets look for predator, food, pits in the Workspace, and create coalitions in GW on finding their respective *concern*. A default attention codelet tries to bring to consciousness anything that may be important, but with very low activation. Competition in GW is started by any one or a combination of triggers, consequently leading to a broadcast which then instantiates appropriate behavior schemes in PM. A scheme consists of a context, an action and a result. A scheme is activated in the presence of its context. For this domain, we created 25 simple schemes in the form of production rules, e.g., there was a scheme created to grab food if it is in a cell where the agent is. From the instantiated behaviors, the agent then chooses a single best behavior and executes the corresponding action at the operating frequency of the AS module. If the agent is ready to choose an action, but there are no behaviors to choose from, either because the behaviors have decayed or because there were no new behaviors instantiated from the last broadcast, it takes a NO_OP action. A NO_OP action instructs the agent to do nothing and the agent is penalized if it takes a NO_OP action. It is yet not clear if humans take a NO_OP action, but for small animals, such an action may very well be biologically plausible. Also, for artificial agents, a NO_OP action is a default action which sometimes can be computationally necessary. The chosen action is then executed by agent's actuators and the agent's health score is updated appropriately.

In every trial, the ALife environment is configured by placing various objects in it randomly one after the other including the agent. Using random configuration of environment ensures that the environment is not biased for any of the triggers to be evaluated. Each trial begins with the agent having some initial health (200) and the agent

is allowed to act until its health drops extremely low (-300), or the trial exceeds the maximum amount of time specified (1500 time steps in the simulation), whichever comes the first. When either of these conditions is met in any run, the run is ended and the next run is started with another random configuration of the environment. Figure 3 shows the graphical user interface (GUI) for the LIDA controlled agent in ALife environment.

Global Workspace Triggers

A trigger is a process that initiates competition between various coalitions competing for consciousness in the Global Workspace. The coalition having the highest activation wins the competition and its content is broadcast to all other modules of LIDA. This broadcast is done serially i.e. only one content is transmitted at any particular instant in time. We studied four triggers that can start this competition in GW. Before we discuss each of them in detail, we provide some additional details about computational implementation of some key concepts.

Coalition. A coalition in GW is a data structure, called a *node structure* that contains nodes and links. The nodes represent entities and the links represent relationships between those entities. Each node and each link has an activation which specifies how relevant/important it is in the current situation. Activation of a *node structure* is the sum of activations of all its nodes and links.

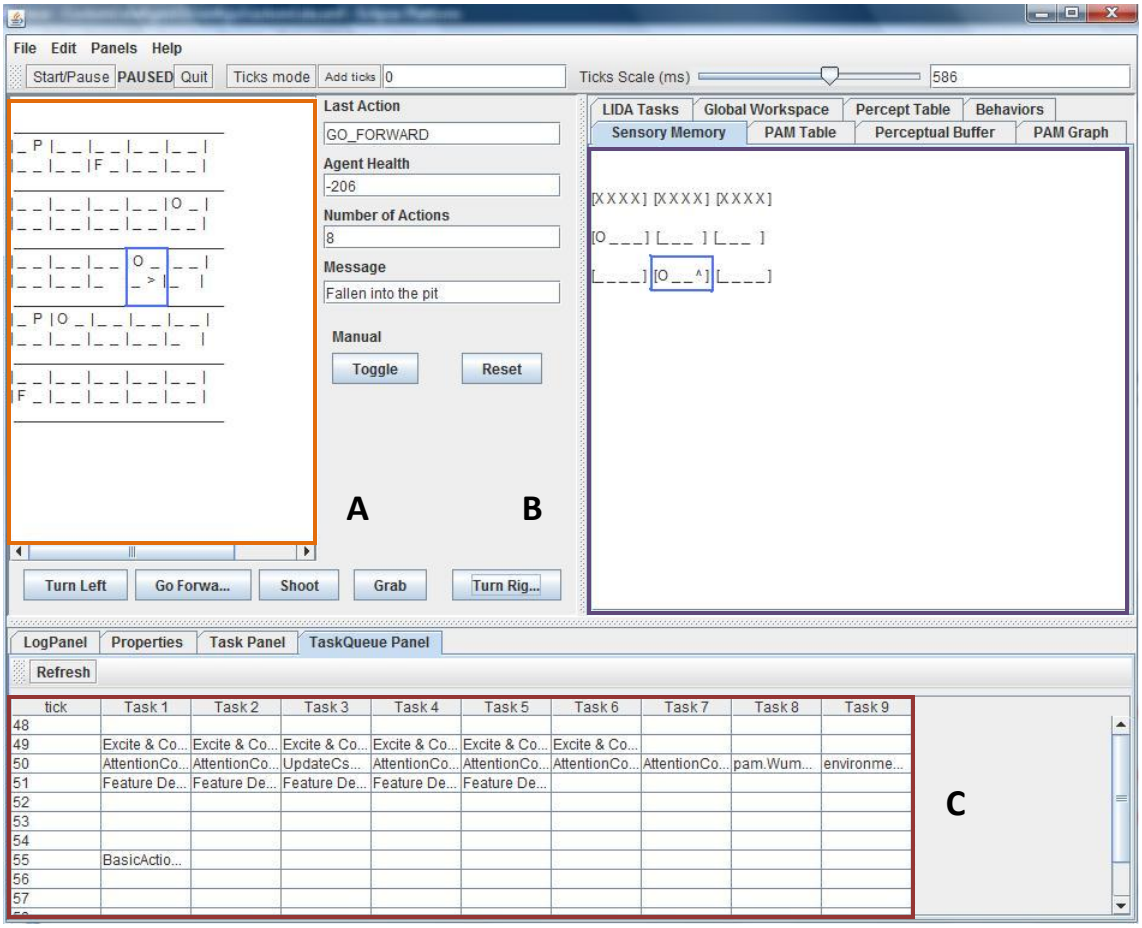

Figure 3. A screen shot of the GUI containing LIDA controlled agent in ALife. A) The top left panel shows the ALife environment grid where each cell (cell containing the agent is highlighted) has four locations for the four kinds of objects it may contain. Character "F" represents food, "O" represents a pit, "P" represents a predator and the agent represented by ">" is facing east. B) The top right panel shows the agent's 3x3 current sense window, where locations are shown with respect to the agent facing north and character "X" represents wall or out of boundary. C) The bottom panel shows the task queue where multiple tasks are scheduled to run at different time points (ticks) in the simulation. Some other information including agent's health and the last action taken by the agent are also displayed. Apart from the panels displayed here, the GUI contains other panels which can be refreshed to monitor the information related to different modules in LIDA.

Each coalition arriving in the GW has an activation which depends on the content of the coalition, i.e., the activation of the *node structure* forming the coalition, and also

on the activation of the attention codelet which creates that coalition. The activation of a coalition is computed as follows:

$$Coalition\ Activation = \frac{(Attention\ Codelet\ Activation\ \times Node\ Structure\ Activation)}{(Number\ of\ Nodes + Number\ of\ Links)}$$

where the activation of a node structure is calculated as,

$$Node\ Structure\ Activation = \sum_{\forall i} Node_i\ Activation + \sum_{\forall j} Link_j\ Activation$$

The coalitions in GW decay at a linear rate until a coalition's activation is reduced to a lower threshold, at which point it is removed from the GW.

Tick. LIDA software framework is a generic, customizable, computational implementation of the conceptual LIDA model. The framework uses multithreading so that various small processes can run in simulated parallel to each other, in the same way that biological minds work. These processes are called LIDA tasks, and a LIDA task manager is responsible for scheduling and executing these tasks. The tasks are stored in a task queue whose each position is a discrete instant of time called a *tick*. Multiple tasks can be scheduled at the same tick. A *tick* can be of any time duration and the duration can be configured, e.g., a *tick* can have duration of 35 milliseconds. In practice, *tick duration* affects the speed of tasks in the simulation. Thus, a *tick* can be seen as the smallest discrete unit of time representing a simulated millisecond in the LIDA framework. Before starting any new competition in GW, we make sure that a broadcast is not being done at the current *tick* so that the seriality of broadcast is maintained.

Let us now explore the GW triggers that can be divided into two categories: a) threshold-based triggers and b) time-based triggers. The four Global Workspace triggers that we studied are:

1. Individual Activation Above Threshold Trigger (IAAT) –This trigger starts the competition in GW, if an incoming coalition has its activation above a certain *threshold*.

This trigger is useful in those moments when the agent perceives something really important in its environment, such as a threat to life or an excellent opportunity to satisfy its hunger. The agent can then pay attention to this state by bringing it to consciousness and taking an appropriate action almost immediately.

2. Aggregate Activation Above Threshold Trigger (AAAT) – At any given moment, there can be more than one coalition inside GW, each of them having some activation. On receiving a new coalition, we check if the aggregate of all the coalitions in GW is above a certain *threshold*, and if it is, this trigger starts the competition in GW.

This trigger is useful when the agent perceives a lot of things in its environment, but none of them are greatly important when considered alone. For example, the agent may perceive danger at some distance ahead, but at the same time it may also perceive food at some other distance. Since each of these perceptions may not be as important as an immediate threat, it will prevent IAAT from triggering a broadcast, so we need something other than IAAT in such a case. This trigger will be applicable in this scenario and the agent can choose an appropriate action if there is enough total activation in GW.

The IAAT and AAAT triggers are applicable when GW receives content that is highly important or when GW contains contents that are important when combined.

However, they will not trigger a broadcast unless there is a new coalition with enough activation, or enough coalitions with some activation. Yet, there can be situations where the agent does not perceive anything important or when it perceives few less important things. In these situations, if we just use threshold-based triggers, we will have an extended unconscious period that would last until one or more coalitions exceed the threshold. Without a broadcast, action schemes will not be instantiated and the agent may end up doing nothing, even though there may be potentially useful coalitions in GW during this period. So we need triggers that can allow conscious processing of less important information in cases when there is no current novel or significantly important content in GW. The time-based triggers play a useful role in such scenarios.

3. No New Coalition Arriving trigger (NNCA) – The NNCA trigger will start the competition if no new coalition has arrived in GW since *delay* ticks have passed.

This trigger is useful when the agent has not perceived anything new in its environment in some time, so no new coalition has been added in GW. However, there may be coalitions in GW which haven't decayed from the previous cognitive cycles. The IAAT and AAAT triggers will not trigger a broadcast unless there is a new coalition in GW. But a broadcast must be done so that appropriate schemes can be instantiated and the agent can take an appropriate action, instead of doing nothing, hence this trigger becomes useful.

4. No Broadcast Occurring trigger (NBO) – This trigger will fire if *delay* ticks have passed since the last GW broadcast. This trigger is useful when there's very little going on in the agent's environment and the agent is not perceiving anything that may be really important or urgent to it. For example, the agent may come into a situation where it does

not perceive anything really important, like food or danger, in its environment. However, it may still perceive things, which are less important for its survival, but which may increase the chances of the agent to locate something important. For example, the agent may perceive a safe location in its front where it can move and look some more for food. Since threshold-based triggers will not be applicable in such situations, the agent may end up having an extended unconscious period. Hence, this trigger can be useful in bringing to consciousness, content that may be currently less important but which can increase the chances of locating something important in the future.

4 Experiments

Performance of various Global Workspace triggers was studied by running several simulations using an ALife environment and a LIDA controlled ALife agent. In each simulation, various objects were placed randomly in the environment using a random number generator, which made some environments easy and some difficult for the agent's health and survival. Each simulation was performed using the same series of 50 randomly generated environments so that there would be no bias from the environment. The agent's health score was recorded for each trial in the simulation.

The experiments were divided into two main categories. First, we carried out a parameter search, where experiments were performed to find the optimum values for each of the trigger parameters. The second set of experiments was done to compare the performance of various triggers, alone and in combination. However, before running any of these experiments, preliminary experiments were done to adjust some of the other internal parameters in the LIDA framework.

Preliminary Experiments

LIDA contains a number of internal parameters which need optimization for computational implementation of an agent in a specific domain. Once optimized, these parameters can be made control variables with fixed values when performing different experiments that vary one or more dependent variables. Pilot studies were done to adjust and optimize these internal parameters for the agent in a selected specific environment, before performing the actual experiments for this study.

Each module in the LIDA framework is responsible for a particular task and each task in the framework is executed periodically after a specified time. The specified time after which the task is executed is governed by the task's *ticks* parameter, e.g., a task can run every five *ticks* (e.g., simulated 50 milliseconds). The tasks run asynchronously in simulated parallel, just like local neural circuits run in parallel in human brains. These 'ticks' parameters for each of the module need to be adjusted so that the resulting LIDA cognitive cycle becomes comparable to the conceptual LIDA cognitive cycle in terms of the timing, and thus neuroscientifically plausible. Research work done by Madl et al. (Madl et al., 2011) on timing of the LIDA cognitive cycle provided an excellent ground for some of these parameters. Table 3 lists the most important timing parameters that were used in this study.

Table 3

Timing Parameters for the ALife Agent

Parameter	**Value (in ticks[1])**	**Value (in ms)**
Sensory Memory ticks	2	20
Feature Detectors ticks	2/3	20/30
Attention Codelets ticks	3	30
Attention Codelet refractory period	5	50
Global Workspace refractory period	5	50
Action Selection ticks	11	110

An important hypothesis of the LIDA model is the discreteness of consciousness, which means that humans have a single conscious content at any given time and two consecutive conscious contents are separated with a short period of unconsciousness.

[1] In this study one tick was equal to simulated ten milliseconds.

Franklin et al. quote, “conscious events occur as a sequence of discrete, coherent episodes separated by quite short periods of no conscious content” (Franklin, Baars, Ramamurthy, & Ventura, 2005, p. 4). This hypothesis along with results from some of the pilot experiments led us to implement a minimum *refractory period* for consciousness, which simply means that a new broadcast cannot take place for some minimum specified time, after a broadcast has happened. In human physiology, where such periods are ubiquitous, a *refractory period* is a period of time during which an organ or cell is incapable of repeating a particular action. According to LIDA, each of the three phases of the cognitive cycle consume roughly 100 ms, while the overlap between the understanding and the attention phase is around 50 ms. We used a refractory period in *ticks* that corresponds to 50 ms in time because it is hypothesized to be the minimum time needed for a conscious broadcast.

Trigger Experiments

Simulations were performed using each trigger individually and systematically varying its parameter in order to find the optimum value. The threshold for the IAAT trigger was varied from 0.1 to 1.0 in equal increments of 0.1, and at each of these threshold values, 50 trials were performed (using 50 randomly chosen environment configurations). The mean score from 50 trials was plotted for each threshold value to find the optimum threshold value. A coalition has its activation between 0.0 and 1.0, hence we used boundary limits of 0.0 and 1.0 for this parameter search.

In a similar way, the threshold for the AAAT trigger was varied from 0.1 to 1.2, and the mean score for each threshold value was plotted. The threshold for AAAT trigger can be as low as 0.0 and can have any upper limit (depends on the number of coalitions in

GW). We performed a parameter search for this threshold in the range of 0.0 to 1.2 because we observed a decreasing trend after threshold of 0.7.

Delay parameters for the NBOT and NNCAT triggers were varied systematically from 3 to 25 in equal increments of 2, and mean scores from 50 trials for each of the parameter values were plotted. The lower limit of delay parameters was chosen as 3 because the attention codelets start at 30 ms (3 ticks). The upper limit of delay parameters was chosen as 25 because the attention phase on an average ends at 200 ms (20 ticks).

After fixing the optimum value for each of the trigger parameters, we compared the triggers individually, and later, all possible combinations of triggers. Experiments were performed using the triggers individually and then as pairs. Finally, experiments using various combinations of three or more triggers were performed. Table 4 shows the list of experiments that were performed.

Table 4

List of Experiments for Comparing the Triggers

Experiment Name	Description
IAAT	LIDA using Individual Activation Above Threshold trigger only
AAAT	LIDA using Aggregate Activation Above Threshold trigger only
NBO	LIDA using No Broadcast Occurring trigger only
NNCA	LIDA using No New Coalition Arriving trigger only
IAAT + AAAT	LIDA using both IAAT and AAAT triggers
IAAT + NBO	LIDA using both IAAT and NBO triggers
IAAT + NNCA	LIDA using both IAAT and NNCA triggers
AAAT + NBO	LIDA using both AAAT and NBO triggers
AAAT + NNCA	LIDA using both AAAT and NNCA triggers
NBO + NNCA	LIDA using both NBO and NNCA triggers
IAAT + AAAT + NBO	LIDA using IAAT, AAAT and NBO triggers
IAAT + AAAT + NNCA	LIDA using IAAT, AAAT and NNCA triggers
IAAT + NBO + NNCA	LIDA using IAAT, NBO and NNCA triggers
AAAT + NBO + NNCA	LIDA using AAAT, NBO and NNCA triggers
IAAT + AAAT + NBO + NNCA	LIDA using all four triggers

5 Results and Discussion

Several experiments were performed using the LIDA controlled agent in an ALife environment, and performance of various Global Workspace triggers was evaluated. We first performed a parameter search for each of the trigger parameters by plotting the mean scores for systematically increased parameter values. The second set of experiments compared the triggers individually and in combination with each other.

Part A: Search for trigger parameters

The ALife agent performed the best using the IAAT trigger with a threshold of 0.4, as seen in Figure 4. We observed that the agent did not take any real actions when the threshold value was greater than or equal to 0.8. This happened because all coalitions had activation value less than 0.8 and the IAAT trigger failed to start the GW competition above a threshold of 0.8. When threshold was kept too low (< 0.3), the GW broadcast occurred too frequently, making the agent take injudicious actions, thus decreasing its health score.

The optimum value for the AAAT trigger threshold was 0.7 as seen in Figure 5. A decreasing trend was observed for the agent's health score after threshold of 0.7. This could be because a higher threshold made the GW broadcast less frequent, and as a result, more NO_OP actions were taken by the agent, affecting its health. Also, if the agent takes more NO_OP actions it reduces its chances of getting food, since the agent can take limited number of actions in a trial.

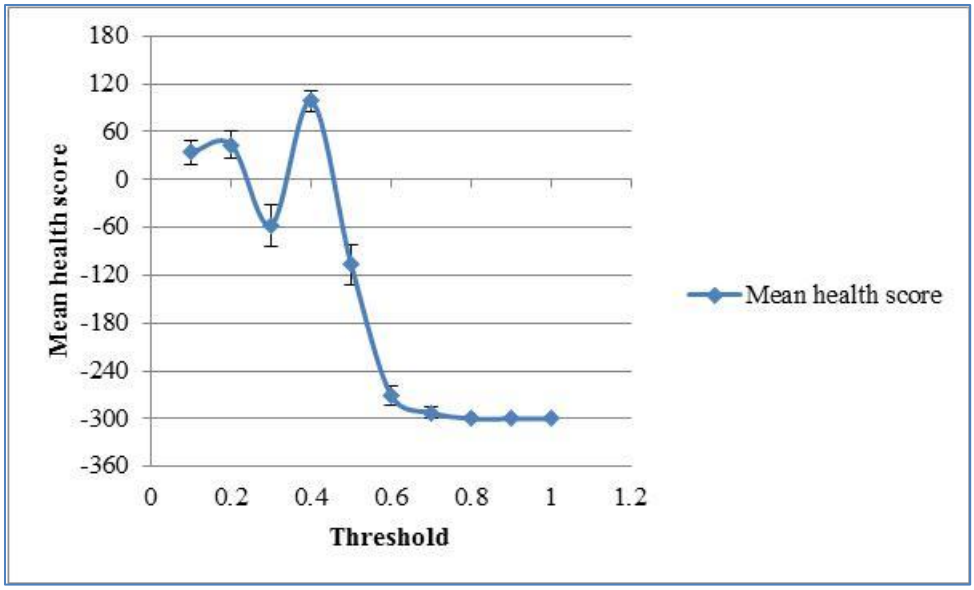

Figure 4. Agent's health score at different threshold values for the IAAT trigger. Error bars indicate the standard error of the mean.

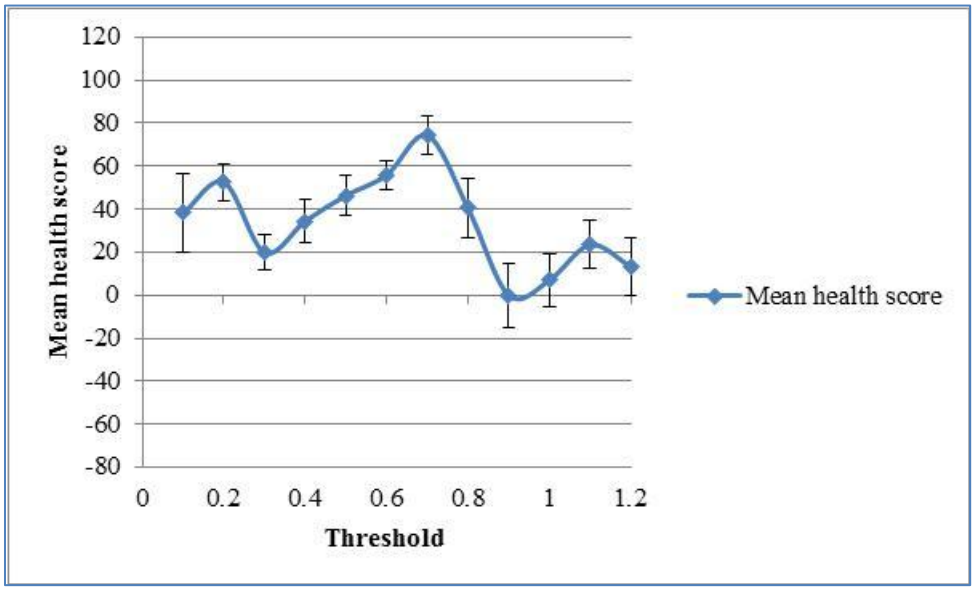

Figure 5. Agent's health score at different threshold values for the AAAT trigger. Error bars indicate the standard error of the mean.

The optimum delay parameter for the NBO trigger came out to be 5 ticks, which is equivalent to 50 milliseconds as shown by Figure 6. The use of this trigger alone,

results in GW broadcast at regular intervals, which is determined by the delay parameter. Hence it is important that the broadcast happens almost as early as possible so that the agent can pay attention to the critical conditions in its environment. A longer delay value could result in more NO_OP actions by the agent, decreasing its health. This can be seen by the low performance of the agent using a high delay value for this trigger. The experiments using the NNCA trigger showed that the optimum value for its decay parameter is 3 ticks (Figure 7).

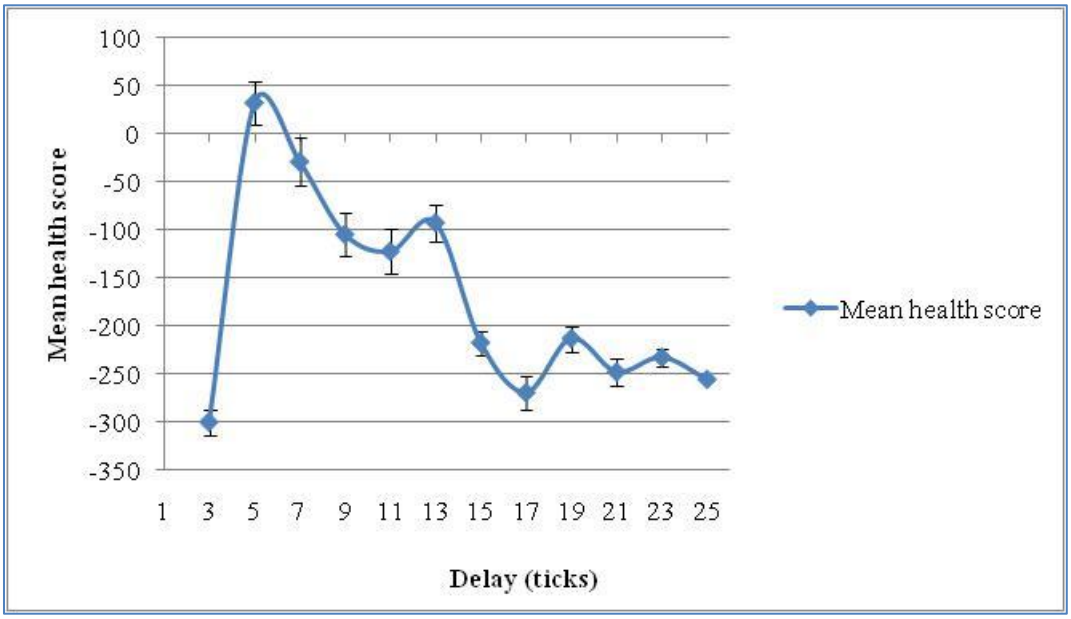

Figure 6. Agent's health score at different delay values for the NBO trigger. Error bars indicate the standard error of the mean.

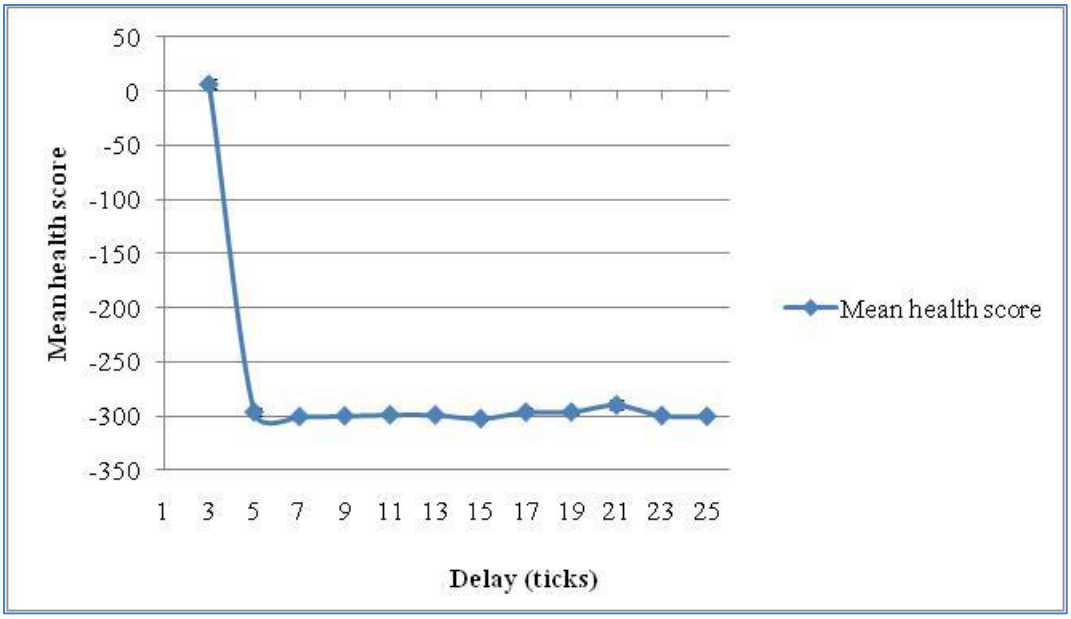

Figure 7. Agent's health score at different delay values for the NNCA trigger. Error bars indicate the standard error of the mean.

Part B: Comparison of triggers

The current technology does not yet have sufficient resolution to explore many of the fine-grained mechanisms that occur in human brains. For example, electroencephalography (EEG) has great temporal resolution, but poor spatial resolution (Nunez & Srinivasan, 2005). Imaging technologies like PET and fMRI, though can provide us with good spatial resolution but only poor temporal resolution. Using individual electrodes, we can get good temporal and special resolution, however they provide limited scope. Hence, currently there is no data/evidence identifying the triggers used by the brains. However, Dr. Baars' latest hypothesis (B. Baars, personal communication, 2011) on consciousness, suggests a dynamic GW (dGW), as opposed to a static GW in the brains. According to him, Global Workspace can be located in any of a large number of possible cell assemblies in brains, and this location of GW may

dynamically change. There can be multiple coalitions at different locations of the brain at any given time. According to this hypothesis, a broadcast can occur when any one of these coalitions achieves enough activation to trigger an ignitions (broadcast). This could mean that the Individual Coalition Above Activation trigger (IAAT) could be used by brains and is, indeed, biologically plausible. He also suggests that other triggers could be used indirectly in brains in such a way that they influence the IAAT trigger. For example, the No Broadcast Occurring (NBO) trigger can lower the threshold of the IAAT trigger if no broadcast has occurred since some amount of time. The IAAT trigger can then start the competition if a coalition's activation meets this new lower threshold.

A comparison of performance of all the triggers when used individually is shown in Figure 8. Comparing all four triggers individually, we observed that the IAAT trigger performed the best with the highest mean health score.

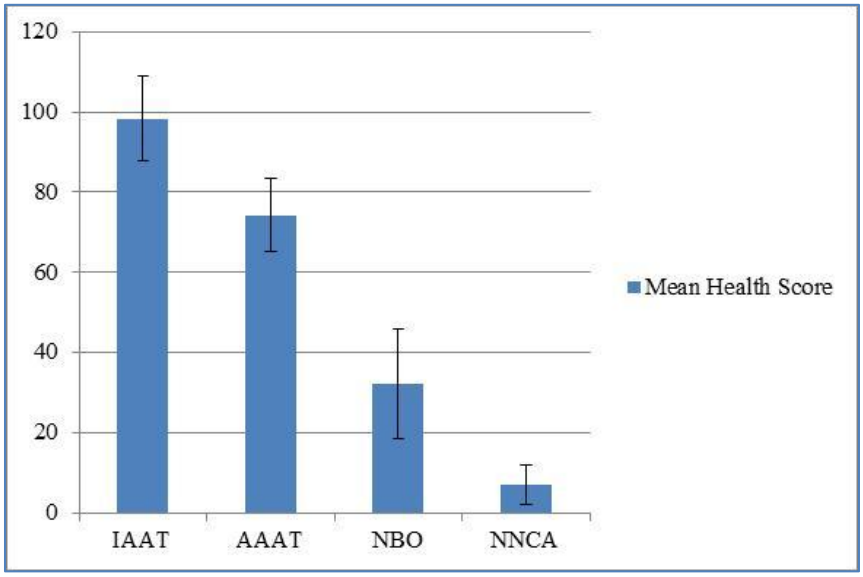

Figure 8. Comparison of the performance of various triggers when used individually

Interestingly, this result aligns very well with the dynamic GW hypothesis stated above. The next best performance was of the AAAT trigger followed by the NBO trigger. It was observed that the agent did not perform well using NNCA trigger alone, when compared to the other triggers. In LIDA model, the NNCA trigger starts the competition if no new coalition is created since the last broadcast. However, in the experiments it was noticed that new coalitions were created by the attention codelets, before the task for this trigger was scheduled to run. As a result, the new coalitions prevented this trigger from starting a broadcast, and the agent took more NO_OP actions lowering its health. Figure 9 shows a comparison of the performance of the IAAT and NBO triggers, used individually, and in combination with each other. When the IAAT trigger was combined with the NBO trigger, the agent chose more relevant actions, and more importantly, in appropriate time, which resulted in its better performance.

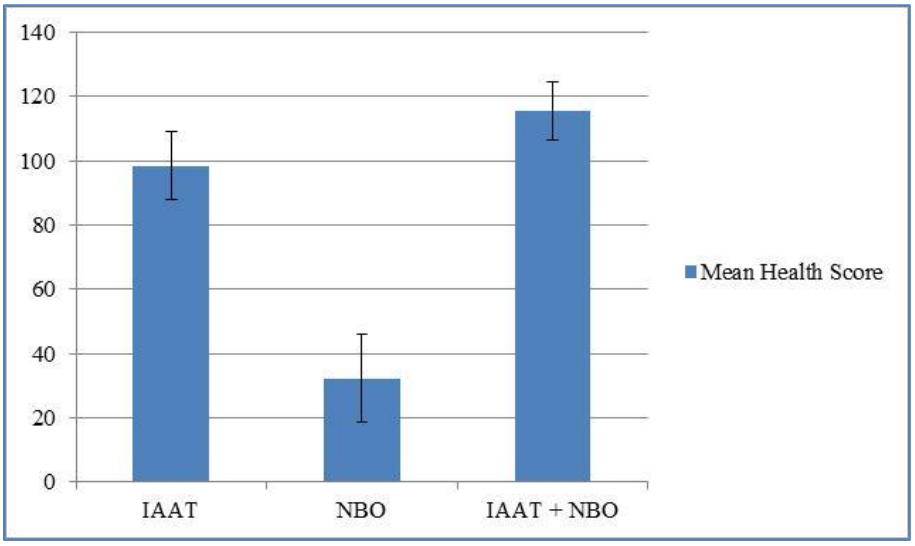

Figure 9. Comparison of the performance of IAAT and NBO when combined

The IAAT and NBO trigger combination performed the best among all trigger combinations involving two triggers (Figure 10). The NNCA trigger when combined with other triggers lowered the performance of the agent as the broadcast happened too frequently, forcing the agent to choose an action indecisively. For example, in the IAAT and NNCA trigger combination, it was seen that the NNCA trigger started the GW broadcast even before a coalition demanding immediate action entered GW, and this made the agent take an injudicious decision, resulting in its poor performance.

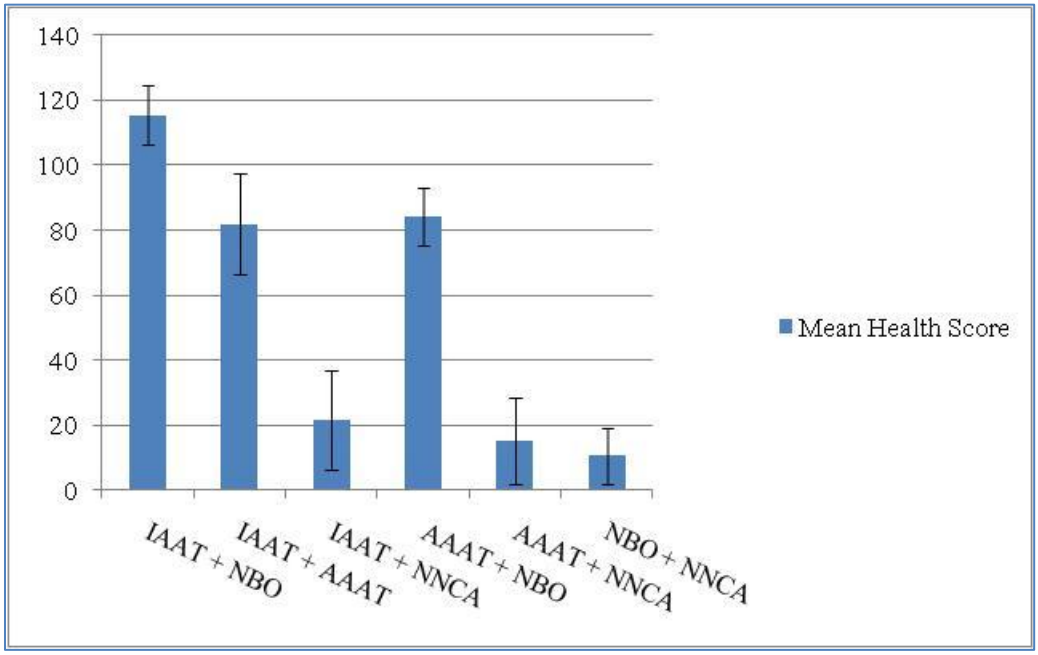

Figure 10. Comparison of the performance using two triggers

Figure 11 shows the comparison of arrangements using three or more triggers. The agent performed the best with a combination of the IAAT, NBO and NNCA triggers. Overall, it was seen that the combination of IAAT and NBO trigger outperformed all other combinations and thus was most suitable for the LIDA-controlled ALife agent.

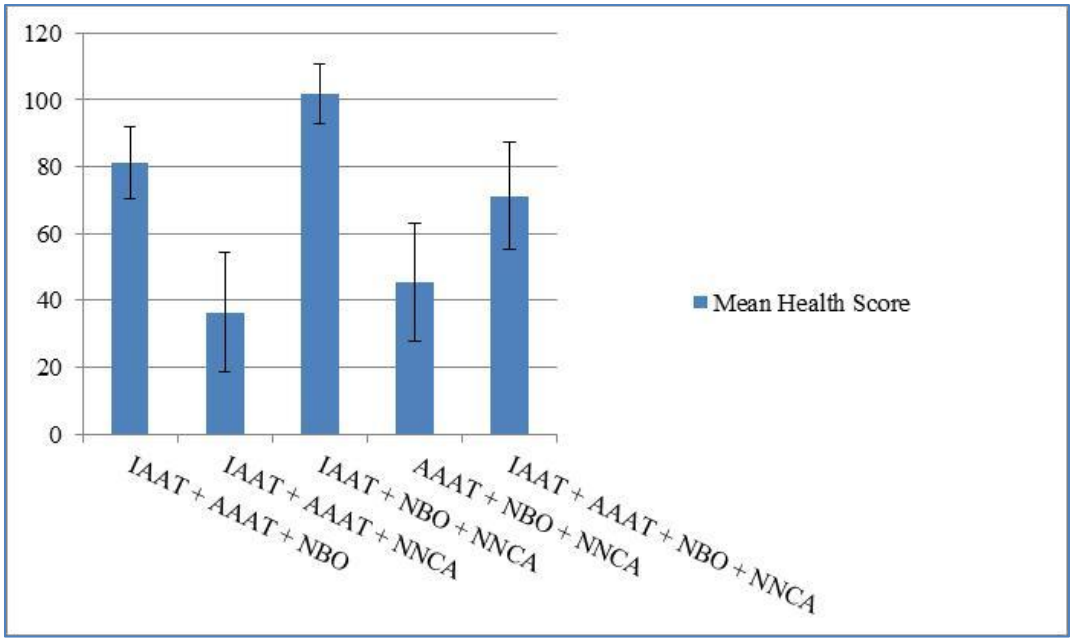

Figure 11. Comparison of the performance using three or more triggers

It is important to note here that the parameter search performed to find the optimum parameter values for each trigger was only done by using the triggers individually. The parameter search, if performed for the combinations of the triggers, may yield different optimum values affecting the trigger performances when evaluated in combination.

The implemented scoring system along with the design of the ALife domain was found to influence the performance of various triggers. The triggers which performed the best in an ALife domain may perform differently in other domains. The limitations of this study included a small variation in the agent's performance when an experiment was repeated using the same parameters and environment. This variation was introduced due to the multithreading in Java. When a number of tasks are scheduled to run at a particular instant of time (tick), the order of execution of these tasks depends on the multithreading

and also on the underlying hardware where the simulation is being run. This order of execution may affect the agent's behavior in an environment, thus introducing a small variability. This variability can be compared to the noise in brains. Brains also have noise, which is random brain activity not important to the mental functions of the brain.

6 Conclusion

The present study evaluated the usefulness of various Global Workspace triggers within a comprehensive, asynchronous model of cognition – LIDA. A LIDA-based ALife agent was built to test the performance of various triggers, which define when an agent should pay attention. An ALife domain similar to the Wumpus World domain was used to study the triggers individually and in combination with each other. A parameter search was also done to find the optimum value for each of the trigger parameters. The IAAT trigger was found to be the best trigger among all four triggers when used individually. The IAAT trigger when combined with the NBO trigger was observed to be the best trigger combination among all the trigger combinations evaluated. Also, the IAAT and NBO trigger combination which had the highest health score for the agent, outperformed all the other triggers individually, or in combination. It was observed that the performance of the triggers was dependent on the scoring system and the domain used in this study. The limitations of this study include a small variation in the agent's health score introduced due to the multithreading in Java.

The findings of the present study contribute to the conceptual LIDA model in several ways. Firstly, the study highlights the importance of a *refractory period,* thus, recommending its introduction, if possible in all the modules, but especially in the GW module. Secondly, it could be inferred from the study that with the asynchronous architecture of the computational LIDA model, it is possible that an agent can take more than one action during a single cognitive cycle. The study also suggests the best trigger arrangement (a combination of two triggers in this case) for the optimum performance of an ALife agent operating in a real environment. Lastly, this study provides important

baseline parameters which could be used by the future researchers studying comparable triggering mechanisms for LIDA's Action Selection module.

References

Baars, B. (1988). *A cognitive theory of consciousness*: Cambridge: Cambridge University Press.

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

Franklin, S., Baars, B., Ramamurthy, U., & Ventura, M. (2005). The Role of Consciousness in Memory. *Brains, Minds and Media Vol. 1*, 1-38.

Franklin, S., & Graesser, A. (1997). Is it an Agent, or Just a Program?: A Taxonomy for Autonomous Agents. *Proceedings of the Workshop on Intelligent Agents III, Agent Theories, Architectures, and Languages* 21-35.

Franklin, S., Kelemen, A., & McCauley, L. (1998). *IDA: A Cognitive Agent Architecture*. Paper presented at the IEEE Conf on Systems, Man and Cybernetics.

Franklin, S., Ramamurthy, U., D'Mello, S., McCauley, L., Negatu, A., Silva, R., et al. (2007). LIDA: A Computational Model of Global Workspace Theory and Developmental Learning.

Hofstadter, D., & Mitchell, M. (1995). The Copycat project: a model of mental fluidity and analogy-making *Fluid concepts and creative analogies* (pp. 205-267): Basic Books, Inc., New York, USA.

Madl, T., Baars, B. J., & Franklin, S. (2011). The Timing of the Cognitive Cycle. *PLoS ONE, 6*(4), e14803.

Maes, P. (1989). How to do the Right Thing. *Connection Science, 1*(3), 291 - 323.

Nunez, P. L., & Srinivasan, R. (2005). *Electric Fields of the Brain: The Neurophysics of EEG* (2nd ed.): Oxford University Press, USA.

Russell, S., & Norvig, P. (2003). *Artificial Intelligence: A Modern Approach*: Prentice Hall.

Zeki, S. (1998). Parallel Processing, Asynchronous Perception, and a Distributed System of Consciousness in Vision. *Neuroscientist, 4*(5), 365-372.