An Attentional Control Mechanism for Reasoning and Learning

Peter Isaev(✉) and Patrick Hammer(✉)

Department of Computer and Information Sciences, College of Science and Technology, Temple University, Philadelphia, PA 19122, USA
{peter.isaev,patrick.hammer}@temple.edu

Abstract. This paper discuses attentional control mechanism of several systems in context of Artificial General Intelligence. Attentional control mechanism of *OpenNARS*, an implementation of Non-Axiomatic Reasoning System for research purposes is being introduced with description of the related functions and demonstration examples. Paper also implicitly compares *OpenNARS* attentional mechanism with the one found in other Artificial General Intelligence systems.

Keywords: Non-axiomatic reasoning · Artificial general intelligence · General machine intelligence · Inference control · Attention mechanism

1 Introduction

For the last decades the field of AGI research has presented numerous systems that aim to rival human level intelligence. In high level of abstraction, some of these systems share similar design principle, in particular they consist of logic and control parts. Being conceptually complex, logic usually is created separately from control mechanisms and often exhibits grounds for achieving anticipated level of intelligence. Later logic is being implemented within a control mechanism of the system, which often becomes massively complex and includes multiple sub-components. Given that AGI system should operate under Assumption of Insufficient Knowledge and Resources (AIKR) [3], resources are always in high demand forcing the system to make a choice for the next reasoning step during the real-time processing. Therefore the correct choice is critical for system performance and efficiency. Picking up the "correct" task or relevant next reasoning step is the main function of Attentional Mechanism that steers behavior of the system in a desired way and allows productive learning.

In systems based on Non-Axiomatic Logic [3], like NARS, role of Attentional mechanism in general is to decide which premises should be selected for inference in real-time during current system's cycle. For NARS, operating under AIKR in the real-time, a new task can arrive at any given moment requiring NARS to work under finite resource constraints and be always open for new tasks. Clearly the choice of "relevant" premises during inference process will influence future

B. Goertzel et al. (Eds.): AGI 2020, LNAI 12177, pp. 221–230, 2020.
https://doi.org/10.1007/978-3-030-52152-3_23

learning vector, final resolution of a supplied task and overall performance and efficiency. Based on above considerations, the attentional control aspect of the *OpenNARS* [2] was thoroughly inspected and revised. While the information present in this paper is mostly related to NARS, we provide some overview and insights of other AGI systems' attentional mechanisms. The next section provides overview of attentional aspects of several related AGI systems, then OpenNARS attentional control and its functionality is being discussed, in Sect. 5 we demonstrate attentional mechanism in action and finally we try to compare attentional aspects between OpenNARS and other AGI systems.

2 Related Works

One of the earlier systems which share resource allocation and attentional mechanism ideas with NARS is the Copycat project [9]. While Copycat can be applied to very narrow domain it features sophisticated attentional and resource allocation mechanisms that incorporate ideas similar to ones found in more complex AGI systems. Copycat is a computer system that tries to discover and build analogies in psychological realistic way, its main idea is to allow mental fluidity achieved by concept slippage which results from building pressure during real-time processing [9]. Copycat control system has three main sub-components: Slipnet, Workspace and Coderack. Slipnet is system's main memory and can be thought as a long-term memory represented by graph where concepts are nodes connected by edges as numerical distances between them. Concepts are used in building bigger structures in Workspace, and Coderack is a place where *codelets*, small working agents, are being created and chosen to complete the work in Workspace. Copycat uses Parallel Terraced Scan [4] as a resource allocation mechanism in order to proceed and discover promising structures in Workspace to be further developed. Function of attentional control is to decide which structure to pick from the Workspace and which *codelets* should be allowed to work. Each structure in Workspace is assigned with *salience*, a dynamic quantity, that determines the probability of acquiring attention from *codelets*. A *codelet*, when created, is placed in a pool with *urgency* value that determines probability of being selected as the next *codelet* to run. *Urgency* estimates an importance of *codelet*'s action which in turn reflects the current state of the system. *Urgency* of a *codelet* is not its priority but rather a relative speed at which the pressures represented by this *codelet* should be attended to. Calculation of *urgency* and *salience* depends on many system's factors and impacts overall dynamics and performance.

Many ideas of Copycat have been employed in LIDA (Learning Intelligent Distribution Agent), the system that models human consciousness [8]. Logical part of LIDA enforces Global Workspace Theory (GWT) (Baars 1988) and implements cognition process in a serial way through use of system cycles. LIDA's control mechanism is immensely complex, its architecture is both symbolic and connectionist, it incorporates several modules with independent architectures and features four types of memory (Perceptual Memory, Procedural Memory,

Episodic Memory, Local Workspace), each with different connectionist architecture. Most task within the system are carried out by *codelets* (small agents) that represent small processes in GWT. System cycle models human cognition and mainly consists of three phases: sense, attend and action selection [5]. Attention phase is implemented through use of attention *codelets* where each looks for "interesting" situation and attempts to bring it to "consciousness" which is modeled by Global Workspace module. Attention *codelets* similarly to daemons look into Workspace and Episodic memory, form coalition of "appropriate" data and bring it into the Global Workspace. A coalition may be viewed as collection of functionally related data. In general, attention is implemented as a filtering process which allows the system to handle information overload situations. Additionally, LIDA implements attentional learning what gives it a capability to improve its own resource management in terms of data selection.

Another AGI system that captured our attention is Auto-catalytic Endogenous Reflective Architecture (AERA), which has been designed as a part of HUMANOBS project: a system able to learn socio-communicative skills by observing people [7]. AERA incorporates an unusual model-based architecture with unique approach to attentional control. In AERA, knowledge is represented as models which are of two types: *forward models* that predict the behavior of entities and *inverse models* that prescribe actions to be taken. Models themselves are executable and its execution is in turn controlled by other models, making the architecture model-driven. Models and other components in AERA are built using low-level building blocks which are objects defined in a specially designed programming language called *Replicode*. There are four fundamental high-level processes which are being continuously performed in a concurrent fashion: Model Acquisition, Reaction, Model Revision and Compaction. Attentional control in AERA can be seen as a part of Reaction process. In general, Reaction activities determine the course of action to pursue goals and are carried out by Attentional Control whose function is to continuously control input saliency within system' specified short time horizon. This approach results in reduction of inputs to the models and more importantly input saliency control allows salience spreading across other objects in the memory including goals and execution traces. Since attentional control is aware about goals, saliency spreading can be controlled and directed by goals.

At present time OpenCog and its implementation OpenCogPrime (OCP) is an advanced and complex integrative AGI system. OCP uses Economic Attention Networks (ECANs), a sophisticated way for attention control and resource allocation [10]. ECANs shares similarities in architecture with connectionist systems but the spread of activation within the system uses equations based on ideas borrowed from economics rather than neural modeling. ECANs is represented as a graph that consists of nodes, links and also HebbianLink and InverseHebbianLink. Each item in a graph, a node or a link, is referred as an *Atom* and assigned two values: Short-Term Importance (STI) and Long-Term Importance (LTI). STI of an *Atom* indicates its immediate urgency at a point in time while LTI indicates amount of value in keeping an *Atom* into the memory. ECANs also

integrates a "forgetting" mechanism that removes certain percentage of *Atoms* with lowest LTI values. Each HebbianLink or InverseHebbianLink designates probability value, that is, given a HebbianLink from A to B it shows the probability if A being in AF so is B, and given InverseHebbianLink from A to B it shows the probability if A being in AF so B is not [10]. ECANs uses defined "economics" equations to update system's values dynamically over time and uses Attentional Focus that treats *Atoms* differently with STI values above certain threshold. Additionally *Atoms* are able to spread its LTI and STI between other *Atoms* connected with HebbianLink or InverseHebbianLink. Clearly equations that modify *Atom*'s values are the critical part of ECANs attentional control.

3 OpenNARS Attentional Mechanism

OpenNARS control mechanism shares some ideas present in systems described above. Its architecture is neither symbolic nor connectionist, rather it incorporates different designs for different components. Attentional control is embedded in OpenNARS architecture through use of **main memory** (concept memory) and **data structure** (Bag), it operates dynamically with system cycles in real-time by employing **budget values**, **truth value** and **budget functions**.

Main Memory in OpenNARS follows a concept-centric memory structure in accordance with the Term Logic the system uses [1]. Main memory can be viewed as a graph where concepts are represented as nodes with its own inner structure. Concepts are linked to each other using *termlinks* based on subterm relationship. For each input or derived task, *task object* is created and is linked to concepts of its subterms using *tasklinks*. Tasks which are a *judgment*, are placed into the *belief table* inside concept node, ranked by their confidence value [1].

Bag is a main component of the system that allows attentional mechanism to efficiently operate. *Bag* is a data structure where the elements are sorted according to their *priority*, and the sampling operation chooses candidates with selection chance proportional to their *priority*. This makes the control strategy similar to Parallel Terraced Scan [4], as it also allows to explore many possible options in parallel, with more computation devoted to options which are identified as being more promising. Please note that *Bag* is different from a priority queue, which just selects the highest priority option, in *Bag* every element even the one with lowest *priority* has an opportunity to be selected. After the selection, a candidate is returned to the *Bag*, with a decrease in *priority* proportional to the *durability* value.

Budget Value is a set of rational values assigned to each data item, it is used to control how much processing should be dedicated to a data item within the system. *Budget value* is a triplet (p, d, q), where p, *Priority*, measures short-term importance, d, *Durability*, a decay rate, describes how fast *Priority* of an item should decay, and q, *Quality*, indicates the long-term importance of the data item. Each value within the *budget value* ranges from 0 to 1.

Truth Value of a statement is a set of two rational numbers (*frequency* and *confidence*) which indicates degree of belief based on the evidence collected

from system's experience. *Frequency* shows amount of positive evidence, while *confidence* indicates degree of reliability of corresponding *frequency*.

Operating Cycle. The main operating cycle of the system is data driven, mainly guided by the priority of data items. It makes effective use of the system's memory structure which is achieved through indefinite repetition of the same inference loop, which is as follows:

1. Add results (derivations and inputs) from global buffer into main memory, triggering potential revisions in the related concepts.
2. Select a concept C from main memory.
3. Select a *tasklink* from C.
4. Select a *termlink* from C.
5. Obtain the highest-confident belief from the concept the *termlink* points to.
6. Apply inference rule with the *tasklink*'s task, and the belief as premises.
7. Adjust *budget value* through use of *budget functions* for data items participated in the inference
8. Input conclusions into global buffer.

4 Control Criteria and Budget Functions

The control mechanism needs to work under AIKR meaning all data structures are bounded in size and eviction strategies to maintain this constraint need to be in place. Additionally, system operating cycle needs to finish roughly in a constant time. A single inference step cannot be interrupted by more important tasks, however an important task is able to interrupt all work carried out over multiple cycles, which captures all the problem solving activities of the system. To allow this happening, the control mechanism has to fulfill additional criteria, for instance, the system should stay responsive to new inputs and derivations. This is achieved through relative forgetting, which makes sure that only contextually relevant items are active at any moment in time. Here the complexity of the inference results matter, since the more complex results need more storage and demand more time to process.

Also historical factors are important in resource allocation, whether a certain inference path was fruitful in the past in a similar context. This is mostly captured through the *quality* of a data item, which can summarize multiple factors, such as "did the selection of the data item led to find answers to questions, or to the fulfillment of a goal?", and more generic considerations such as "how much evidence was summarized by the inference?" as captured by *Budget Inference*. A key here is to see that the criteria for selection and forgetting are quite different. When a selection is made, the context reflected by the *priority* of a data item usually (but not always) matters more than its historic value. For forgetting, on the other hand, the long term *quality* of the item is of interest, though new data item needs to have a chance to prove its usefulness. Many of these considerations have been discussed in [5] as well, and the need to take these considerations into account altogether is what makes designing attentional control for AGI systems a difficult task. In OpenNARS it has led to the development of concrete budget

functions, which are not final but take many of the discussed factors into consideration. The goal of Budget functions is to initialize and adjust *budget value* in the real-time for every data item given its current budget and truth values. Please see the tables for initialization and update of *budget value*.

Table 1. Budget Initialization for each data item

	Priority	Durability	Quality
Task	default value	default value	*Truth_to_Quality()*
Concept	parent task value	parent task value	parent task value
TaskLink	related concept's value	related concept's value	related concept's value
TermLink	$\frac{TaskPriority}{\sqrt{numTermLinks}}$	related concept's task value	related concept's task value

After task derivation, the task's budget obtained by *budget inference* is changed based on how much it is fulfilled in its concept (with same term). Beliefs can satisfy goals to varying degree: the higher their truth expectation (see below), the more will the goal (or question) task be de-priorized. Also, priority will be increased for the tasklink and termlink used in the same inference step, see Table 3. OpenNARS also implements relative forgetting: after an item was selected, participated in inference and put back to the memory its *priority* is decreased by *durability* factor to allow fair competition for resources for other items. Once *priority* drops below certain threshold, item is being removed from the memory.

Short description of Budget Update functions from Table 2:

Table 2. Budget Update Functions

Function and inputs	Priority	Durability	Quality
budgetInference *TruthValue tv, Task t, TermLink b*	**Task Update:or**(priority of t, priority of b) **Term Link Update:** **min**(1,**or**(priority of b, **or**(*Truth_to_Quality(tv)*, priority of b.target)))	**Task Update:** **and**(durability of t, durability of b) **Term Link Update:** **or**(durability of b, *Truth_to_Quality(tv)*)	**Task Update:** *Truth_to_Quality(tv)* **Term Link Update:** no update
merge *BudgetValue b, BudgetValue a*	***max****(priority of a, priority of b)*	***max****(durability of a, durability of b)*	***max****(quality of a, quality of b)*
activate *Concept c, BudgetValue b*	***or****(priority of c, priority of b)*	***avg****(durability of c, durability of b)*	*quality of c*
revise *TruthValue t, TruthValue b, TruthValue r*	***or****(t.priority, r.confidence-* ***max****(t.confidence, b.confidence))*	***avg****(t.durability, r.confidence-* ***max****(t.confidence, b.confidence))*	*Truth_to_Quality(r)*

1. **budgetInference** creates a budget for derived task and also updates budget for selected concept's termlink
2. **merge** revises budget when merging identical items
3. **activate** updates currently selected concept's *budget value*
4. **revise** assigns budget to item whose truth value derived using revision rule.

Budget functions use utility functions defined separately and common to numerous other evaluations in OpenNARS. The four utility functions are present below:

1. **TruthExp** truth expectation defined as $confidence * (frequency - 0.5) + 0.5$
2. **Truth_to_Quality** is defined as $max(TruthExp, 0.75 * (1 - TruthExp))$
3. **or**$(x_1 ... x_n)$ is defined as $1 - \prod_{i=1}^{n}(1 - x_i)$
4. **and**$(x_1 ... x_n)$ is defined as $\prod_{i=1}^{n} x_i$.

5 Experiments

Experiment 1. The first experiment shows a reasoning tasks and the selections made by the control system, and serves as an example to understand how budget of derivation exactly is calculated according to the budget functions.

At first, two tasks are entered by the user, ($cat \rightarrow animal$) and ($dog \rightarrow animal$). Both get a default truth value attached, which is frequency 1.0 and confidence 0.9. The system trace outputs:

Table 3. Task Satisfaction where q_s is the solution quality, the confidence of the solution, such as a belief to a question or goal, if the term is equal to the question, and else its truth expectation.

Derived task priority	Tasklink priority	Termlink priority
min*(1-q_s, TaskPriority)*	***or*** *(****min****(1-q_s, TasklinkPriority))*	***or*** *(q_s, TermlinkPriority)*

```
!!! Perceived: $0.8000;0.8000;0.9500$ <cat --> animal>. %1.00;0.90%
!!! Perceived: $0.8000;0.8000;0.9500$ <dog --> animal>. %1.00;0.90%
```

We also see initialized default *budget values* (p, d, q), where $p = 0.8$, $d = 0.8$, $q = 0.95$ and q were calculated using truth value (f, c) of the task via:

$truthToQuality(f, c) = max(exp(f, c), (1 - exp(f, c)) * 0.75)$ where $exp(f, c) = (c * (f - \frac{1}{2}) + \frac{1}{2})$. The factor 0.75 is used to assign a higher quality value for positive (frequency > 0.5) results than negative ones.

Now in the first cycle, at first the new input tasks are inserted from global buffer into memory. Then the concept-based operating cycle as described before, chooses *animal* as concept, ($cat \rightarrow animal$) as task via the chosen tasklink from concept *animal*, and ($dog \rightarrow animal$) as belief via the chosen termlink from concept *animal*:

```
 * Selected Concept: animal
 * Selected TaskLink: $0.5657;0.8000;0.9500$ _@(T4-2) <cat --> animal>. %1.00;0.90%
 * Selected TermLink: $0.5657;0.8000;0.9500$ _@(T4-2) <dog --> animal>
 * Selected Belief: <dog --> animal>. %1.0000;0.9000%
```

(Here T4 stands for termlink type "COMPOUND_STATEMENT" meaning the termlink points into a statement, and the index is 2, pointing to *animal* within the statement, which is the case for both links.)

The links got their priority from distributing the task priority among the components using the function: $\frac{p}{\sqrt{n}}$ where n is the amount of subterms, which is 2, so the result is $\frac{0.8}{\sqrt{2}} = 0.5657$, the durability and quality is the one of the task, though gets updated after the following derivation formed with the intersection truth function, which led to a truth value of $(f, c) = (1.0, 0.81)$:

```
!!! Derived: $0.9131;0.1338;0.1810$ <(|,cat,dog) --> animal>. %1.00;0.81%
 from task: $0.80;0.80;0.95$ <cat --> animal>. %1.00;0.90%
 from belief: <dog --> animal>. %1.00;0.90%
```

The complexity of the derived term (which stands for "cats and dogs are both animals" is 5. The termlink priority was 0.5658. The derived priority of 0.9131 was obtained by $or(0.5658, 0.8) = 1 - (1 - 0.5658) * (1 - 0.8) = 0.91316$. And the derived quality 0.1810 was obtained from $\frac{truthToQuality(1.0,0.81)}{c} = \frac{0.905}{5}$. Also the link budgets are now updated by increasing them with the *or* function, where both target activation (priority of the belief concept) and $q^* = \frac{q_{result}}{complexity}$ are "added" to the existing link priority, meaning $p_{termlink_{new}} = or(p_{termlink_{old}}, p_{beliefconcept}, q^*)$. Other values are in the budget tables above.

Experiment 2. This example shows properties of the control system on a higher level. It demands the system to form a certain subset of letters from a–j:

```
<{a} --> letter>. ... <{j} --> letter>.
//<{a} --> letter>? //<{a,f} --> letter>? (*) //<{a,f,g} --> letter>? <{a,f,g,j} --> letter>?
```

When the questions marked with "//" are given to the system (after 1000 inference steps before each question) together with the other input, the system arrives with the answer to the last question within 6151 inference steps (OpenNARS v3.0.4). When on the other hand only the question marked with a star is provided additional to the final question, it takes 7728 steps. With only the final question it takes the system 260320 inference steps to find the answer. This shall serve as an example of how contextual priming can help in the search for solutions. The key budget function allowing this is *activate*, which increases the *priority* of a concept when a task with the same term arrives within it. In this case it's the appropriately timed user questions which trigger this form of priming. Contextual priming is also a key in avoiding combinatorial explosion in reasoning, additionally to the term logic which makes sure the premises to derive a conclusion are semantically related by sharing a common term.

6 Discussions

It is difficult to compare OpenNARS with the attentional controls of systems present in Sect. 2 since architectures are very different, however some similarities can be observed. As one might see OpenNARS's attentional mechanism is quite complex and operates in the real-time during every inference. Once a data item

is participating in an inference, its *budget value* as well as the one of the related items are inevitably affected resulting in a budget activation spread within the main memory. Spread of attentional values (*salience*, STI, LTI) also exists in AERA and OCP. AERA implements the attentional control partially as an input data filtering to a model, and then *salience* is being spread to other models allowing control of input data. In LIDA, attentional control is implemented through use of attention codelets which pick an "interesting" data from Workspace, form a coalition with data from Episodic Memory and move it to Global Workspace. LIDA attentional control can be viewed as data filtering process that filters low importance items and safeguards the system from information overload.

OCP on the other hand is more similar to OpenNARS, its attentional mechanism is embedded in its architecture. ECAN applies equations to update STI, LTI, HebbianLink and InverseHebbianLink probabilities, similarly to OpenNARS Budget Functions that update *budget value* of a data item. In ECAN, STI and LTI spread is happening through HebbianLinks and InverseHebbianLinks. Finally, the Copycat project, uses *salience* for structures in Workspace and *urgency* for codelets. Copycat's resource allocation ideas share similarities with OpenNARS's approach of selecting items. It selects codelets from Coderack with selection chance proportional to the codelet's *urgency*, while in OpenNARS items's *priority* is treated as probability for selection allowing lowest *priority* items to compete for resources as well. Many items will be filtered out completely though, due to the bounded size of the *Bag*.

7 Conclusion

Important details of *OpenNARS*'s control mechanism and attentional control were described, their motivation explained and demonstration provided. OpenNARS has been applied in applications such as [6], though its control system, despite its successes, was not yet published in detail. Inference control is a major problem to be solved for reasoning-based AGI systems such as OpenNARS, which makes documenting the advancements even more crucial. It will help the AGI field to find better attention mechanisms with proper ways to take the usefulness, relevance, truth, and complexity of results into consideration. Our future research will include important metrics to measure control system capabilities, which can become a basis to compare different reasoning-based AGI approaches.

References

1. Wang, P.: Non-Axiomatic Logic: A Model of Intelligent Reasoning. World Scientific, Singapore (2013)
2. Hammer, P., Lofthouse, T., Wang, P.: The OpenNARS implementation of the non-axiomatic reasoning system. In: Steunebrink, B., Wang, P., Goertzel, B. (eds.) International Conference on Artificial General Intelligence, pp. 160–170. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-41649-6_16
3. Wang, P.: Insufficient Knowledge and Resources-A Biological Constraint and its Functional Implications. In: 2009 AAAI Fall Symposium Series, October 2009

4. Rehling, J., Hofstadter, D.: The parallel terraced scan: an optimization for an agent-oriented architecture. In: 1997 IEEE International Conference on Intelligent Processing Systems (Cat. No. 97TH8335), vol. 1, pp. 900–904. IEEE, October 1997
5. Helgason, H.P.: General attention mechanism for artificial intelligence systems. University of Reykjavik, Ph.D., June 2013. https://en.ru.is/media/td/Helgi_Pall_Helgason_PhD_CS_HR.pdf
6. Hammer, P., Lofthouse, T., Fenoglio, E., Latapie, H.: A reasoning based model for anomaly detection in the Smart City domain. In: Advances in Intelligent Systems and Computing (2020)
7. Nivel, E., et al.: Autonomous Endogenous Reflective Architecture (2013)
8. Franklin, S., Madl, T., D'mello, S., Snaider, J.: LIDA: a systems-level architecture for cognition emotion and learning. IEEE Trans. Autonom. Mental Dev. **6**(1), 19–41 (2013)
9. Hofstadter, D.R., Mitchell, M.: The copycat project: A model of mental fluidity and analogy-making - D, pp. 205–267. Fluid Concepts and Creative Analogies, Hofstadter and the Fluid Analogies Research group (1995)
10. Ikle, M., Pitt, J., Goertzel, B., Sellman, G.: Economic attention networks: Associative memory and resource allocation for general intelligence (2009)