Adaptive Neuro-Symbolic Network Agent

Patrick Hammer

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

Abstract. This paper describes Adaptive Neuro-Symbolic Network Agent, a new design of a sensorimotor agent that adapts to its environment by building concepts based on Sparse Distributed Representations of sensorimotor sequences. Utilizing Non-Axiomatic Reasoning System theory, it is able to learn directional correlative links between concept activations that were caused by the appearing of observed and derived event sequences. These directed correlations are encoded as predictive links between concepts, and the system uses them for directed concept-driven activation spreading, prediction, anticipatory control, and decision-making, ultimately allowing the system to operate autonomously, driven by current event and concept activity, while working under the Assumption of Insufficient Knowledge and Resources.

Keywords: Non-Axiomatic Reasoning · Sensorimotor · Artificial General Intelligence · Procedure Learning · Autonomous Agent

1 Introduction

Adaptive Neuro-Symbolic Network Agent (ANSNA), is a new design of a sensorimotor agent derived from Non-Axiomatic Reasoning System (NARS) theory proposed by Pei Wang (see [1]). It adapts to its environment by building concepts based on Sparse Distributed Representations [2] of sensorimotor sequences, rather than based on Compound Terms that are typical for NARS. It does so by taking theory of compositionality of bit vectors as proposed by [3] into account, which not only captures union and difference operations between bit vectors, but also ways to encode hierarchical structure within them.

Making use of Non-Axiomatic Reasoning System theory, ANSNA is able to learn directional correlative links between concept activations that were caused by the appearing of observed and derived event sequences. These directed correlations are encoded as predictive links between concepts, and the system uses them for directed concept-driven activation spreading, prediction, anticipatory control and decision-making. All that allows the system to operate autonomously under the Assumption of Insufficient Knowledge and Resources, driven by current context, determined by event and concept activity.

2 Similar work and philosophical differences

ANSNA borrows most of its theory from the Non-Axiomatic Reasoning System proposed by Pei Wang (see [1]), while using the inference control theory of ALANN [5], which is a NARS-variant designed by Tony Lofthouse. What makes ANSNA really different from NARS is however the complete absence of *Terms* and explicit *Inheritance* relationships, coming from a philosophically very different path: while NARS tries to model a general-purpose thinking process with highly flexible ways to compare, transform, and generally deal with any kind of information that can somehow be expressed in *Narsese* (NARS's formal internal and I/O language), ANSNA concentrates completely on sensorimotor.

For NARS, sensorimotor capability, which consists mainly of procedural and temporal inference on sensor & motor events, is just a special case of rich reasoning abilities its Non-Axiomatic Logic (NAL) supports. NAL also includes declarative reasoning abilities about sets, arbitrary relations, and inheritance-relationships that are all there to support dealing with conceptual knowledge that doesn't necessarily have any grounding in actual sensorimotor experience. ANSNA takes the position where knowledge that has no possible grounding in the system's sensorimotor experience is not necessarily meaningless (as it can clearly relate to other knowledge), but surely was so far useless to a goal-driven decision-maker, as it would mean that the meaning of that knowledge is completely orthogonal to everything ANSNA has ever experienced through its sensors so far, both external and internal. In NARS this situation is by far not unusual, a user entering a new Inheritance relationship (term123 $\rightarrow$ term242) consistent only of new terms, term123 and term242, leaves the system's memory with a floating pair of concepts that have so far no relation to any other concepts whatsoever, meaning also no relation to sensorimotor concepts, and how such a relation should be established through correlations is a difficult problem. Such a problem does not exist in ANSNA, as it is assumed that all information is consumed through external (vision, touch, sound, temperature, other modalities...) and internal sensors (battery level, structural integrity, etc.).

According to ANSNA philosophy, relating new user-given abstract terms to sensorimotor experience is not something an AGI has to do, but that building compositions of sensorimotor patterns is everything necessary. That is, because in ANSNA every composition simply cannot even be "not grounded", since every information, without exception, ultimately is forced to enter ANSNA through the system's sensors. Also in a NARS operating in a robot without Narsese-communication channel, it is usually not happening, and not at all necessary, that new atomic terms will be created, in such a case the set of atomic terms are pre-defined by the designer, consisting of pre-defined sensor encodings and probably revisable background knowledge that was loaded on the robot beforehand. In that sense, a semantic code is inevitable, meaning the universe of mental discourse will be spanned by possible compositions of events following pre-defined encodings of sensory data (plus combinations with background knowledge, in NARS). Even though NARS itself does not assume a fixed semantic code, in that case it is undeniably present. This is however no contradiction with that such a system can acquire the meaning of observed events, where the meaning of an event has both structural and empirical aspects.

Structural meaning is determined by the composition following the semantic code, which encodes how the pattern is observed/composed from sensorimotor experience. For instance there is no way for the system to see the observation of a red ball as structurally identical to an observed blue ball. However, it needs to be possible for the system to learn that a blue ball carries overlapping meaning, not only by being a similar structural composition / semantic code word, but also that nudging a blue ball in similar circumstances, will have similar consequences like nudging a red ball in similar contexts. And that can be done without having the user entering an explicit Inheritance relationship into the system, and without an explicit Inheritance altogether, as whether experienced event a is a special case of another event b can implicitly be represented by sensorimotor relations, that is, if a leads to the consequences we expect from b, it is naturally a special case of the former even though it may structurally differ.

Of course, the semantic code needs to be rich, not in quantity, but in quality. Same as a set of lego technic pieces needs to be rich in variety and fit together nicely to support the construction of a large variety of machines, the semantic code needs to be rich in variety and fit together in such a way, that the agent is able to conceptualize experienced aspects of its environment in an effective way. This can happen through a large variety of perceptual attributes, such as, for example, Color, PositionX, PositionY, Pitch, Frequency, Temperature, Pressure and Battery Level. Color, PositionX and PositionY can encode information from a visual field, for instance. Once a basic semantic code is in place, the encoders are present, everything the system experiences will be seen in terms of the attributes these encoders present, by ANSNA. The more comprehensive, the richer the context will be, and the better will ANSNA be able to make sense of its environment through compositions of sensorimotor events. This leads to the last key difference to OpenNARS and ANSNA, the usage of Sparse Distributed Representations (long, sparse bit vectors, SDR's), and usage of Pentti Kanerva's [3] insights about how hierarchical structure can be encoded in them. Clearly, differently than Sparse Distributed Memory (SDM) [6], ANSNA is not just a model of memory, and thus, as we will see, its event-based design requirements make its memory architecture different than SDM, while preserving some of SDM's key properties. For instance, mapping events with similar SDR's to similar concepts, supporting content-addressable memory.

3 Data Structures

ANSNA's memory consists of two priority queues, one contains concepts and the other current events (Events Buffer).

Event: Each Event consists of a SDR with a NAL Truth Value, an Occurrence Time, and a Attention Value that consists of the priority of the event and a durability value that indicates the decay rate of the priority over time.

A SDR is a large bit-vector with most bits being zero, in ANSNA all SDR's are of equal length n.

SDR structure: With a,b being SDR's we can now define the following functions calculating a new SDR based on a existing one, using theory borrowed from P. Kanerva [3]: SDRSet$(a, b) := a|b$ where $|$ is the bitwise or operation. SDRTuple$(a, b) := \Pi_S(a) \oplus \Pi_P(b)$ where Π_S and Π_P are two random permutations selected when ANSNA starts up, they remain the same after that.

Additionally encoding functions E as proposed in [7] are used to encode similar numbers to similar SDR's, and terms are encoded into random SDR's deterministically. This way, arbitrary hierarchical compositions can be encoded into ANSNA, and as we will see later, effectively compared with each other based on a per-bit basis. For now it is sufficient to see that two input encodings $SDRTuple(E(brightness), E(3.23))$ and $SDRTuple(E(brightness), E(3.5))$ will lead to similar SDR's, meaning most 1-bits will overlap. We will omit E from now on, and see that $SDRSet(green, light)$ will have more 1-bits in common with $light$ than $sound$. Of course $SDRTuple$ and $SDRSet$ can be arbitrary nested with each other, essentially forming a tree which leafs are for instance SDR-encoded terms or numbers, and structurally similar trees will lead to similar SDR's.

Concept: Concepts in ANSNA are summarized sensorimotor experience, they are the components of ANSNA's content-addressable memory system and are named by interpolations of the events SDR's that matched to it (described in more detail in the next section). Processed events can match to different concepts with various degree, but in a basic implementation a winner-takes-all approach can be taken, matching the event only to the most specific matching case that was kept in memory, and processing it as such.

Each concept has a SDR (its identifier), and Attention value consisting of a priority and a durability value, a Usage value, indicating when the concept was last used (meaning it won the match competition for an event, as we will see later) and how often it was used since its existence. Also it has a table of pre- and post-condition implications that are essentially predictive links, specifying which concepts activate which others, and a FIFO for belief and goal events, and has multiple responsibilities:

To categorize incoming events by matching them to its SDR: to become good representatives, concepts have to encode useful and stable aspects of a situation, conceptual interpolation, explained in the next section, helps here; To support revision, prediction and explanation for native events, events for which this concept wins the matching competition; To maintain how relevant the concept is currently and how useful it was in total so far; Learning and revising preconditons and consequences by interacting with an for temporal inference incoming event.

Matching events to concepts: An event can match to multiple concepts with a truth value "penalty" according to the match. Let S and P be a SDR. We want that S can be said to be a special case of P, or can stand for P, denoted by $S \to P$, if most of the bits in P also occur in S, but not necessarily vice versa.

So S =SDRSet(red,ball) should be a special case of P =SDRSet(ball). It has most the features of ball, but also has the redness feature, meaning a red ball can effectively stand for, or be treated as a ball too.

We will now formalize this idea using a NAL truth value, which is a frequency-confidence tuple $(f, c) = (\frac{w_+}{w_+ + w_-}, \frac{w_+ + w_-}{w_+ + w_- + 1})$ where w_+ is positive evidence and w_- negative evidence. The truth value of $S \rightarrow P$ can be established as follows: Let's define each 1-bit in the SDR to be a NAL sentence (see [8]), where each of these 1-bits, at position i, in S, encode $bit_i = 1$.

One case of positive evidence for $S \rightarrow P$, is a common property S and P both share. Such as the fact that bit_5 is a 1-bit. On the other hand, a case of negative evidence would be a property possessed by P that S does not possess. Given that, we can define the positive evidence as: $w_+ := |\{i \in \{1, ..., n\} | S_i = P_i = 1\}|$ and the negative evidence as $w_- := |\{i \in \{1, ..., n\} | S_i = 1 \wedge P_i = 0\}|$.

If the event E has truth value T_E, to apply the penalty of "treating it as concept C", the truth value becomes Truth_Deduction(T_{match}, T_E), which will then be used in the inference rule within the concept for deriving further events.

That is motived by that if event E is a special case of the pattern it is encoded by, $SDRE$, and $SDRE$ is a special case of $SDRC$, as the match determined, then we have $E \rightarrow SDRE$ with truth value T_E and $SDRE \rightarrow SDRC$ with truth value $T_{match} := SDR\_Inheritance(S, P)$. Using the deduction rule as specified in [8], we end up with $E \rightarrow SDRC$, allowing to treat the event as if it would have the SDR SDRC.

Please note there is also a symmetric match defined by Truth_Intersection(SDR_Inheritance(a,b), SDR_Inheritance(b,a)) as we will need later. For a tuple of truth values $((f, c), (f_2, c_2))$ Truth_Intersection leads to $(f * f_2, c * c_2)$ and Truth_Deduction to $(f * f_2, f * f_2 * c * c_2)$, for the other truth functions we will use, please see [8], they have all been described by Pei Wang in detail.

Event FIFO and Revision: While pushing a new event to the first position when a matched event enters a concept's FIFO, to resolve goal conflicts in respect to a current decision, in the goal event FIFO, revision with the highest confident element when projected to the goal occurrence time (where projected means multiplicatively penalized for occurrence time difference dt according to α^{dt}, where α is a truth projection decay parameter) has to happen, the result will then be pushed to the first FIFO position. Of course, the revision (which sums up the positive, and negative evidence of both premises) can also happen in the belief event FIFO, this make sure that two conflicting sensory signals that happen concurrently, will be merged, allowing to better deal with contradicting sensory information. [1]

Implication Table and Revision: In NARS terms, Implications in ANSNA are eternal beliefs of the form $a \Rightarrow b$, which essentially becomes a predictive link for a and a retrospective link in b, each going to a separate implication table (preconditions and postconditions).

[1] A detail: As in [9], only revise if the evidental base does not overlap , and only if the revised element when projected to the occurrence-time middle between both elements is higher than the premises's.

An implication table combines different implications, for instance $a \Rightarrow b$ and $a \Rightarrow c$ to describe the different consequences of a in the postcondition table of concept a. Implication tables are ranked by the truth expectations of the beliefs, which for a given truth value (f, c) is defined as $(c * (f - \frac{1}{2}) + \frac{1}{2})$.

Different than in OpenNARS, where it is clear whether revision can happen dependent on whether the terms are equal, two items in ANSNA can have different degree of SDR overlap. To deal with this, both revision premises are penalized with symmetric SDR match SDR_Similarity, leading to $Truth1$ and $Truth2$ using Truth_Intersection, and revision will only occur if revision(Truth1,Truth2) has a higher confidence than both Truth1 and Truth2. When a new item enters the table, it is both revised with the closest SDR candidate (the revised result will be added to the table, if it was a proper result), and also the original Implication will be added to the table.

Conceptual Interpolation: Conceptual interpolation, inspired by [6], is the process by which concept's SDR adapts to the SDR's of the matched events, in such a way that the SDR of the concept becomes the average case among the matched event SDR's. This allows the concepts to become useful "prototypes" under the presence of noise, useful in the sense that a newly seen noisy pattern can be reconstructed. A way to implement this is idea is to add a counter for each bit in the SDR. Each 1-bit of the matched event increases the corresponding counter by 1*u, and each 0-bit decreases it by 1*u, where $u =$Truth_Expectation(SDR_Inheritance(e,c)), meaning an event that better matches to the concept will have a stronger influence on it. If the counter is 0 or smaller, the corresponding concept SDR's bit will be 0, else 1. This effectively means that iff there is more positive evidence for the bit in the matched event SDR's to be 1 than 0, it will be 1 in the concept SDR they were matched to too.

4 Attentional Control

While on a conceptual level Attentional Control in ANSNA allows the processing of different items with individual speeds (as also NARS [1], [12] and Hofstadter's group's creations [11]), the details in ANSNA mostly follow the Adaptive Logic and Neural Network (ALANN) control model by Tony Lofthouse, which was developed for a NARS implementation over the last two decades, based on expertise about Spiking Neural Networks. Although a convincing prototype exists [5], unfortunately this model was not published in a scientific publication yet, so its background is explained in addition to implementation details.

Every NARS, and AGI in general, faces the problem of fulfilling practically infinite resource-demands with a finite amount of resources [12] which are ultimately limited by the processor speed and the RAM available on the machine it runs on. An Attention model [13] can solve this problem as it allows to selectively perform inferences based on contextual cues, which primary importance the ALANN model stresses. It does so by building an analogy to the brain, which is known to be able to save energy by keeping only a tiny proportion of neurons active at the same time [14]. In biological neural networks, spike cascades appear, where spikes are sent from one neuron to the next, and, potentially even further, if the action potential threshold of the source neuron is overcome ([15]), while avoiding re-activations through cyclic connections by enforcing a certain refractory/latency period. The priority value of the spike sent to the target neuron depends on the synapse strength and the current action potential, the latter we will call concept priority. In ALANN, the synapse strength is assumed to correspond to the strength of a certain experienced pattern, which is summarized by a NAL [8] truth value corresponding to a belief that is related to the concept the neuron represents. Using NAL as a foundation, this is natural, as concept node *Lighting* can have related beliefs like $Lighting \Rightarrow Thunder$, in that sense, the belief acts like a link connecting concept *Lighting* to $Thunder$, making the *Lighting* concept emit $Thunder$ events that are received by the latter, and whenever that happens, also the concept priority of $Thunder$ will be increased naturally due to the spike derivation, based on the priority value of the spike, and will decrease quickly. From now on, we will refer to spikes as events, neurons as concepts, synapse strength as belief truth, also to take a safe distance from actual claims of how the wetware actually functions. In ANSNA, action potential thresholds are never fixed, instead it is realized by enforcing a fixed number of active events to be selected from a global priority queue that is ranked by the event priorities, and where the topmost k items are selected. Using this model, ANSNA consists of the following attention update functions:

Attention_forgetEvent: Forget an event using exponential decay. To make lazy update possible, the decay is stronger the longer it wasn't selected anymore. Also this one needs to be radical, there is only a very small window in time in which it is likely for the target concept to generate further derivations, to make sure derivations are still contextually relevant.

Attention_forgetConcept: Forget a concept with exponential decay, again, the more, the longer it wasn't selected anymore, additionally a lower "priority threshold" is established, that is dependent on the concept's usefulness. This threshold hampers useful concepts to be forgotten. Usefulness is calculated in the following way: age = currentTime - lastUsed, v = useCount / age, usefulness = v / (v + 1.0); Additionally the neural-network-motivated activation spreading functions applied to event derivations are:

Attention_activateConcept: Activates a concept because an event was matched to it, proportional to the priority of the event. The idea here is that the concept sums up the appearing event priorities while leaking priority over time, this way the active concepts tend to be currently contextually relevant ones.

Attention_deriveEvent: The derived event gets higher priority if the involved concept had a high priority (the derivation was contextually relevant), and also gets higher priority if the truth expectation of the for the derivation used belief (a belief event of belief_event FIFO, or an Implication from a pre- or post-condition table, as we will see later) was high (the synapse had high strength).

Attention_inputEvent: Priority positively correlated with the truth expectation of the input event.

5 Operating cycle

Inference Schemas: The following describes all types of inference that can happen in the operating cycle introduced next, and the truth functions that apply are defined in [8], where a leading "!" means goal, and "." means belief:

- Revision, in Event FIFO, and in Implication Table (Link growth):
  {Implication/Event a, Implication/Event a} $\vdash$ Implication/Event a
- Deduction (Prediction):
  {Event a., Implication ($a \Rightarrow b$).} $\vdash$ Event b.
  {Event b!, Implication ($a \Rightarrow b$).} $\vdash$ Event a!
- Induction (Link formation):
  {Event a., Event b.} $\vdash$ Implication (a => b).
- Abduction (Prediction):
  {Event b., Implication ($a \Rightarrow b$).} $\vdash$ Event a.
  {Event a!, Implication ($a \Rightarrow b$).} $\vdash$ Event b!
- Intersection (Concept formation):
  {Event a., Event b., after(b, a)} $\vdash$ Event SDRTuple(a, b).
  {Event a., Event b., concurrent(b, a)} $\vdash$ Event SDRSet(a, b).
  where concurrent and after are excluding each other: when the occurrence time of a and b is closer than a global system parameter, concurrent(a, b) is true, else either after(b, a) or after(a, b) is true.

Operating Cycle: In each cycle, a fixed number of events (input or derived) get taken out from Events Buffer and processed: a concept will be created for them (if one with exact same SDR doesn't already exist), and they match the best asymmetrically matched concept available (not including the created one), also increasing its priority using Attention_activateConcept. The event (which truth value was reduced consistent with the asymmetric match explained previously) then interacts with the events within the concepts FIFO for revision, as explained previously. Also it interacts with the postcondition implication table (the highest truth-expectation elements, a choice rule), triggering a Deduction[2] if it is a belief event, and an abduction if it is a goal event. And it interacts with the precondition implication table, triggering an Abduction if it is a belief event, and a deduction if it is a goal event, both consistent with the Schemas.

Also the event gets sent to the k highest-priority concepts (not including the matched one) as a "foreign concept", not reducing its truth value (this interaction is not a match, just a correlation in activity between event and concept!). The only purpose of that interaction is to compose new, more complex temporal sequences that are themselves events to be processed, consistent with the Intersection Schema in the table, using Attention_deriveEvent to determine the derived event's Attention value. Additionally, sequence (a, b) leads to the formation of hypothesis $a \Rightarrow b$ which directly enters the postcondition implication table of a as a "predictive link" and precondition implication table of b as a "retrospective link". (Time durations are stored too, and averaged on revision)

All derived sequences enter the global Event Buffer, of which all elements taken out from re-enter with adjusted Attention value as defined by Attention_forgetEvent. Note that also the k used concepts get their priority reduced by Attention_forgetConcept. This means that all the attention updates are driven by event processing. All summarized:

[2] which generates an Anticipation, that if it won't get confirmed, adds negative evidence to the implication (predictive link) that generated the prediction. (as AERA)

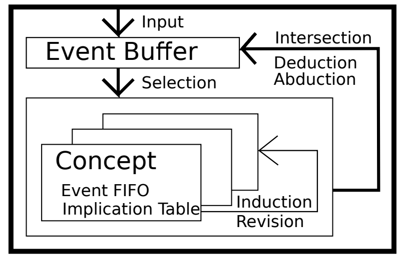

Fig. 1. Overview with Event Buffer and concepts, plus their predictive links. Operating cycle selects events from Event Buffer (priority-biased), lets them interact with the matched concept for Intersection, Deduction and Abduction, and with high-priority concepts for Temporal Induction, and as result derives further events that end up in Event Buffer, and predictive links that end up in the implication tables.

Decision Making Decision Making in ANSNA was taken from NARS [16] and adjusted to fit well to ANSNA's memory model:

Operations: These are a (SDR, Action) tuple, Action is a software procedure without arguments, expected to finish in constant time. They are registered using ANSNA_RegisterOperation(SDR sdr, Action procedure) method. For now the SDR serves mostly as an ID, but formats for motor operations allow the system to see similar parametrizations as similar, for instance the SDR encoding of (motor1,0.7) will naturally be more similar to (motor1,0.8) than to (motor1,0.2), which opens interesting opportunities for fine-grained control.

Decision Making Rule: When a goal event gets matched to a concept and added to its FIFO as described earlier, the goal event, or instead the revised one in case that revision happened, if of form (SDR,Op_SDR_i), determines the operation (Op_SDR_i, Action_i) stored in the system. In that case, the event gets projected to the current moment, leading to a certain truth value T_P. Now the system retrieves the next event b from belief_event FIFO that has no associated operation and has the highest truth confidence of its truth value T_b when projected to the current time and calculates T_{Result} =Deduction(T_P, T_b). If this truth value's expectation is above the system's decision threshold parameter, the corresponding procedure $Action_i$ gets called, capturing context and intention, and the truth of the procedure knowledge is considered by goal-derivation.

Procedure Learning: To make the system aware of the execution of an action, for each of the k highest priority-concepts (that are selected in each cycle, as described previously), the first belief event FIFO element gets "copied", This copy receives a new SDR, being SDRSet(OldSDR, Op_SDR$_i$) (also allows for compound operations), and is then added to the FIFO without revision, making the system effectively re-interpret the event as being a precondition under which the operation was executed, so that when a next event with SDR c interacts with the concept for temporal inference, ((OldSDR,Op_SDR_i) $\Rightarrow$ c) will naturally be formed with temporal induction, a piece of procedure knowledge, specfiying that the execution of Op_i leads to c under the condition of OldSDR.

Motor Babbling: To trigger executions at the beginning where no procedure knowledge exists yet, the system invokes random motor operations from time to time, a process called Motor Babbling. Without any initial operations, the system couldn't learn how it can affect the environment, so this serves as an initial trigger for procedure learning. The same idea is used in [17]. Initial reflexes are also a potentially helpful, similar ones like the grab reflex in humans are possible too, but these are more domain-specific.

6 Conclusion

A new autonomous sensorimotor agent architecture, Adaptive Neuro-Symbolic Network Agent, is proposed. Differently than NARS from Pei Wang, which it is derived from, it uses SDR's for knowledge representation, and a inference control mechanism inspired by spiking neural network derived from Tony Lofthouse's easily parallelizable ALANN model. Its key benefits, besides being more concise than NARS, lie in the ability to process a large quantity of information effectively, and to mine temporal-spatial patterns in its experience that allow it to predict what will happen next, and make decisions accordingly, to realize its goals.

References

1. Wang, P. (2004). Toward a unified artificial intelligence. In AAAI Fall Symposium on Achieving Human-Level Intelligence through Integrated Research and Systems (pp. 83-90).
2. Hawkins, J., & George, D. (2006). Hierarchical temporal memory: Concepts, theory and terminology (p. 2). Technical report, Numenta.
3. Kanerva, P. (2009). Hyperdimensional computing: An introduction to computing in distributed representation with high-dimensional random vectors. Cognitive computation, 1(2), 139-159.
4. Rinkus, G. J. (2014). Sparsey^{TM}: event recognition via deep hierarchical sparse distributed codes. Frontiers in computational neuroscience, 8, 160.
5. Lofthouse, T. (2018). Adaptive Logic and Neural Network (ALANN). Retrieved from https://github.com/opennars/ALANN2018
6. Kanerva, P. (1988). Sparse distributed memory. MIT press.
7. Purdy, S. (2016). Encoding data for HTM systems. arXiv preprint arXiv:1602.05925.
8. Wang, P. (2013). Non-axiomatic logic: A model of intelligent reasoning. World Scientific.
9. Hammer, P., Lofthouse, T., & Wang, P. (2016, July). The OpenNARS implementation of the non-axiomatic reasoning system. In International conference on artificial general intelligence (pp. 160-170). Springer, Cham.
10. Rogers, D. (1988). Kanerva's sparse distributed memory: An associative memory algorithm well-suited to the connection machine.
11. Rehling, J., & Hofstadter, D. (1997, October). 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.
12. Wang, P. (2009, October). Insufficient Knowledge and ResourcesA Biological Constraint and Its Functional Implications. In 2009 AAAI Fall Symposium Series.
13. Helgason, H. P. (2013). General attention mechanism for artificial intelligence systems (Doctoral dissertation, Ph. D. dissertation, Reykjavik University, 2013.[Online]. Available: http://skemman. is/en/item/view/1946/16163).
14. Harris, J. J., Jolivet, R., & Attwell, D. (2012). Synaptic energy use and supply. Neuron, 75(5), 762-777.
15. Bean, B. P. (2007). The action potential in mammalian central neurons. Nature Reviews Neuroscience, 8(6), 451.
16. Hammer, P., & Lofthouse, T. (2018, August). Goal-Directed Procedure Learning. In International Conference on Artificial General Intelligence (pp. 77-86). Springer, Cham.
17. Nivel, E., & risson, K. R. (2013). Autocatalytic endogenous reflective architecture.