The OpenNARS Implementation of the Non-Axiomatic Reasoning System

Patrick Hammer[1(✉)], Tony Lofthouse[2], and Pei Wang[3]

[1] Institute for Software Technology,
Graz University of Technology, Inffeldgasse 16b/II, Graz, Austria
patrickhammer9@hotmail.com
[2] Evolving Solutions Ltd., Newbury, UK
Tony.Lofthouse@GMILab.com
[3] Department of Computer and Information Sciences,
Temple University, Philadelphia, PA 19122, USA
pei.wang@temple.edu

Abstract. This paper describes the implementation of a Non-Axiomatic Reasoning System (NARS), a unified AGI system which works under the assumption of insufficient knowledge and resources (AIKR). The system's architecture, memory structure, inference engine, and control mechanism are described in detail.

1 Introduction

NARS is an adaptive system that works under the Assumption of Insufficient Knowledge and Resources (AIKR) [6,8], meaning the system has to work under the restrictions of being: *Finite:* the information processing capability of the system's hardware is fixed, *Real-time:* all tasks have time constraints attached to them and *Open:* no constraint is put on the content of the experience that the system may have, as long as it's expressible in the interface language [8].

Built in the framework of a reasoning system, NARS has a memory, a logic component and a control component. The logic component consists of inference rules that work on statements, where the statements are goals, questions and beliefs. A statement can be eternal (non time-dependent) or an event (time-dependent). Beliefs are statements that the system believes to be true to a certain degree and goals are statements the system desires to be true to a certain extent. An inference task is a statement to be processed, with additional control relevant information.

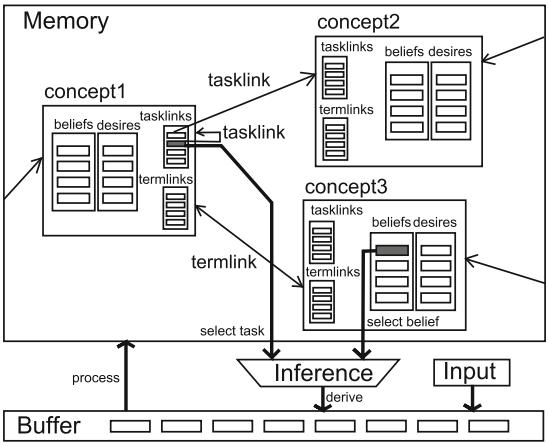
NARS utilises the Non-Axiomatic Logic (NAL) [9] for inference and the Narsese language for representing statements. The language and the logic are outside the scope of this document. The aim of this paper is to describe the current implementation of NARS in detail. The following aspects of the implementation are focused on: memory management with concept centric processing, non-deterministic selection capabilities allowing anytime-processing of tasks, resource constraint management, a logic system with meta rule DSL and Trie based execution engine, temporal inference control (including temporal windows, temporal

B. Steunebrink et al. (Eds.): AGI 2016, LNAI 9782, pp. 160–170, 2016.
DOI: 10.1007/978-3-319-41649-6_16

chaining, and interval handling), projection and eternalization, anticipation, and attentional control via a budget based approach.

2 Memory

This section describes the architecture of the memory module, how NAL grammar statements form a 'Belief Network' and the interdependence of the budget. The memory module supports three primary operations: firstly, to return the best ranked belief or goal for inference within concepts (local inference), secondly, to provide a pair of contextually relevant and semantically related statements for inference between concepts (general inference), and finally, to add statements to memory whilst maintaining the space constraints on the system.



The working process of NARS can be considered as unbounded repetitions of an inference cycle that consists of the following sequence of steps [9]:

1. get a concept from memory
2. get a task and belief related to the selected concept
3. derive new tasks from the selected task and belief and put them into buffer
4. put the involved items back into the corresponding bags
5. put the new tasks into the corresponding bags after processing from buffer

NARS utilises elements of metadata (Budget and Stamp) that serve several purposes: they prevent certain forms of invalid inference such as double counting evidence and cyclic reasoning, abstract temporal requirements away from the Narsese grammar, and provide certain implementation efficiencies.

Budget, considered as metadata for the purpose of this paper, determines the allocation of system resources (time and space) and is defined as $(p, d, q) \in [0, 1] \times (0, 1) \times [0, 1]$.

Also, each statement in NARS has a Stamp defined as $(id, t_{cr}, t_{oc}, C, E) \in \mathbb{N} \times \mathbb{N} \times \mathbb{N} \times \mathcal{P}(\mathbb{N})$ where id represents a unique ID, t_{cr} a creation time (in inference

cycles), t_{oc} an occurrence time (in inference cycles), C a syntactic complexity (the number of subterms in the associated term) and E an evidential set.
Curve Bag is a data structure that supports a probabilistic selection according to the item priority distribution. The priority value p of the items in the bag maps to their access frequency by a predefined monotonically increasing function. This data structure is called "Curve Bag" since it allows us to define a custom curve which is highly flexible and allows emotional parameters and introspective operators to have influence on this selection. The remaining factors of Budget, the d durability and q quality parameter, get their meaning from the forgetting function: Whenever an item is selected from the bag, its priority will be decreased according to d and q, namely with $q_r = q * r$, $d_p = p - q_r$ (r being a system parameter) the new priority is then: $p' = q_r + p * d^{\frac{1}{H*p}}$ if $d_p > 0$, otherwise $p' = q_r$, where H is a forgetting rate system parameter. This ensures that forgetting does not cause priority to decrease below quality, after re-scaling by r.

The memory consists of a Curve Bag of Concepts, where Concepts are containers for: Tasklink and Termlink Curve Bags, along with belief and goal tables. The belief and goal tables are ranked tables (Sect. 8, sub-section Ranking). A concept, named by a term, combines the beliefs and goals with this term in it, and is connected to other concepts which share a common sub-term or super-term via Termlinks.

3 Logic Module

The logic module is an instantiation of the Non-Axiomatic Logic (NAL). It is composed of two components: an inference rule domain specific language (Meta Rule DSL) and an inference rule execution unit. The meta rule DSL should not be confused with the NAL grammar rules, these are separate and distinct. The system currently implements 200+ inference rules, containing forward and backward rules for reasoning under uncertainty.

Meta Rule DSL. The meta Rule DSL was developed to serve three main purposes: to provide a flexible methodology to quickly experiment with alternate inference rules, to support the goal of creating a literate program, and to substantially improve the quality of the software implementation.

Inference rules take the following form:

$$T, B, P_1, ..., P_k \vdash (C_1, ..., C_n)$$

where T represents the first premise (precondition, corresponding to the task to be processed), B represents the second premise (precondition, corresponding to the belief retrieved for the task), and $P_1, ..., P_k$ are additional preconditions which represent logical predicates dependent on T, B, $C1, ..., C_n$. Each "conclusion" (or postcondition) C_i of $C_1, .., C_n$ has the form (D_i, M_i) where D_i represents the term of the derived task the conclusion C_i defines, and M_i provides additional meta-information, such as which truth function will be used to decide the truth or desire of the conclusion, how the temporal information

will be processed, or whether backwards inference is allowed. The DSL incorporates the Narsese grammar to retain consistency of syntax and conciseness of representation.

Inference Rule Execution. The role of the inference Rule Execution unit is twofold: firstly, to parse the Meta Rule DSL into an efficient and executable representation, and secondly, to select and execute the relevant inference rules. An optimised Trie is used to store the rule representation, whilst a Trie Deriver is used to select and 'execute' the relevant inference rules.

Trie Representation - In the Meta Rule DSL, each inference rule has a set of preconditions. These preconditions are stored as nodes in the Trie, where common preconditions form a common node (as with the Rete algorithm [3]). This leads to a natural structuring of the conditions where non-leaf nodes store the preconditions and leaf nodes form sets of post conditions that represent valid derivations for a pair of input statements.

Trie Deriver - The deriver is responsible for inference: it matches from memory selected premise pairs to the relevant inference rules such that conclusions can be obtained. The matching of rules to statements is simply a matter of traversing the Trie, keyed on the matching preconditions. If the traversal ends at a leaf node then this is a valid matching inference rule(s), leaf nodes can contain more then one inference rule. Each traversal, if valid, returns a list of postconditions/conclusions of the matched rules.

Since the complexity of statements is bounded due to AIKR, and the depth of this trie is bounded by the finiteness of the inference rules, applying the Trie Deriver to a pair of statements is upper bounded in execution time by a constant. This is an important consideration as NARS needs to respond to tasks in real-time, whereby, no single inference step can exceed a roughly constant time.

4 Temporal Inference Control

An adaptive agent existing in a real-time environment needs to be capable of reasoning about time. To support reasoning with time the non-temporal NAL inference rules are extended by adding temporal variants. Temporal inference is distinguished by several features: utilisation of a Temporal Window, Temporal Chaining, and Interval Handling, along with Projection, Eternalization and Anticipation, discussed in the following sections.

Temporal Window - As argued in [2], human beings have the ability to synchronize multiple stimulus events, when they are experienced within a Temporal Window of roughly 80 ms, as if they were experienced concurrently. These so called subjective events behave like a point in time as well as an interval in time [7]. A similar approach is used in NARS where a $DURATION$ parameter defines the temporal window of synchronization, whereby, events occurring within the Temporal Window will be deemed to have occurred concurrently.

Temporal Chaining - Due to the AIKR, NARS does not allow arbitrary temporal relations to be formed, in fact the inference execution unit will only allow

semantically related statements (those which correspond to concepts which are connected with each other via termlinks) to be used in derivations. This leads to the question of, how do we temporally relate semantically unrelated events?

The approach taken in NARS is to perform inference between each incoming event with the previous incoming event in order to create compound events which link the (previously semantically unrelated) events together. Although perception can form more complex temporal compound events than this, the same principle applies.

These compound events can then be used by the inference system with other semantically related statements to form further derivations. In this way, complex chains of temporal reasoning can be formed as also demanded for perception.

Interval Handling - When an event a (for example, "wheel starts turning") enters the system, its occurrence time is recorded, but its duration is not known at this time. Even without a duration, the event a can still be related to previous events as the occurrence time is available.

If eventually an event b (for example, "wheel stops turning") enters the system, the system can derive an event (a, I, b) which has a custom duration and encodes "the wheel was turning from this time to this time", which behaves essentially as an interval in interval algebra as a special case. However this interval number I raises another question: To what extent does the duration of an event, i.e. the interval number I, affect how the statement should be observed? We took the approach, to assume similar scales, based on the scale of the interval, would be considered as similar observations. For example the interval of 1 s and 1.2 s will be observed as the same, similarly with 1 h and 1.2 h. If there is need for a further distinction, a clock operator can provide the system with additional context.

The *Duration* time window provides a tolerance that allows the system to observe re-occurring patterns in time, coming from asynchronous input channels, which would otherwise be seen as different if the specific events are incoming in a different order, albeit, in millisecond scale.

5 Projection and Eternalization

When two semantically related statements are selected for inference, (if not stated otherwise by the specific rule) it is necessary to map the occurrence time of the belief to the occurrence time of the task, using the current time as reference.

This mapping function is called "projection" and describes how the truth value of a statement decreases when projected to another occurrence time. In this operation, the confidence of the belief is decreased by a factor

$$k_c = \frac{|t_B - t_T|}{|t_B - t_C| + |t_T - t_C|}$$

where t_B is the original occurrence time of the belief, t_T the occurrence time it is projected to, and t_C the current time. The new confidence of the belief is then

$$c_{new} = (1 - k_c) * c_{old}$$

Eternalization is a special form of induction, where the occurrence time is dropped, so the conclusion is about the general situation. The eternalized confidence value is obtained with

$$c_{eternal} = \frac{1}{k + c_{temporal}}$$

where k is a global evidential horizon personality parameter [9].

In inference, whenever an event is derived, the eternalized version is also derived. However the existence of eternal statements presents a problem: How to justify inference between two premises, about different times? In order to deal with this scenario, there are two possible routes: the inference rule is a temporal rule which measures the time between the premises, and takes it into account when its conclusion is built, or, one of the following cases applies:

1. *Premise*1 is eternal, and *premise*2 is temporal. Here, *premise*2 is eternalized before applying inference.
2. *Premise*1 is temporal, and *premise*2 is eternal. Since *premise*2 is eternal, it also holds at the occurrence time of premise1, so inference can occur directly.
3. *Premise*1 is temporal, *premise*2 is temporal. In this case *premise*2 is projected to the occurrence time of *premise*1, and also eternalized. Inference now happens between *premise*1 and the stronger in confidence outcome, either the result of the projection or the result of eternalization.
4. Both are eternal, in which case the derivation can happen directly.

In all the cases, the occurrence time of the first premise (usually the task), is assigned to the occurrence time of the derived task, and possibly a statement-dependent time-shift as specified by some temporal inference rules, dependent on the term encoded intervals, which measure time between events, is applied.

6 Anticipation

In NARS predictive statements usually take the form:

$$antecedent \Rightarrow consequent$$

where observing *antecedent* leads to the derived event *consequent*, on which the system can form an expectation on whether it will be observed as predicted, this is called Anticipation. With Anticipation the system is able to find negative evidence for previously learned predictive beliefs which generate wrong predictions. [6,10]

If the event happens, in the sense that a new input event with the same term as the anticipated event is observed, the anticipation was successful (confirmation), in which case nothing special needs to be done, since the statement will be confirmed via the normal process of temporal induction.

If the predicted event does not happen then the system needs to recognise this. This is achieved by introducing a negative input event, $not(a)$. Note that in

this case, such a negative input event has high budget and significantly influences the attention of the system.

Anticipation introduces three challenges: firstly, how to ensure that the system doesn't confirm its own predictions? secondly, how to ensure that the system only anticipates events which are observable and hence overcome the issue that negative events are generated for events which are not observable? and thirdly, how to deal with occurrence time tolerance as well as tolerance in truth value.

The first is handled by letting only input events (not derived events) confirm a prediction. The second shows that the closed-world-assumption (CWA) is not applicable in general, just because something isn't observed doesn't mean it didn't happen in general. This issue is overcome by letting only those predictions, which correspond to observable concepts, generate anticipations. When a new input event enters the system, the corresponding concept is marked observable, in this way the observability of concepts is tracked. Regarding the third issue, currently the system assumes that the event does not happen if it doesn't occur within time $[t_{oc} - \frac{|t_{oc}-t_{cur}|}{u}, t_{oc} + \frac{|t_{oc}-t_{cur}|}{u}]$ where t_{oc} is the occurrence time of the anticipated event, and t_{cur} is the current time, with u usually being set to 2. To allow tolerance in truth, anticipation as well as the confirmation currently uses tasks with frequency greater than a threshold, by default 0.5, this tolerance handling method may be refined in the future and is still in discussion. Also note that by this treatment, conceptual events like "Our team wins the football match" have to be decomposed down to directly observable events by inference. Whether and how this can be improved, is also still in discussion.

7 Evidence Tracking

One of the most important notions in NARS is the idea of evidence, note that the truth value of a statement is essentially a (w_+, w_-) pair, where w_+ represents positive evidence, and w_- represents negative evidence, or alternatively as confidence c and frequency f tuple, where $f = \frac{w_+}{w_+ + w_-}$ and confidence is $c = \frac{w_+ + w_-}{k + w_+ + w_-}$, where k is a global personality parameter that indicates a global evidential horizon. For full details on truth value derivations see [9]. Evidence in NARS follows these principles:

1. Evidence can only be used once for each statement.
2. A record of evidence used in each derivation must be maintained, although given AIKR (as also assumed in [6]), this is only a partial record, which is not an issue in practice.
3. There can be positive and negative evidence for the same statement.
4. Evidence is not only the key factor to determine truth, but also the key to judge the independence of the premises in a step of inference.

As described previously, each statement has a stamp which contains an evidence set, E. Following each derivation, a new E is created, by interleaving the two evidence sets of the premises, which is then truncated to a maximum

length by removing the oldest evidence. Interleaving the evidence sets is important and ensures an even distribution of evidence from both evidence sets. The evidence set, E, initially contains the unique statement *id* from the stamp. Prior to derivation, evidence sets of the involved premises are checked for intersection, if they intersect then there is overlapping evidence between the premises and no derivation is allowed (as this would double count evidence).

8 Processing of New and Derived Tasks

This step consists of processing new inputs and derivations selected from the buffer Curve Bag, by applying temporal chaining for new input events followed by ranking based selection for local inference. Here the Revision Rule is applied to belief and goal tasks, and the Choice Rule is applied to question and goal tasks. Additionally the Decision Rule is applied to a goal task [9].

Temporal Chaining - As discussed in Anticipation, it is important to distinguish between new inputs and derivations, because only new input events invoke Temporal Chaining. When a new input event enters the system, inference is automatically triggered with the previous new input event [10], generating a temporal derivation (Sect. 4, sub-section Temporal Chaining).

Ranking - Belief and Goal tables are ordered according to a ranking function, where the confidence is determined after projecting each new belief or goal, to the target time. When the ranking is done for selective questions [9], the function is e/C where e is the expectation value of the statement and C its complexity. In all other cases the confidence of the statement is used for ranking.

Adding to Belief/Desire Table - Once the ranking of a new belief or goal is determined, this ranking specifies the position of the entry in the table. If the table is full, then the lowest ranked entry is deleted to maintain the maximum capacity limit.

Selecting Belief for Inference - When a belief from the belief table is taken out after the selection of the task for inference (Sect. 9, sub-section Phase 2), it is done so by ranking all entries in the belief table according to the occurrence time of the task. The best entry is selected for inference. This also holds for local inference, where a new incoming belief task selects the best candidate to revise with. The new belief and the revised one are then added as described in the previous section. If the task is a question or goal, the new belief overwrites its best solution, dependent on whether it is higher ranked according to the ranking function as described in Ranking.

Revision - When a belief or goal task is processed (selected as task in an inference cycle), it is projected to the current time. Now the highest ranked entry in the belief / goal table in respect to the current moment is determined. When the task is able to revise with this one, this is done and we are finished. If the task is a goal, the Decision rule is also applied:

Decision - If the goal task is an operation (which is an event the system can trigger itself) the desire value expectation, measured with $expectation(x) = (c * (f - \frac{1}{2}) + \frac{1}{2})$ (with f being the frequency, c the confidence) of the highest ranked desire is determined and if it exceeds a certain threshold, the system executes this operation. After the execution, an event, stating that this operation was executed, is input into the system. This event is then available for use in temporal chaining supporting learning about the consequences of the systems own operations in different contexts.

9 Attentional Control

The attentional control stage is primarily concerned with managing the Attentional Focus of NARS. This is achieved with a three phase process of: selecting contextually relevant and semantically related tasks for inference, creating or updating budget values based on user requirements and/or inference results, and finally, updating memory with the results of the updated task and concepts.

Phase 1: Premises for inference are selected according to the following scheme:

1. Select a concept from memory (according to Curve Bag semantics).
2. Select a tasklink (with related task) from this concept.
3. Select a termlink from this concept.
4. Select a belief from the concept the termlink points to, ranked by the task.

Phase 2: This phase forms new statements (tasks), with new metadata, from the derivations. The task linked by the tasklink used in inference determines the statement type and the occurrence time of the new task (unless the inference rule states otherwise, which may also shift the occurrence time). The Budget of a new task is defined as (where T is the truth value of the new task and $h \in [0, 1]$ is a personality parameter giving high quality to tasks of high frequency):

priority: $or(priority(tasklink), priority(termlink))$
durability: $durability(tasklink) * durability(termlink) * \frac{1}{C}$
quality: $max(expectation(T), (1 - expectation(T)) * h) * \frac{1}{C}$

where, for quality, $\frac{1}{C}$ is applied for backward inference, and $or(a, b) = 1 - ((1 - a) * (1 - b))$ [1]. This budget is also used for the tasklink created for the new task. Next the termlinks are strengthened by the derivation. Here Hebb's rule is used: $priority(termlink)' = or(priority(termlink), or(quality, and(a, b)))$ where a is the concept priority referred by the tasklink, and b is the concept priority referred by the termlink. Additionally, the durability of the termlink is also increased: $durability(termlink)' = or(durability(termlink), quality)$.

Phase 3: Select new tasks from the buffer Curve Bag, process them, and insert them into memory:

1. If Concept C_T does not exist, where T is the task, create it and any other required concepts to match the sub-terms of the task, along with the necessary termlinks.

   Finally, the concept, containing T, is activated by adding the priority of the task to the concept priority, and using the maximum of the task and concept duration as the new concept duration as well as the maximum of derived task and concept quality as the new concept quality. In this way concepts activate each other context-sensitively and in a directed manner.
2. Construct a tasklink with the budget of T for this task and add it to C_T (note that the task will additionally also be linked from an in inference by the termlink selected subterm concept).
3. Add the task to its statement type related table in C_T.
4. Insert C_T, and sub-term concepts, if any, into the concept bag (memory).

10 Conclusions

The current OpenNARS implementation, described by this document, follows a unified principle of cognition whereby reasoning is carried out within an inference cycle.

To our knowledge, OpenNARS is the only implementation of an AGI system that captures perception, reasoning, prediction, planning and decision making with a single unified principle. In particular we believe the handling of temporal inference, as described in this paper, is a new approach and demonstrates many of the aspects required for an agent to learn and act within a real-time environment.

Although it is difficult for this implementation to be compared to other systems, there are certain aspects that make NARS similar to some other AGI projects, such as AERA [6], OpenCog [4] and SOAR [5], though detailed comparison with them is beyond the scope of this paper.

OpenNARS continues to be a research platform with different aspects of the design at varying levels of maturity. The logic prior to the introduction of temporal logic is considered stable. Temporal logic, introspection and the budget updating functions are work in progress and are not considered optimal at this stage. Perception, introspective mental operators and emotional attentional control are the focus for the next phase of our research. The attentional control is not currently sufficient to handle a high bandwidth perception stream.

The current implementation, OpenNARS v1.7.0, is available for download at: http://opennars.github.io/opennars. The download package contains examples of learning by experience, and demonstrations of the aforementioned cognitive functions as well as practical use cases for the system.

References

1. Bonissone, P.: Summarizing and propogating uncertain information with triangular norms. Int. J. Approximate Reasoning **1**, 71–101 (1987)
2. Eagleman, D.M., Sejnowski, T.J.: Motion integration and postdiction in visual awareness. Science **287**, 2036–2038 (2000)
3. Forgy, C.L.: Rete: a fast algorithm for the many pattern/many object match problem. Artif. Intell. **19**(1), 17–37 (1982)
4. Goertzel, B., Pennachin, C., Geisweiller, N.: Engineering General Intelligence, Part 1 and 2. Springer, Heidelberg (2014)
5. Laird, J.: The Soar Cognitive Architecture. MIT Press, Cambridge (2012)
6. Eric, N., Thórisson, K.R., Dindo, H., Pezzulo, G., Rodriguez, M., Corbato, C., Steunebrink, B., Ognibene, D., Chella, A., Schmidhuber, J., Sanz, R., Helgason, H.P.: Autocatalytic Endogenous Reflective Architecture, Accepted May 13, 2013. http://hdl.handle.net/1946/15083. ISSN 1670-5777. Published: April 2013
7. Pöppel, E., Bao, Y.: Temporal Windows as a Bridge from Objective to Subjective Time, The Philosophy, Psychology, and Neuroscience of Temporality. The MIT Press (2014)
8. Wang, P.: Rigid Flexibility - The Logic of Intelligence. Springer, Heidelberg (2006)
9. Wang, P.: Non-Axiomatic Logic: A Model of Intelligent Reasoning. World Scientific, Singapore (2013)
10. Wang, P., Hammer, P.: Issues in temporal and causal inference. In: Bieger, J., Goertzel, B., Potapov, A. (eds.) AGI 2015. LNCS, vol. 9205, pp. 208–217. Springer, Heidelberg (2015)