OpenNARS Demonstration of Autonomous Learning and Decision-Making

Patrick Hammer[1], **Tony Lofthouse**[2], **Pei Wang**[3]
[1] Institute for Software Technology
Graz University of Technology, Inffeldgasse 16b/II, 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

1 Introduction

NARS is a general-purpose reasoning system that is designed to be adaptive and capable of working with insufficient knowledge and resources. This, together with its ability to deal with uncertainty, qualifies it to be used as a core component of autonomous systems. Such systems often face novel situations they were not prepared for. Hence, a certain capability to adapt and improvise can be beneficial. NARS is capable of both, autonomous learning, and decision-making without requiring supervision. The purpose of this demo is to show these capabilities using our latest NARS implementation, OpenNARS v1.6.5. The examples we introduce involve unsupervised categorization and recognition of patterns (see [8]) in sensory channels, knowledge acquisition and procedural learning based on a stream of experience, and goal-oriented behaviours based on learned knowledge. Also, aspects of the implications of autonomous decision-making under the Assumption of Insufficient Knowledge and Resources (AIKR), which is also described in [7], are demonstrated.

2 System Description

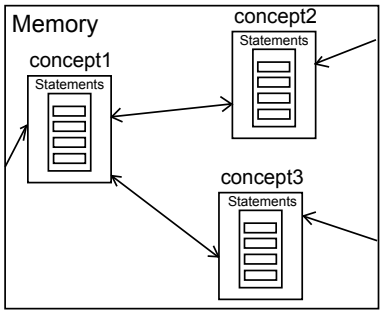
The system's experience can be interpreted as a stream of sentences that enter the system at specific times, which we call events. Over the lifetime of the system, an unspecified amount of such sentences enters the system. Considering a set of events, different patterns (both spatial and temporal) can be extracted and predictive statements derived. Hereby, the extraction of patterns happens through inference, by combining two statements and generating one or multiple resulting statements, depending on the amount of inference rules that apply to the structure of the premises. Statements themselves can be about anything: from a single observed event to an abstract property or pattern. Inference in NARS is not only responsible for deducing what follows from hypotheses, but also to build them in the first place. In the systems memory, all such statements end up in the related container we call "concept", and these concepts are then what the system keeps from its experience, they become the storage elements that summarize what the system experienced. Due to the assumption of insufficient knowledge and resources, only a subset of concepts can be retained. Determining which concepts to retain and which to remove is the role of the control system and forms part of the attention mechanism as also described in [4] and [2].

2.1 Memory

Memory in NARS describes the data structure within the system in which statements are stored. It supports both, the mentioned "concept" containers as well as a forgetting mechanism to obey AIKR.

Concepts

NARS' concept-centric memory model works this way: it assigns a container, called Concept, to each term in the system. All statements are stored in the concept their term refers to. Additionally, concepts are linked to other concepts based on the sub- and super-terms of the term they are referred by. Through this, statements in the system have their individual Semantic Neighbourhood, which is illustrated by the following picture:



Forgetting

Constant memory capacity can only be maintained when statements, and concepts are removed from memory to make space for new data items (Absolute Forgetting). This is realized by removing the lowest priority item when the maximum capacity is reached. Here, the priority of statements starts with an initial value, and concept priority partly reflects the priorities of the statements that are stored in it. Note that the initial priority value is decided by different criteria, such as the total amount of evidence w that is summarized by the

statement, the premise statement priorities and the syntactical complexity of the statement. Additionally the priority of all data items is decaying over time (Relative Forgetting), by their Durability value as described in [2].

2.2 Control Mechanism

The control mechanism describes the strategy according to which two statements are selected for matching against inference rule preconditions to potentially produce new conclusions.

Probabilistic Inference Control

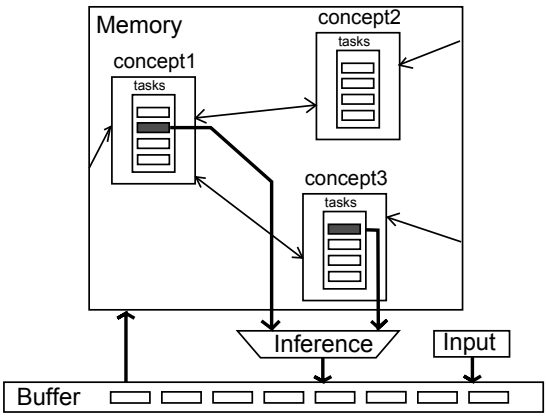
In NARS, data item selection is done in a probabilistic manner. Elements of higher priority have higher chance to be selected, and additionally new statements increase the priorities of the concept they are stored in. Both together leads to the effect that re-observed patterns in experience can not only stabilize in memory, but also will be favourably selected in the future in similar situations. This is a property whose importance for perception and cognition in general is also argued in [3].

Working Cycle

Each inference step consists of:

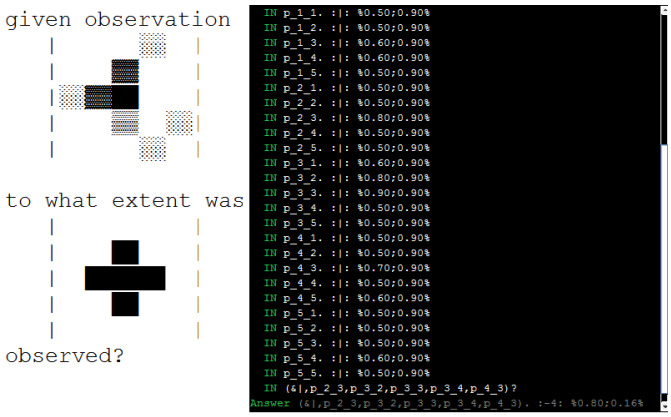
1. Probabilistically select a concept C from memory.
2. Probabilistically select a first premise T from it.
3. Probabilistically select a concept D in the Semantic Neighbourhood of C.
4. Select the second premise B of highest confidence c from D.
5. Apply inference with T and B as premises, generating result statements.

In total, we can now see the entire picture of the system, with having the working cycle selecting two premises from two adjacent concepts (semantic neighbours), and applying NAL inference rules (see [5]) that match the structure of the premises. The results of this derivation are then fed back into a buffer, and processed thereafter, ending in existing concepts, or in new ones, depending on whether a concept for the derived statement term already exists.



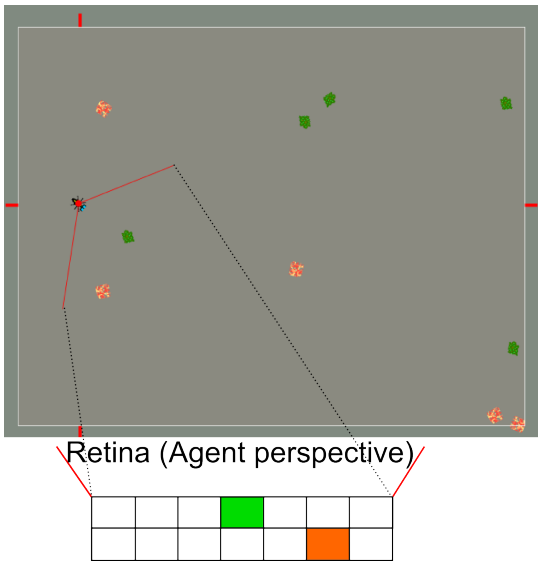
3 Pattern Recognition under AIKR

If an agent is to act in an environment it must be able to sense that environment. The following provides a brief explanation of how perception is encoded within the simulation. Given a sequence of events e_i with their specific truth values, the question arises: to what extent is a certain composition of these events observed? The following figure shows two patterns: an observed pattern and a matched pattern. The screenshot shows one possible encoding of these patterns in Narsese (the OpenNARS input language). Perceived patterns are often incomplete or uncertain and can represent different things. Hence, one challenge for the system is to determine to what extent the observed pattern matches known patterns, in the systems experience. Since not all possible combinations can be stored and exhaustively matched, there are many factors that determine how much resources are spent on specific patterns: Is it a frequently occurring pattern, how important is the related concept, is it related to the fulfilment of a goal, is it temporally relevant, etc.



3.1 Microworld

Microworld is a simulation with multi-dimensional retina input. In this simulation, the agent is required to catch green objects in a 2D world. It should avoid red objects entirely by seeing connections between what it observes, the actions it takes, and the fulfilment of its goals. The only goal here is to catch green objects while avoiding red objects. Whenever a collision with a green object happens, positive reward is obtained, and for collisions with red objects, negative reward. Here, the system demonstrates its ability to work with variable time pressure, and has to deal with different situations in a case-by-case manner (see [6]), largely based on learned and constantly updated procedural knowledge.



Problem Encoding

Here we use a 1-dimensional vision array, where pixel input is given by $(\{pixel_i\} \rightarrow [on])$, which corresponds to "pixel i is on". The reward representation is achieved with goal event $(SELF \rightarrow [rewarded])$, but with having a negative belief event, $\neg(SELF \rightarrow [rewarded])$ as input when a collision with a red object happens, while the event enters without negation on collision with a green object. The agent is automatically moving forward in each step, however can steer by the two operations $((*, SELF) \rightarrow \hat{}left)$ and $((*, SELF) \rightarrow \hat{}right)$. The semantics of these statements intuitively match our intended usage, but can be read in more detail in [5].

Results

Green objects preferred?

Trials	Time per Trial	$\theta = \frac{g}{r}$ on average g... Green objects eaten r... Red objects eaten
30	15 Minutes	1.63

The simulation ran with 19 frames per second on average, leading to 5 seconds for the agent to move from top to bottom and about 7 seconds from left to right. The objects were randomly re-placed whenever they collided with the agent.

Interpretation

In this example, the amount of green and red objects, spawning at random positions, is equal, and also their sizes are equal. To get a feeling for θ: a random moving agent would have θ converge to 1.0 as the amount of red and green objects it collides with will even out in the long run. With $\theta = 1.63$, NARS however demonstrates a strong tendency towards colliding with green objects within just fifteen minutes of learning time, throughout the 30 attempted trials.

4 Discussion

This demonstration shows the capability of OpenNARS to act as an autonomous learning and decision-making system. The Microworld example shows many of the capabilities required by an autonomous system such as: perception, reasoning, decision-making and goal-orientated behaviour. Whilst the Microworld simulation is simple in nature, the challenges can be scaled to more complex environments using the ability of OpenNARS to deal with uncertain and contradictory information, along with the ability to work under the assumption of insufficient knowledge and resources. It also shows how OpenNARS can act as an autonomous system with goal-oriented behaviour driven by user supplied intrinsic goals. Whilst this is a proof of concept for an autonomous system design our belief is that 'safe' AGI is developed through an education process rather than solely providing intrinsic goals or drives [1]. This demonstration is run using OpenNARS version 1.6.5. This is a development branch where the improvements of other versions have been retrospectively added, also featuring self-control abilities as described in [9]. It is single threaded, except for the GUI, and has better inference performance than the other branches. Additionally, using multiple instances of OpenNARS appears to be a more effective way to scale within a distributed environment rather than using a distributed graph. Research in this area is ongoing.

5 Acknowledgements

The authors thank the anonymous reviewers for their helpful comments and suggestions.

References

[1] Bieger J, Thórisson K, Wang P: Safe Baby AGI, Proceedings AGI-15 (2015)

[2] Hammer P, Lofthouse T, Wang P: The OpenNARS Implementation of the Non-Axiomatic Reasoning System, Proceedings AGI-16 (2016)

[3] Hofstadter D., The parallel terraced scan: an optimization for an agent-oriented architecture, Intelligent Processing Systems, 1997. ICIPS '97. 1997 IEEE

[4] Wang P: Rigid Flexibility – The Logic of Intelligence, Springer (2006)

[5] Wang P: Non-Axiomatic Logic: A Model of Intelligent Reasoning, World Scientific, Singapore (2013)

[6] Wang P: Solving a Problem With or Without a Program, Journal of Artificial General Intelligence 3(3):43-73 (2013)

[7] Wang P, Hammer P: Assumptions of Decision-Making Models in AGI, Proceedings AGI-15 (2015)

[8] Wang P, Li X: Different Conceptions of Learning: Function Approximation vs Self-Organisation, Proceedings AGI-16 (2016)

[9] Wang P, Li X, Hammer P: Self-Awareness and Self-Control in NARS, Proceedings AGI-17 (2017)