2024 IEEE International Conference on Robotics and Automation (ICRA 2024)
May 13-17, 2024. Yokohama, Japan

Non-Axiomatic Reasoning for an Autonomous Mobile Robot

Patrick Hammer, Peter Isaev, Lei Feng, Robert Johansson, Jana Tumova

Abstract— **We present the integration of a Non-Axiomatic Reasoning System (NARS) with mobile robots for planning and decision making. NARS enables robots to effectively handle uncertainty in real-time with complete sensor and actuator integration, thereby ensuring adaptability to evolving scenarios. We discuss essential parts of the logic, the architecture and working principles of NARS, and the integration of NARS as a ROS node. A case study is provided demonstrating the system's proficiency to carry out a garbage collection task in an open-air environment by operating a mobile robot with manipulator arm, and we demonstrate its ability to learn about the place-dependent accumulation of garbage items. Case study also reveals that our approach performs more effectively on the overall task than the Belief-Desire-Intention model we compared with.**

I. INTRODUCTION

In the rapidly evolving field of robotics, the development of autonomous systems has become a focal point of research to allow for the automation of crucial tasks within society that are currently performed manually. This includes not only inspection tasks but also tasks which demand some form of manipulator such as garbage collection, which is labor-intensive and critical for maintaining cleanliness in urban environments. To address such challenges, we introduce a novel approach by employing a Non-Axiomatic Reasoning System (NARS) for planning and decision making of a mobile robot with manipulator arm. (illustrated in Fig. 1).

NARS is an advanced reasoning system that handles uncertainty and learns cause-effect relations from observation in real-time, rather than solely using a predefined set of rules, making it a valuable asset for applications and domains requiring adaptability and flexibility. In the paper, we delve into the essential components of the logic behind NARS, as well as the key working mechanisms permitting NARS-based robots to perform their task efficiently and robustly.

Our contribution is an integration of NARS into Robot Operating System (ROS) for mobile robots which we provide as a portable open-source package with a comprehensive set of sensory information encoders and an interface for the reasoning system to instruct motor actions. Hereby we will answer the question whether NARS is a suitable reasoning technique for autonomous ROS-based mobile robots and if its distinguished features such as real-time learning and uncertainty handling can be beneficial for such applications.

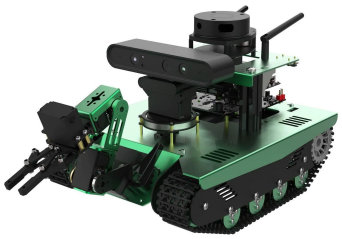

Fig. 1. Yahboom Transbot

To illustrate the effectiveness of our solution, we present a case study conducted in the city of Aarhus, Denmark, using Yahboom Transbot (Fig. 1). We experimentally demonstrate the system's ability to successfully carry out a garbage collection task in an open environment, and more reliably than a Belief-Desire-Intention model. Furthermore we show the systems's capacity to learn about the expected accumulation of trash objects at different places. This adaptability is a cornerstone of our system, as it enables the robot to respond to changes in the environment and to deal with unexpected disturbances which can appear at runtime.

II. RELATED WORKS

The strength of practical means-end reasoning lies in the ability to effectively utilize high-level knowledge for planning purposes, which is easily expressible and communicable by humans. However, available Practical Reasoning solutions with logic derived from First Order Predicate Logic, such as [1]–[4], lack structure learning and handling of uncertainty which could enhance the adaptability of task planners [5]. This also includes non-monotonic reasoning systems [6] which, albeit may retract existing hypotheses when facing contradicting information, are not robust in settings with inherent uncertainties where even the best hypotheses sometimes are contradicted by new data. There also has been a work regarding structure learning from data [7], such as the learning of new cause-effect relations using Schema Network [8] and Inductive Logic Programming (ILP) [9]. Nevertheless, both schemas and induced FOPL predicates demand to be consistent with all previously observed events making them inapplicable in non-deterministic and complex environments with inherent uncertainties, an issue also faced with description logics [10]. While probabilistic extensions

*This research has been carried out as part of the Vinnova Competence Center for Trustworthy Edge Computing Systems and Applications at KTH Royal Institute of Technology.

Patrick Hammer and Jana Tumova are with Division of Robotics, Perception and Learning, KTH Royal Institute of Technology, 114 28 Stockholm, Sweden pahammer@kth.se, tumova@kth.se. Peter Isaev is with the Department of Computer and Information Systems, Temple University, Philadelphia, PA 19122, USA peter.isaev@temple.edu. Lei Feng is with the Unit of Mechatronics and Embedded Control Systems, KTH Royal Institute of Technology, SE-100 44 Stockholm, Sweden lfeng@kth.se. Robert Johansson is with Department of Psychology, Stockholm University, 114 19 Stockholm, Sweden robert.johansson@psychology.su.se. The authors are also affiliated with Digital Futures.

of ILP have been created, solutions such as [11] are not suitable for real-time application on mobile robots.

Other common systems operating with fuzzy logic [12] or known probability values, such as Probabilistic STRIPS extensions [13] that utilize Probabilistic Planning Domain Definition Language (PPDDL [14]) take uncertainties into consideration, but they depend on both pre-defined causal structure and value assignments [15]. There also has been works [16] to utilize [17] or update probability values in Partially Observable Markov Decision Processes [18], but these lack structure learning, i.e. learning of new causal relationships. Following these approaches, the sample space sizes on which the probability evaluations are based are not explicitly tracked, which, besides having to adhere to the axioms of probability spaces, makes incremental updates difficult. In contrast, NARS provides a way to efficiently support structure learning with local memory updates, and perform incremental evidence updates of existing hypotheses, whereby memory capacity constraints are respected [19].

Our continued work [20] is largely motivated by the fact that this type of reasoning system can enhance a robot's ability to handle uncertainty, including "unknown unknowns".

III. Non-Axiomatic Reasoning for Robotics

NARS consists of a logic and control mechanism with associated memory structure. In this section, we recall the most relevant definitions and concepts to allow the reader to understand the principles of NARS.

A. Non-Axiomatic Logic

Non-Axiomatic Logic (NAL) [21] specifies which information can be derived from combinations of at most two premises and supports uncertainty estimation. It incorporates inference rules for learning from event streams, goal reasoning and decision making, as well as functions to measure the uncertainty of statements based on evidential support. Hence, one can build a means-end reasoner able to learn from experience using NAL.

Definition 1. Truth Value *in NAL is defined based on positive evidence* $w^+ \in \mathbb{R}$ *and negative evidence* $w^- \in \mathbb{R}$ *which supports or contradicts a statement (also referred to as belief or hypothesis), and the total evidence* $w := w^+ + w^-$*, each of which is zero or greater. Based on these evidence values, the* truth value *and* truth expectation *in NAL are defined from the frequency and confidence tuple* (f, c)*:*

$$f := \frac{w^+}{w} \in [0,1], c := \frac{w}{w+1} \in [0,1)$$

$$expectation(f,c) := (c * (f - \frac{1}{2}) + \frac{1}{2})$$

Definition 2. Inheritance Statement $< A \rightarrow B >$ *indicates* A *to be a special case of* B*, where* A *and* B *are terms corresponding to unique identifiers.*

For instance, "robots are machines" may be encoded as an inheritance statement $< robot \rightarrow machine >$. Terms can also be marked as instances and properties denoted by $\{instance\}$ and $[property]$ respectively. That way, for example, the statement "Henry being green in color" can be expressed as $< \{henry\} \rightarrow [green] >$. Furthermore, relations can be expressed via product terms $(a * b)$, which allow to express relationships such as "a bottle is left of a cup" as $< (bottle * cup) \rightarrow left >$. Such statements are similar to predicates in predicate logic [21].

Definition 3. Events *are time-stamped statements with a time-dependent truth value, where the confidence decreases with increasing time distance between the premises the inference is being done with (Projection, see [21]).*

When two events as premises are used in inference, the confidence of the second premise is discounted by the factor $\beta^{|\Delta t|}$ with $\Delta t = time(B) - time(A)$, where β is the truth projection decay, a hyperparameter. Also, multiple events form sequences considering their temporal order, whereby sequence (A, B) denotes B happened after A.

Definition 4. Temporal Implications *are statements of the form* $(A \Rightarrow B)$*, while* Procedural Implications *have shape* $((A, Op) \Rightarrow B)$*, where* A *and* B *are statements from events, and* Op *is an operation term referring to a procedure, typically motor-related, the system can invoke at any time.*

The former denotes that B will happen after A, and the latter that B will happen when Op is executed right after A has happened. For example, $(switchOn \Rightarrow lightOn)$ denotes that light is on after the switch is activated.

To calculate the truth values of implications, one needs to define calculations for w^+ and w^- [21]. Considering binary events for hypothesis $(A \Rightarrow B)$, w^+ is the amount of cases in which A happened and B happened after it, and w^- is the amount of cases where A happened but B did not happen thereafter. Slightly more complex but following the same idea, given $((A, Op) \Rightarrow B)$, w^+ is the amount of cases in which A happened, Op was executed and B happened after it, while w^- is the amount of cases where A happened and Op was executed but B did not happen thereafter. Then, using Projection, implications are formed via the Induction rule

$$\{A, B\} \vdash (A \Rightarrow B)$$

with the truth of the conclusion being computed using truth function for induction:

$$f_{ind}((f_A, c_A), (f_B, c_B)) = (f_B, \frac{f_A * c_A * c_B}{f_A * c_A * c_B + 1})$$

whereby the confidence decay (projecting event A to event B) is subsequently applied to the conclusion based on Δt.

B. Inference Control Mechanism

Inference control is concerned with which premises to pick and which inference rules to apply on a cyclic basis.

To control the process of forming temporal and procedural implications from input events (to calculate their evidence), a sliding-window is utilized (a first-in-first-out buffer, a common approach in Data Stream Mining [22]) which only holds the latest k events. By iterating this structure, both the positive and the negative cases can be identified (for more details, see [23]), where the Induction rule is applied

in each case. Hereby, if an induced temporal implication is a duplicate (already existed in memory), its truth value is revised by applying the Revision Rule adding up the evidences of the new and old implications: $w^+ = w_1^+ + w_2^+$, $w^- = w_1^- + w_2^-$. This ensures implications receive additional positive or negative evidence dependent on whether the consequent will happen when the antecedent has happened.

Definition 5. *Goals G are represented as temporal implication $(G \Rightarrow D)$ where D stands for "desired state", and their desire value is defined as the truth value of this implication.*

In each system cycle, one goal (if existent) is selected to either trigger a decision or lead to the derivations of subgoals. For this purpose, the existing procedural implications of the form $((A_i, Op_j) \Rightarrow G)$ are iterated over. When they have a sufficiently high truth value, and event A_i recently happened, it will generate a high desire value for the reasoner to execute Op_j. The desire expectations of these operations are compared, and the operation from the candidate which leads to the highest desire value expectation will be executed if above the decision threshold (a hyperparameter). Alternatively, all the preconditions (all A_i) of the considered implications will be derived as subgoals, competing for attention and processing in a bounded priority queue ranked by desire value expectation. The desire value of the subgoal is obtained using the deduction inference rule between the implication and the goal [21]. To determine the operation's desire value, one additional deduction step is required to take the precondition truth value into account. This corresponds to the following inference rule:

$$\{(X \Rightarrow G), (G \Rightarrow D)\} \vdash (X \Rightarrow D)$$

in case X being of the form (Y, Op) the rule becomes:

$$\{((A, Op) \Rightarrow D), A\} \vdash (Op \Rightarrow D)$$

which means when this temporal implication is present, the system desires to execute Op only after A happens.

The desire values of subgoals are the following respectively:

$$desire(A) = f_{ded}(desire(G), truth(((A, Op) \Rightarrow G))$$

for the subgoal which corresponds to the antecedent of the implication, and

$$desire(Op) = f_{ded}(desire((A, Op)), truth(A))$$

for the operation subgoal to potentially execute if A has happened, with f_{ded} being (as in [21]):

$$f_{ded}((f_1, c_1), (f_2, c_2)) = (f_1 * f_2, f_1 * f_2 * c_1 * c_2)$$

Using this model, decision making is concerned with realizing a goal by executing an operation which most likely and sufficiently likely leads to its fulfillment. When there is no such candidate to accomplish in a single step, subgoals are derived from which a candidate might fulfill this requirement or again lead to further subgoaling. This is similar to backward planning from a goal to current circumstances, while preferring to process more attainable goals by taking uncertainties of events and implications into consideration. Alg. 1 summarizes goal processing.

Algorithm 1: Decision and subgoaling

Input: Goal G
Result: Execute Op or subgoaling
subgoals = {}, bestDesire = 0.0
forall $((X, Op) \Rightarrow G) \in memory$ **do**
  $subgoals = subgoals \cup \{X\}$
  if $desire(Op) >$ *bestDesire* **then**
    bestDesire = $desire(Op)$, bestOp = Op
  end
end
if $bestDesire >$ *DECISION_THRESHOLD* **then**
  execute($bestOp$)
else
  forall $s \in subgoals$ **do**
    derive s (for potential next step selection)
  end
end

IV. ROS INTEGRATION

To make NARS easily portable on mobile robots, we provide an open-source software package which embeds the reasoner into ROS Melodic, where more widely useful nodes for mobile robots have been integrated and connected to the reasoning node. This includes standard ROS nodes for Simultaneous Localization and Mapping (SLAM), the navigation stack, nodes for visual object detection, and related encoders to map the pre-processed information into a format the reasoner can work with. In Fig. 3 we see the channel nodes providing perceptual input to the NARS reasoner from the various sensors illustrated in Fig. 2.

- **Vision:** A broadly applicable vision channel that can interface convolutional neural network models for object detection such as provided by darknet_ros with the reasoning system. The vision channel uses discretized relative to the center location information along with the output label to form statements of the form $< objectLabel \rightarrow [direction] >$ similar as in [24] but with the relative location encoding where $position$ for the X-axis is either $left$, $center$ or $right$. This form of encoding makes NARS aware of the detected object types and their position in the camera's field of view.
- **Obstacles:** An event coding providing nearest-obstacle awareness based on laser scan input. Hereby distance and direction to the nearest object within 2 meters of range are determined. This is utilized to make the reasoning system aware of objects to be avoided in close proximity. The event encodings in this case are $< obstacle \rightarrow [direction] >, < obstacle \rightarrow [free] >$ with $direction \in \{left, front, right\}$.
- **Localization:** Encoders from the output of localization estimations such as based on SLAM, as mobile robots often utilize. The estimated location is periodically (every second) sent to the reasoner as a relational statement $< (SELF * location) \rightarrow [at] >$ whereby $location$ encodes the robot's pose (location and rotation).

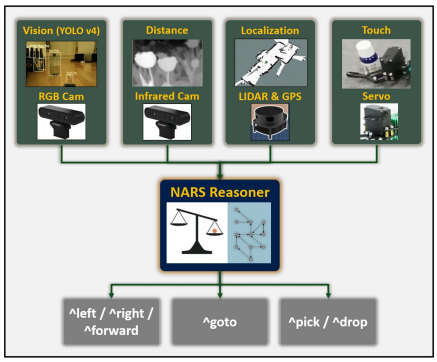

Fig. 2. Multi-modal integration illustration

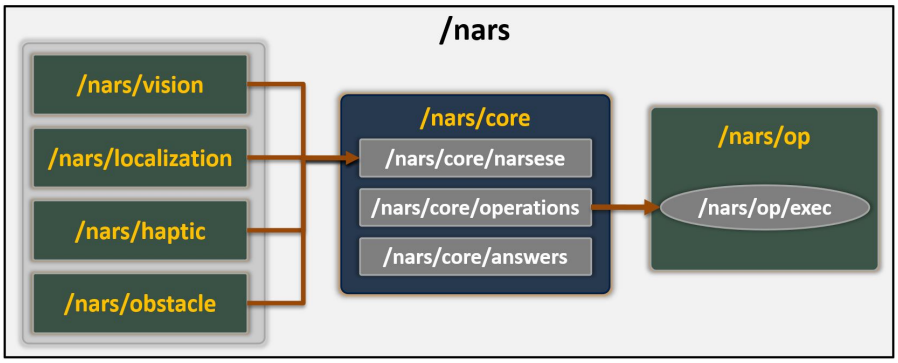

Fig. 3. Corresponding ROS network for NARS

- **Haptic** is responsible for manipulator feedback of a gripper. It indicates, with a Boolean whether the robot has managed to get a hold on an object via an applicable motor procedure. The feedback is encoded as an event $< gripper \rightarrow [holding] >$ or $< gripper \rightarrow [free] >$.
- **Navigation:** NARS is able to send navigation goals to the ROS navigation stack. Thus, the reasoning system can invoke local base movements for moving *^forward*, *^left*, *^right*, and may send goals to reach distant locations on the map obtained from SLAM (via *^goto*).
- **Manipulation:** The system can invoke motor procedures for object manipulation (such as *^pick* and *^drop* which can be manually coded or defined with MoveIt).

The reasoner core node (as in Fig. 2), encapsulating NARS, accepts new inputs as strings formatted in the formal language the reasoner works with, by subscribing to the Narsese topic of the ROS node. To let NARS control the robot, operation executions are redirected to the $Exec$ node for further dispatch. Hereby the ROS navigation stack is utilized for robot base movements, by publishing $move\_base/goal$ desired target location goals (both local-relative, and map-absolute) and canceling the previous ones when new operations are invoked by NARS (via $move\_base/cancel$). Also, the manipulator arm of the robot is controlled by adjusting the angles of the servo motors based on the location of the detected object to pick and its corresponding depth estimate.

V. Case Study: Garbage Collection

The case study presents the testing and optimization of an autonomous crawler robot (Fig. 4) controlled by our NARS ROS package in a real-world environment for the purpose of garbage collection. The chosen test site is located in Aarhus, Denmark and is recognized as one of the largest of its kind for autonomous robots in Europe, offering a diverse range of environments for comprehensive testing. The testbed encompasses a bustling recreational area, a concrete football field, and a park. These diverse settings allowed us to simulate various real-world scenarios that the robot might encounter during its operation. The testbed's design and public utilization closely mimics the typical trash accumulation patterns observed in urban areas, thereby providing a realistic environment for testing the robot's performance.

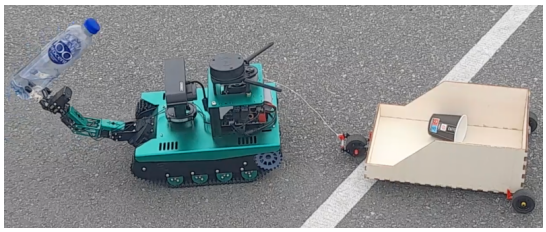

Fig. 4. Garbage collection in Aarhus robotics testbed

A. Integration on Transbot

We demonstrate that NARS ROS package is capable of:

- operating the robot with sub-second perception-reason-action cycle featuring full sensor & actuator integration
- learning and updating hypothesis evidence to deal with failure and learning about the garbage items distribution within the environment
- performing the overall task more effectively than the Belief-Desire-Intention model we compared with
- utilizing high-level knowledge and operator input regarding checkpoints during the reasoning process.

We deliberately utilized Yahboom Transbot (Fig. 1) to integrate and demonstrate the capabilities of our ROS package on low-cost hardware. Transbot is a mobile robot with tracks and a manipulator arm with 3 degrees of freedom. It incorporates the following sensors: an RGB and depth camera (combined, via Astra Pro), a 2D Lidar sensor (RPLidar), and an angle reading sensor for each servo motor of its manipulator. In terms of compute power, Transbot uses a Jetson Nano with an ARM CPU, low-powered NVIDIA Tegra GPU and 4 GBs of RAM.

We chose the following software configuration:

- **Reasoner:** OpeNARS for Applications (ONA) v0.9.2[1] with default parameter configuration.
- **Vision:** YOLOv4 from tensorrt_demos[2] was utilized as object detection model, as it is small enough to be accelerated using the on-board GPU of Jetson Nano, providing low but task-sufficient 4 frames per second.

[1]Repository: https://github.com/opennars/OpenNARS-for-Applications
[2]Repository: https://github.com/jkjung-avt/tensorrt_demos

TABLE I
OBJECT DETECTION PERFORMANCE

Condition	Detection Rate	Time
Evening Sun	61%	4 PM - 6 PM
Cloudy	70%	11 AM - 1 PM
Sunny	83%	11 AM - 1 PM

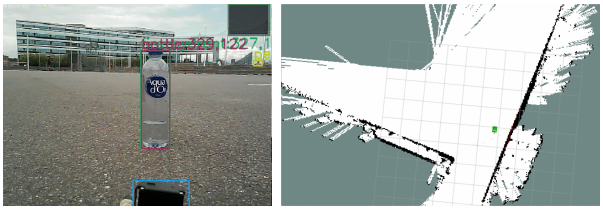

Fig. 5. Visual object detection and corner mapping

- **Haptic:** Since the servo motors of the manipulator arm do not provide a sensor measuring the current, we instead check whether the rate, with which angle of the gripper changes during the relevant part of the pick operation, is below a threshold. This is being used to generate the previously described Boolean flag indicating whether the gripper is holding an object.
- **Manipulation:** For manipulation we created a hand-crafted motor procedure based on proportional control, using the detected bounding box location as control input, as well as the feedback event generated from the haptics handling. While this is clearly not state-of-the-art, it is sufficient for this part of the task.

B. Testing Conditions and Observations

The robot was tested to collect different types of trash objects such as bottles and cups of different materials including plastic, paper and glass. The robot's performance was evaluated under various weather and lighting conditions, including sunny, cloudy, and low lighting conditions. Because the solution consists of multiple components and failure is often caused by one or more of them, we report the success rate of the different components under the different conditions. For reproducibility purposes the final quantitative measures were taken in a 22x22 meters square area within the football field of the testbed. Also the conditions present during testing and their respective times were recorded.

Object detection model: We found the YOLOv4 object detection model of the robot which was pre-trained on ImageNet to be sensitive to the lighting conditions during the day, with the detection performance for bottles and cups decreasing from approximately 83% to 61% in the evening sun. The detection rate was around 70% in cloudy conditions, and optimal lighting conditions were observed between 11 AM and 1 PM under clear skies, as illustrated in Table I. These numbers are reported with a YOLOv4's minimum confidence value of 0.3. Further reduction in the confidence level led to a strong increase of false positive detections, and sometimes to duplicate bounding boxes, while an increase resulted in missing relevant detections more frequently.

SLAM: Localization beyond the Lidar range of 10m led the robot to fully depend on its odometry and IMU. This proved to be sufficient when a corner or significant markings were mapped beforehand and when the Lidar-blind area was only traversed by the robot without any checkpoints in that area which would cause intermediate rotations of the base.

However, extensive navigation with waypoints within such blind spots of the map beyond Lidar range to obstacles resulted in failures of SLAM building a consistent map and in successive localization errors. Therefore, in our setup, we ensured the corners were mapped out before operation and checkpoints were laid out at the boundaries of the testbed so that the robot could traverse the middle of the testbed in a relatively straight line without causing localization issues.

Manipulation: The proportional manipulator control approach exhibited issues on-site that were not observed in the lab in a way that friction on a concrete surface, due to gravel and other factors, behaved more unpredictably. This initially caused bottles to lean over in most cases before the gripper was able to secure them, as the controller approached the target object too aggressively. The issue was resolved by setting smaller constants in our proportional actuator controller, which avoids overshooting the target object when aiming at and grasping. Additionally, outdoor imagery reduced manipulation performance in conditions with strong shadows (evening sun) due to less accurate bounding boxes.

C. Final Test Setup and Results

The final test setup involved the robot collecting trash items in a designated testbed within the football field of the area in sunny conditions. Hereby we followed a repeated procedure that ensures reproducibility between the test runs and software adjustments, as well as replication of the same setup with other techniques for comparison purposes. While the test runs were performed within the public area where quantitative measures were taken, the area was fenced off to avoid any external interference because of safety regulations.

The **Test Procedure** we employed is as follows:

1) The reasoner is given the background knowledge:

```
// Go to next checkpoint to arrive there:
<(<gripper --> [open]> &/
<({SELF} * (next * $P)) --> ^goto>) =/>
 <({SELF} * {$P}) --> at>>.
// If gripper is open and trash is seen,
// pick it up to hold it
<((<trash --> [see]> &/ ^pick) =/>
 <gripper --> [hold]>>.
// If the gripper is holding an object,
// drop it to the trailer
<(<gripper --> [hold]> &/ ^drop) =/> G>.
// bottles and cups are both trash
<bottle --> trash>.
<cup --> trash>.
```

2) The robot was positioned at a starting point (left-upper corner in Fig. 6) within the testbed and sent on an initial control route to generate the initial map which has the corners of the testbed included.
3) Bottles and cups of different sizes (10 objects, within the gripper width limits) and materials (plastic, paper and glass) were placed at the checkpoints in upright position, and the success rate in object detection and manipulator control was recorded during the mission.

TABLE II
OBJECT COLLECTION PERFORMANCE

Method	Success Rate	Runs
NARS	68%	30
BDI	56%	30

Fig. 6. Checkpoint placement (orange) and example trajectory (blue), collected garbage items (green), missed garbage items (red)

4) Checkpoints were established in a circular route through the testbed (as illustrated in Fig. 6), which the robot used as named guidance points to navigate.

While for grasping well-detected objects we observed a success rate of 85%, the overall success rate of finding and collecting the existing trash objects peaked at 68%, resulting in slightly more than two-thirds of the placed objects being collected. This corresponds roughly to the 83% detection rate multiplied by the 85% success rate of the pick operation, minus 2% error introduced by rare occasions of improper decision making by the reasoning system (such as moving on to another checkpoint even though trash was detected).

D. Comparison with BDI model

We compared our solution with a Belief-Desire-Intention (BDI) model as a baseline (via Jason v3.2.0 [1]), for which above background knowledge was directly translated. With the BDI model we only reached 56% success rate on the task, which can largely be explained by a lack of evidence updating for the success of picking up individual objects. In our runs the BDI model running with same knowledge tried to repeatedly re-pick objects it could not pick (such as objects that had been leaned over by the robot, as it still matched the plan precondition) until it did not see an object and moved on to the next checkpoint, while NARS, due to its evidence updating mechanisms, moved on to the next observed object or another checkpoint after repeated failure. In NARS, consistent with the working principles introduced before, this is due to updating the hypothesis for picking up a specific object instance, e.g.

```
<(<{instanceX} --> [see]> &/ ^pick) =/>
 <gripper --> [hold]>>.
```

receives negative evidence when the object instance is seen however the confirmative event indicating the gripper is holding an object is not provided after executing the pick operation. Now, while we are aware that a similar principle of applying success rate counting could have been programmed into a BDI system by extending the knowledge base accordingly, it is important to note that this kind of evidence updating is central to NARS and is useful not just to learn which objects can or cannot be picked up, but also to learn other potentially useful knowledge about the environment and to deal with different kinds of disturbances in that way. In this particular use case, NARS also learned at which checkpoints trash is seen more frequently, which can be used to make the robot first explore areas where trash accumulates more often. In our setting for a particular checkpoint P_i, NARS forms individual hypotheses as:

```
<<({SELF} * {Pi}) --> at> =/> <trash --> [see]>>.
```

with confidence value representing how often the robot has visited checkpoint P_i, and its frequency at which fraction thereof trash has been observed there. While truth values are automatically utilized in decision making using the mechanisms described in earlier sections, it can also be explicitly queried which makes the robot's behavior transparent. For example, the reasoner may be asked whether and where it has seen trash with the question query as follows:

```
<<({SELF} * {?1}) --> at> =/> <trash --> [see]>>?
```

It is important to note, that the above question is not required for the hypothesis formation or its evidence update, but serves only for the user to check the evidence of the statement.

VI. CONCLUSIONS

We have presented the principles of NARS and our integration of the system into ROS which signifies a substantial progression in the integration of autonomous mobile robots with reasoning systems. Furthermore, we have demonstrated the feasibility of our approach in a case study for a garbage collection task where our approach outperformed the BDI model baseline. The case study shows the strengths of NARS technology in being able to use human-interpretable knowledge effectively while also supporting learning by extending knowledge with inductive reasoning and revision mechanism driven by new evidence gained from input events. Most importantly, NARS has not previously been maturely integrated with ROS, and currently, our portable open-source software package connecting the system with standard ROS components and sophisticated models for object detection and related event encodings is ready to provide value to the wider robotics community.

Future work will include improving the components, such as switching to ROS2, to newer object detection models, as well as incorporating more recent methods for actuator control. Additionally, continuing on our results regarding the learning of expected garbage accumulation, we will explore the utility of inductive reasoning and evidence updating for route optimization.

ACKNOWLEDGMENT

We are thankful to Aarhus City Lab for supporting our test application and providing access to the testing facility in Aarhus, Denmark.

REFERENCES

[1] R. H. Bordini and J. F. Hübner, "Bdi agent programming in agentspeak using jason," in *Computational Logic in Multi-Agent Systems: 6th International Workshop, CLIMA VI, London, UK, June 27-29, 2005, Revised Selected and Invited Papers 6*. Springer, 2006, pp. 143–164.

[2] A. Ferrein, G. Steinbauer, and S. Vassos, "Action-based imperative programming with yagi," in *Workshops at the Twenty-Sixth AAAI Conference on Artificial Intelligence*, 2012.

[3] B. Andres, D. Rajaratnam, O. Sabuncu, and T. Schaub, "Integrating asp into ros for reasoning in robots," in *Logic Programming and Nonmonotonic Reasoning: 13th International Conference, LPNMR 2015, Lexington, KY, USA, September 27-30, 2015. Proceedings 13*. Springer, 2015, pp. 69–82.

[4] F. Rovida, M. Crosby, D. Holz, A. S. Polydoros, B. Großmann, R. P. Petrick, and V. Krüger, "Skiros—a skill-based robot control platform on top of ros," *Robot Operating System (ROS) The Complete Reference (Volume 2)*, pp. 121–160, 2017.

[5] H. Li and X. Ding, "Adaptive and intelligent robot task planning for home service: A review," *Engineering Applications of Artificial Intelligence*, vol. 117, p. 105618, 2023.

[6] J.-L. Vilchis-Medina, K. Godary-Déjean, and C. Lesire, "Autonomous decision-making with incomplete information and safety rules based on non-monotonic reasoning," *IEEE Robotics and Automation Letters*, vol. 6, no. 4, pp. 8357–8362, 2021.

[7] M. Diehl, C. Paxton, and K. Ramirez-Amaro, "Automated generation of robotic planning domains from observations," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2021, pp. 6732–6738.

[8] K. Kansky, T. Silver, D. A. Mély, M. Eldawy, M. Lázaro-Gredilla, X. Lou, N. Dorfman, S. Sidor, S. Phoenix, and D. George, "Schema networks: Zero-shot transfer with a generative causal model of intuitive physics," in *International conference on machine learning*. PMLR, 2017, pp. 1809–1818.

[9] S. Karapinar and S. Sariel, "Cognitive robots learning failure contexts through real-world experimentation," *Autonomous Robots*, vol. 39, pp. 469–485, 2015.

[10] M. Tenorth and M. Beetz, "Knowrob—knowledge processing for autonomous personal robots," in *2009 IEEE/RSJ international conference on intelligent robots and systems*. IEEE, 2009, pp. 4261–4266.

[11] F. Riguzzi, E. Bellodi, and R. Zese, "A history of probabilistic inductive logic programming," *Frontiers in Robotics and AI*, vol. 1, p. 6, 2014.

[12] R. Qiu, Z. Ji, A. Noyvirt, A. Soroka, R. Setchi, D. T. Pham, S. Xu, N. Shivarov, L. Pigini, G. Arbeiter, *et al.*, "Towards robust personal assistant robots: Experience gained in the srs project," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2012, pp. 1651–1657.

[13] F. Teichteil-Königsbuch, V. Vidal, and G. Infantes, "Extending classical planning heuristics to probabilistic planning with dead-ends," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 25, no. 1, 2011, pp. 1017–1022.

[14] H. L. Younes and M. L. Littman, "Ppddl1. 0: An extension to pddl for expressing planning domains with probabilistic effects," *Techn. Rep. CMU-CS-04-162*, vol. 2, p. 99, 2004.

[15] T. Hellström, "The relevance of causation in robotics: A review, categorization, and analysis," *Paladyn, Journal of Behavioral Robotics*, vol. 12, no. 1, pp. 238–255, 2021.

[16] S. Zhang, F. S. Bao, and M. Sridharan, "Asp-pomdp: Integrating non-monotonic logical reasoning and probabilistic planning on mobile robots," in *Proceedings of the AAMAS*. Citeseer, 2012, pp. 185–201.

[17] M. Lauri and R. Ritala, "Planning for robotic exploration based on forward simulation," *Robotics and Autonomous Systems*, vol. 83, pp. 15–31, 2016.

[18] M. T. Spaan, "Partially observable markov decision processes," *Reinforcement learning: State-of-the-art*, pp. 387–414, 2012.

[19] P. Wang, "Insufficient knowledge and resources—a biological constraint and its functional implications," in *2009 AAAI Fall Symposium Series*, 2009.

[20] P. Hammer, P. Isaev, T. Lofthouse, and R. Johansson, "Ona for autonomous ros-based robots," in *International Conference on Artificial General Intelligence*. Springer, 2022, pp. 231–242.

[21] P. Wang, *Non-axiomatic logic: A model of intelligent reasoning*. World Scientific, 2013.

[22] S. Pramod and O. Vyas, "Data stream mining: A review on windowing approach," *Global Journal of Computer Science and Technology Software & Data Engineering*, vol. 12, no. 11, pp. 26–30, 2012.

[23] P. Hammer, "Reasoning-learning systems based on non-axiomatic reasoning system theory," in *International Workshop on Self-Supervised Learning*. PMLR, 2022, pp. 89–107.

[24] P. Hammer, T. Lofthouse, E. Fenoglio, H. Latapie, and P. Wang, "A reasoning based model for anomaly detection in the smart city domain," in *Intelligent Systems and Applications: Proceedings of the 2020 Intelligent Systems Conference (IntelliSys) Volume 2*. Springer, 2021, pp. 144–159.