Autonomous Decision-Making With Incomplete Information and Safety Rules Based on Non-Monotonic Reasoning

José-Luis Vilchis Medina, Karen Godary-Dejean, Charles Lesire

Autonomous Decision-Making with Incomplete Information and Safety Rules based on Non-Monotonic Reasoning

José-Luis Vilchis-Medina[1], Karen Godary-Déjean[2], and Charles Lesire[1]

***Abstract*—In this article we propose a decision process integrating Non-Monotonic Reasoning (NMR), embedded in a deliberative architecture. The NMR process uses *Default Logic* to implement *goal reasoning*, managing partially observable or incomplete information, allowing the design of default behaviours completed by the handling of specific situations, in order to manage the current mission objective as well as safety rules. We illustrate our approach through an application of an underwater robot performing a marine biology mission.**

***Index Terms*—AI-based methods, Formal methods in Robotics and Automation, Cognitive control architectures, Marine robotics.**

I. INTRODUCTION

OVER the past decades, autonomous robots have been used in environments where risk is high or access is difficult for humans. However, there is still work to be done when these robots have to integrate automated reasoning: autonomous robots will face unforeseen events (changes in the environment, uncertainty information, failures) and will then have to adapt their behaviour. More specifically, we are interested in giving to these robots *goal reasoning*. Goal reasoning will make the robot able to decide what should be its current objective according to the current situation, in order to ensure both the mission aims and safety constraints. Goal reasoning is a must-have for long-term autonomy in robotics [1].

In this paper, we are more specifically interested in a marine biology application in which an autonomous underwater vehicle must film fishes along specific *transects* [2] defined by marine biologists. To perform a correct transect (w.r.t. the outcomes expected by the biologists), the robot must follow a straight line enforcing some specific constraints [3]. In this paper, we are not interested in the trajectory control of the robot, but rather on the *goal reasoning* process. Depending on the current situation, the goal reasoning process can decide either to fulfil mission objectives, i.e. transects defined by the biologists, or change the objective to respect the constraints and ensure the safety of the robot (e.g. going back to a home point, or urgently surfacing). Due to the intrinsic uncertainty of the robot environment, it is necessary for the goal reasoning process to manage uncertain information. While some automated planning methods allow to manage uncertainty, for instance using contingent approaches [4] or probabilistic planning [5], they manage uncertainties related to the achievement of *one* objective (or optimizing *one* utility function), and do not allow to integrate goal reasoning, i.e. the ability to adapt the current objectives according to the situation.

Such goal reasoning capability generally roots in *Knowledge Reasoning* (KR) approaches. The KR approach that addresses incomplete and contradictory information is called *Non-Monotonic Reasoning* (NMR), in which the reasoning process can make some assumptions, and revise the conclusions according to further observations. In this paper, we are more specifically interested in *Default logic* [6], a non-monotonic logic in which we can *reason by default*, i.e. we can derive consequences only because of lack of evidence of the contrary. Such reasoning is indeed very relevant and flexible to handle autonomous robotic situations, in which we want to specify some *default* behaviours, except when observing specific situations where a specific reasoning should be applied, such as applying safety rules, or changing the current mission goal to adapt to environment changes.

In this paper, we propose a decision process for autonomous systems that uses NMR when incomplete or possibly contradictory information must be considered. The NMR process implements *goal reasoning*, determining which goals are relevant according to the observed situation. Then an automated planning algorithm computes an action plan to achieve this goal. Finally, the NMR process decides what action the robot should actually perform: either the first action of this plan, or another action imposed by a specific rule (typically a safety rule). In Sec. II, we describe related works. Next, we remind the basic concepts of Default Logic. We then present our contribution in Sec. IV. We show some results in Sec. V, and finally conclude.

II. RELATED WORKS

In Robotics, automated reasoning with logical or formal models is often based on techniques like model-checking or temporal logic. However the purpose of model-checking

Manuscript received: February, 24, 2021; Revised May, 29, 2021; Accepted June, 30, 2021.

This paper was recommended for publication by Editor Markus Vincze upon evaluation of the Associate Editor and Reviewers' comments. This work was supported by the I-Site MUSE of the Univ. of Montpellier through the French National Research Agency with reference ANR-16-IDEX-0006.

[1]José-Luis Vilchis-Medina and Charles Lesire are with ONERA/DTIS, University of Toulouse, France {jose.vilchis_medina,charles.lesire}@onera.fr

[2]Karen Godary-Déjean is with LIRMM/EXPLORE, University of Montpellier, France karen.godary-dejean@umontpellier.fr

Digital Object Identifier (DOI): 0000-0001-5724-5920

is generally to verify properties on a discrete-event system model [7], [8]. Other works used Linear Temporal Logic for under-actuated robots planning [9], [10], describing behaviours of motion in a first-order language and using a theorem solver to obtain moves. Nevertheless, these approaches fail to capture the *non-monotonic* properties.

Knowledge Representation and Reasoning (KRR) has proposed languages to reason about actions and changes, which have been the basis of formal theory of actions [11] and dynamic modelling [12], [13]. More recently, a noticeable effort has been made in leveraging KRR processes for robotic applications, more specifically for human-robot interactions, in the KnowRob framework [14], where queries are done to a knowledge base to deduce information about the environment or the robot tasks. However, they still rely on a monotonic reasoning and do not include uncertainty or incompleteness at the logic level.

Robotics applications that integrate NMR generally use Answer Set Programming (ASP) [15]. ASP is a declarative language based on a stable model paradigm. Among other applications, ASP has been used in robotics for human-robot collaboration through dialog to handle underspecification and further support knowledge accumulation [16], or for planning with time-bounded generation of actions [17]. However, ASP generally has difficulties to reason on all classes of stable models related to a program (e.g. to make a choice of a model according to a user-defined criteria or to compare across the models), and as far as we know, no methodology based on ASP has been proposed for goal reasoning in robotics.

However, there are works that strive to solve problems through NMR based on *default reasoning*, e.g., for decision support in naval missions [18], and UAV control [19]. Default logic is indeed a relevant reasoning framework for robotic missions in which we must handle safety rules and unknown environments. However, the latter works use default logic at the control level, without integration of long-term reasoning, nor providing a methodology for applying default logic.

In this context we propose a decision architecture that intensively uses *default reasoning*, to manage partial and/or contradictory observations and safety rules. We claim that default logic is an appropriate formal tool to design the *goal reasoning* that must take place in an autonomous robot, as it allows to define a default behaviour, and then to specialize this default behaviour by specific rules depending on the encountered (partially observed or incomplete) situation. This reasoning is done at a high-level of abstractions.

III. DEFAULT LOGIC

Default logic is one of the best known formalization for commonsense reasoning, introduced by Reiter [6]. This kind of formalization allows to infer arguments based on partial and/or contradictory information as premises. A *default theory* Δ is a pair (D, W), where D is a set of defaults and W a set of formulas in First-Order Logic (FOL). A default $d \in D$ is defined by a quadruplet $X, A(X), B(X), C(X)$, with $X = (x_1, \ldots, x_n)$ a vector of (non-quantified) free variables, and $A(X), B(X), C(X)$ well-formed formulas (wffs) over X, and is represented as:

$$d = \frac{A(X) : B(X)}{C(X)} \tag{1}$$

$A(X)$ are the *prerequisites*, $B(X)$ the *justifications*, $C(X)$ the *consequences*. Intuitively a default means: "if $A(X)$ is true, and there is no evidence that $B(X)$ might be false, then $C(X)$ can be true".

The possible situations that can be derived from a default theory Δ are called *extensions*. An extension E^Δ can be seen as a set of believes of acceptable alternatives according to a theory Δ. Formally, an extension E^Δ is defined as a smallest fixed-point set for which the following property holds: "if d is a default of D, whose the prerequisite is in E^Δ, and the negation of its justification is not in E^Δ, then the consequence of d is in E^Δ", and defined as $E^\Delta = \bigcup_{i=0}^{\infty} E_i$ with [6]:

$$E_0 = W \tag{2}$$

$$\forall i > 0,\ E_{i+1} = Th(E_i) \cup \Big\{ C(X) \mid \frac{A(X) : B(X)}{C(X)} \in D, \quad A(X) \in E_i, \neg B(X) \notin E^\Delta \Big\} \tag{3}$$

where $Th(E_i)$ is the set of *closed wffs* (i.e. with no free variables) that are provable from E_i. However, extensions are difficult to compute in practice since condition $\neg B \notin E^\Delta$ (3) assumes that E^Δ is known, while E^Δ is not yet computed.

Normal default theories [6] is a specific class of default theories in which all defaults have the form $\frac{A(X):C(X)}{C(X)}$, that can be read "if $A(X)$ is true, and there is no evidence that $C(X)$ might be false, then $C(X)$ can be true". The consequence of this formulation is that (3) can be rewritten [6]:

$$\forall i > 0,\ E_{i+1} = \ Th(E_i) \cup \Big\{ C(X) \mid \frac{A(X) : C(X)}{C(X)} \in D, \quad A(X) \in E_i,\ \neg C(X) \notin E_i \Big\} \tag{4}$$

Normal default theories have two main advantages: (1) at least one extension is always guaranteed to exist, and (2) computation of extensions using Horn clauses has a quasi-linear complexity [20], [21].

IV. DECISION-MAKING WITH DEFAULT LOGIC

In order to handle incomplete or contradictory information in the goal reasoning process of an autonomous robot, we have proposed the decision architecture depicted in Fig. 1. This architecture is based on a ***perception - reasoning - action*** scheme, focusing here on the reasoning part.

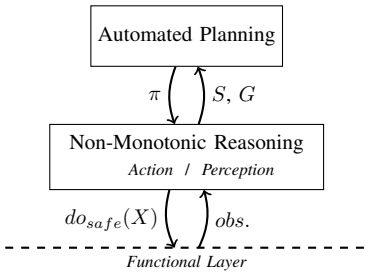


Fig. 1: Decision layer based on NMR.

The NMR is made on some observations (obs) coming from the robot functional layer, being values of the internal states of the system, environment sensing, failures, etc. From these observations, we build and evaluate a default theory $\Delta = (D, W)$, where formulas in D have the form $\frac{A(X):C(X)}{C(X)}$

and formulas in W have the form $A(X) \rightarrow C(X)$. Then an extension of Δ is computed, and two situations may occur:

- the extension contains a do_{safe} statement, which indicates that an action must be executed immediately, as a reactive response to the observations; in that case, the automated planning part is not called and the action is chosen to be executed;
- the extension does not contain a do_{safe} statement, but produces a current situation estimation S and a goal[1] G, which are given to the automated planning module. Then this module provides a plan π which indicates the next action(s) to be executed.

Once an action has been chosen to be executed, we evaluate again this action in Δ. The extension resulting from Δ must then define a do_{safe} statement, that corresponds to the action that will really be executed on the robot. In the following, we first briefly describe how we abstracted the functional layer (observations and actions) through a *skill* formal model, then we describe the design process of the default theory we used in the architecture, with examples from our marine biology application.

A. Functional Layer Abstraction

In order to formalize the interactions with the robot functional layer, we adopted a representation of the robot capabilities and features through a formal *skill model* [22]. The skills represent the actions in our reasoning model, that can be triggered through the do_{safe} predicate. These skill models have an executable semantics, described by required inputs, behaviours, expected outcomes, preconditions, etc. In addition to skills, the model also provides two complementary elements: *resources*, modeled as finite state-machine, and *data*. The skills toolchain includes code generators that provide: (1) a library to interface with the functional layer, in order to retrieve the resource and data values (used as observations for the NMR), and to trigger skill (i.e. action) execution, and (2) a PDDL formalization of skills used by the automated planning algorithm.

B. Non-monotonic Reasoning Model

The first step of the deliberative scheme we propose consist in evaluating observations with respect to the default theory Δ. It has a relevant role because it allows to deduce conclusions from observable (functional layer data and resource states) and/or non-observable information, whose model is defined using defaults. Moreover, in addition to estimating non-observable information, Δ can adapt the mission objective to the current observed situation, either defining a new *goal* for the automated planning process, or directly performing an *emergency* action.

In this paper, we propose some *design patterns*, that can be seen as modeling guidelines to define such a default theory Δ for goal reasoning and safety management of a robotic system. We illustrate these guidelines through examples of logical rules for our marine application.

In order to structure Δ in a design perspective, we breakdown Δ in subsets which are specific to the type of information they deal with. First, Δ must contain a set of facts, summarized in a set W_{obs} (that can come either from *observations* of the functional layer, or be static information about the environment). Δ can also contain formulas that correspond to rules relative to three other classes: *state estimation* in Δ_{est} (that allows to infer values of unobserved states), *goal management* in Δ_{goal} and emergency *safety* rules in Δ_{safety}. Each of these sets is composed of a subset of defaults and a subset of FOL formulas. Thus we have:

$$D = \quad D_{est} \cup D_{goal} \cup D_{safety} \tag{5}$$

$$W = \quad W_{obs} \cup W_{est} \cup W_{goal} \cup W_{safety} \tag{6}$$

Modeling guidelines for each of these sets are given in the following paragraphs.

1) Observed Facts: The propositions that are used in W_{obs} are directly deduced from the skill-based model of the functional layer, that correspond to the data that can be read from this layer, resource states, and skill execution statuses.

Let's look at the model we developed for our marine robot application. W_{obs} contains observed propositions, such as the robot position through the at predicate, the state of sensors, and the status of internal information, such as the precision level of the robot localization. For instance, a typical initial situation in our mission is when our robot is on the surface: so the GPS sensor could be captured, while the USBL (acoustic) sensor could not be. This situation is modeled by the formula:

$$at(home) \wedge \neg usbl\_captured \wedge gps\_captured \wedge on\_surface \tag{7}$$

2) Predicate Estimation: $\Delta_{est} = (D_{est}, W_{est})$ is the part of the default theory that models formulas to infer the truth value of some *hidden* state variables, i.e. that are not directly observed from the functional layer.

Designing formulas in Δ_{est} is very specific to the application. The only guideline that we can enforce is to restrain the C part of the formulas (i.e., the consequences) to only rely on estimated propositions, and never on elements of W_{obs}.

In our application, this typically corresponds to the *localized* predicate, that models the fact that we consider the robot well localized enough to perform a specific task. We want to model the following informal behaviour:

1) we can generally assume that the robot is localized enough to navigate;
2) however, if the USBL signal has not been captured, and neither was the GPS signal, then the robot has never been geo-referenced and we consider that the localization is not good enough.

It results in the following model, where statement 1) is modelled as a default $d_{loc} \in D_{est}$ (8), while statement 2) is an exception to d_{loc}, modelled as classical logical formula $\varphi_{loc} \in W_{est}$ (9).

$$d_{loc} = \frac{\top : localized}{localized} \tag{8}$$

$$\varphi_{loc} = \neg usbl\_was\_captured \wedge \neg gps\_was\_captured \rightarrow \neg localized \tag{9}$$

[1]When several goals are computed in an extension, then a single one is selected thanks to specific rules.

3) Safety rules: Emergency rules consists in situations where we want to directly apply an action instead of relying on an automated planning process. To design such safety rules of our model, we have defined a specific predicate, $do_{safe}(a)$, where a is a possible action of the functional layer. This predicate represents the fact that action a has to be performed as a consequence to the current situation.

In our application, safety rules correspond for example to situations when we observe a critical failure due to safety sensors of the robot (e.g., internal temperature or pressure, water ingress). In that case, we have either to shut down the robot to ensure its integrity, or to immediately surface.

Formulas in Δ_{safety} must then comply with the following guidelines: safety defaults pattern (D_{safety}) are given in eq. (10); FOL formulas patterns (W_{safety}) are given in eq. (11).

$$\frac{A(X) : do_{safe}(a)}{do_{safe}(a)} \tag{10}$$

$$A(X) \rightarrow do_{safe}(a) \text{ or } A(X) \rightarrow \neg do_{safe}(a) \tag{11}$$

where $A(X)$ is a formula over observed or estimated predicates (from W_{obs} and Δ_{est}), and a is an action. Note that formulas in W_{safety} can "cancel" the application of a safety action (when it is negated) as such formulas may correspond to exception to a default safety rule. In our robotic application, the model that corresponds to the loss of a safety sensor is:

$$\frac{safety\_sensor\_failure(X) : do_{safe}(shut\_down())}{do_{safe}(shut\_down())} \tag{12}$$

Note that the possibility to model *defaults* saves us from listing all the safety sensors, leading to a smaller number of rules. As an exception to this, we can model two formulas (one negative and one positive) that express that, for a specific sensor, instead of shutdown, we want to surface.

4) Goal Reasoning: The formulas in Δ_{goal} aim at deducing the current mission objective. To express these formulas, we have introduced a new predicate, $goal(X)$, where X is a formula over the observable propositions, that correspond to elements of the functional layer (e.g., the robot position, the achievement of an action, the state of a sensor). Proposition $goal(X)$ then means that we want X to be achieved, i.e. to be the current mission objective.

Formulas in Δ_{goal} must then comply with the following guidelines: goal reasoning defaults patterns (D_{goal}) are given in eq. (13); FOL formulas patterns (W_{goal}) are given in eq. (14).

$$\frac{A(X) : goal(Y)}{goal(Y)} \text{ or } \frac{A(X) : \neg goal(Y)}{\neg goal(Y)} \tag{13}$$

$$A(X) \rightarrow goal(Y) \text{ or } A(X) \rightarrow \neg goal(Y) \tag{14}$$

where $A(X)$ is a formula over observable or estimated predicates,and Y a formula on observable predicates only. In our application, a goal reasoning can be informally described as: (1) the general mission objective is to perform a transect between p_A and p_B; (2) except if the localisation is too bad to do a transect. This behaviour is modeled through a default and an exception to this default:

$$\frac{\top : goal(transect\_done(p_A, p_B))}{goal(transect\_done(p_A, p_B))} \tag{15}$$

$$\neg localized \rightarrow \neg goal(transect\_done(X, Y)) \tag{16}$$

C. Non-Monotonic Reasoning Process

In the previous paragraphs, we have seen how to model the Default Theory Δ to integrate goal reasoning and safety management. In this paragraph, we will discuss the execution loop of the NMR process. In our application, we have implemented a periodic loop, where, at each period, we apply the steps described below. The decision architecture can then be seen as a *continuous planning* architecture.

1) Observation: First, we get observations from the functional layer, and fill W_{obs} with the observed predicates.

2) Computing extensions: We compute the set of extensions E^{Δ} corresponding to the current default theory.

3) Applying safety actions: If the computed extension has any $do_{safe}(a)$ predicate, then we directly execute the action a (by triggering the corresponding skill in the functional layer), and wait for the next period.

4) Automated planning: If there is no safety action in E^{Δ}, we split E_{Δ} in two sets: the set of goals $G = \{X, s.t.\ goal(X) \in E^{\Delta}\}$, and the set of states $S = E^{\Delta} - G$. We then use an automated planning algorithm to compute the plan π that leads to G from S. In our architecture, we have used the well-known FF algorithm [23]. Note that if G is empty, there is no current goal, and the mission is then finished.

5) Verifying the planned action: Before executing the first action of plan π, we want to ensure that this action is not contradictory with the safety behaviours modeled in the NMR. We then add the first action a_0 of π to Δ, and compute a new extension E^{Δ}_{π}. This step is modeled through a specific predicate, $next\_action(a_0)$, and a default rule:

$$\frac{next\_action(a_0) : do_{safe}(a_0)}{do_{safe}(a_0)} \tag{17}$$

This equation models the fact that, if nothing prevents to execute a_0, then we can execute it. It is then possible to add in the knowledge database an exception to this default. For instance, in our application, we defined the formula:

$$next\_action(X) \wedge \neg enough\_energy(X) \rightarrow do_{safe}(go\_boat) \tag{18}$$

meaning that if the energy is not sufficient to perform the next planned action X, we decide to go back to the boat and abort the mission. E^{Δ}_{π} must then contain a do_{safe} predicate, indicating the action to execute, which will be most of the time the planned action to reach the current goal, except if a safety rule imposed an alternative action.

V. RESULTS

We have implemented the proposed architecture using the ROS2 middleware, with the skill management layer generated from [22], and a specific ROS2 node implementing the decision process in Python/Prolog. The skill model defines 3 data and 9 resources, leading to 12 observable state variables, and 10 skills/actions. Most of the behaviours we have modeled in the default theory have been defined based on a fault analysis of our robot [24]. We have also integrated several goal reasoning complementary behaviours. In the end, our default theory consists of 44 rules, including 17 defaults and 27 exceptions as FOL formulas.

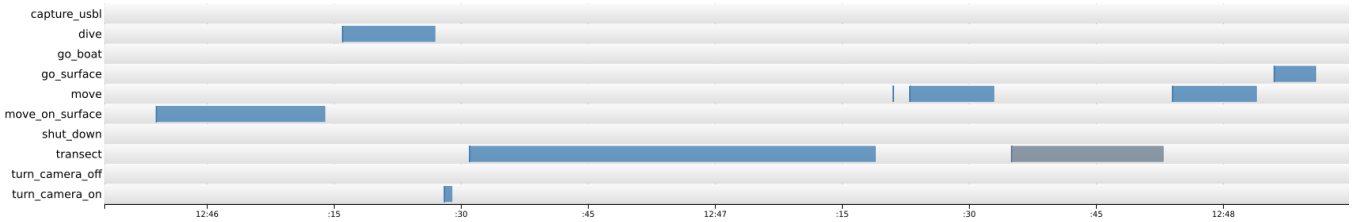


Fig. 2: Timeline of the skill execution. Blue segments indicate successful skill executions, and gray segment shows a skill interruption.

To evaluate our approach, we made a set of simulations, activating the several goal reasoning and safety rules. In this section, we first present a simulation run in order to illustrate the approach, and then report an evaluation of computation times when the size of the models increases.

A. Simulated Scenario

Figure 2 shows the skills executions (from the functional layer perspective) during our simulated scenario. In this scenario, the robot must perform two transects. The successive actions performed by the robot are to move to a first location, then to dive, activate the video camera, and perform the first transect. Then, the robot moves to the start point of the second transect. During the second transect, the localization precision drops under a threshold, leading to the estimation of proposition $\neg localized$. As a consequence, the transect is cancelled, and the robot goes directly to its home point.

In this simulation, the decision architecture ran at a period of 1 second. Figure 3 shows the computation times of the NMR process (to compute extensions) and the FF planning algorithm. The computation times are quite stable all along

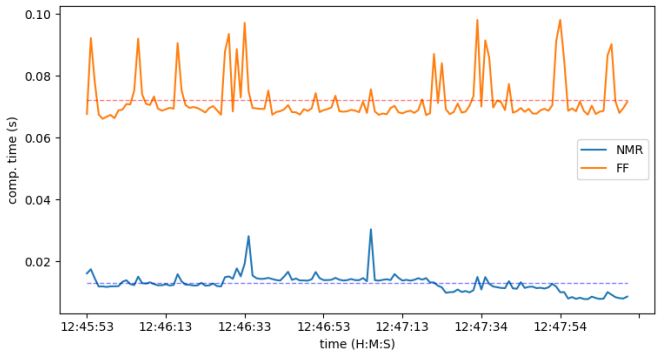


Fig. 3: Evolution of the computation time of the NMR and Planning (FF) processes. The dashed lines indicate their respective mean values.

the mission, and have quite low values, the total computation time being below 0.1 second. Note that the FF time is the time measured when calling FF as an external process, i.e. including file parsing.

B. Computation Time Evaluation

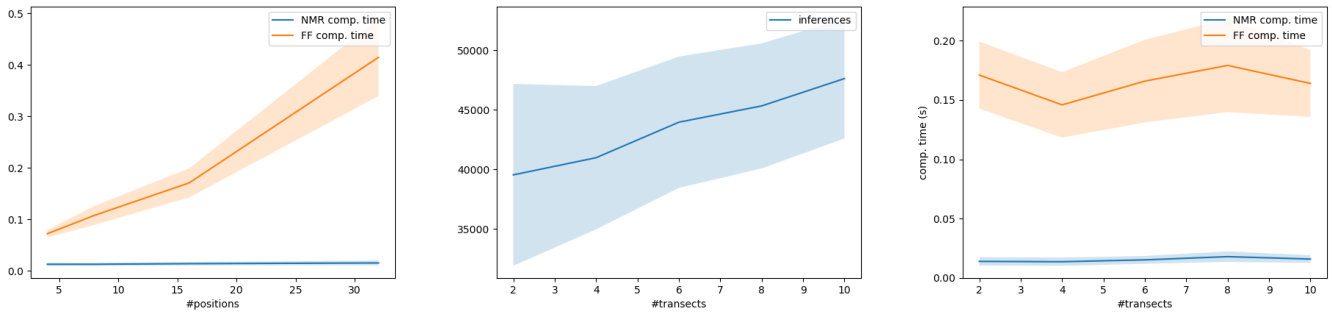
In the previous scenario, the mission objectives defined by the biologists included 2 transects, and the number of possible positions of the robot was restrained to the transects start and end points, as well as the home point and the boat position. While this situation correspond to an actual experiment specified by the marine biologists using our robot, the complexity of the problem is limited. In this section, we evaluated the number of inferences and the computation times when we increase the size of the problem (Fig. 4).

Figure 4a shows the evolution of the computation times when the number of possible positions increases up to 32 positions, which correspond to large problem: in the proposed architecture, we are not interested in trajectory planning, but only on goal reasoning, and the model must then only involve the positions that may have an impact on the mission objectives. We can see that the computation time of FF grows, but the absolute values stay reasonable. The computation time of the NMR is constant, which is expected as there is still only two transects to perform, and the NMR model only relies on the positions attached to the mission objectives. The evolution of the number of inferences (not shown due to lack of place) confirms the constant behaviour of the NMR.

Figure 4b shows the evolution of the number of logical inferences done during the computation of extensions, with respect to the number of mission objectives. In this setup, we fixed the number of positions to 16, and defined from 2 to 10 transects. We can notice that the number of inferences grows linearly, which is consistent with the theoretical complexity of Normal Default Theory [6], [20]. Figure 4c shows the evolution of the computation times. Even if the number of goals increases, we can notice that the computations times are almost static whatever the number of transects.

VI. CONCLUSION

In this paper, we have proposed a new decision architecture based on a *non-monotonic reasoning*, more particularly *default reasoning*, that encompasses goal reasoning and safety management, two major features in long-term autonomy of robotic systems. We presented the concept of the architecture, along with guidelines to model the several behaviours in default logic, relying on specific predicates to manage goals and emergency actions. The main decision-making process first gathers observations from the functional layer, then evaluates the default theory to compute an *extension*. This extension



(a) Comp. times w.r.t. environment size (b) Inferences w.r.t. number of goals (c) Comp. times w.r.t. number of goals

Fig. 4: Evaluation of the NMR architecture processes w.r.t. the size of the model. Plain curves represent the average value. Light areas indicate the standard deviation envelope.

may include a do_{safe} statement, with an action to execute immediately, or a goal state to achieve. In the latter case, we use the FF algorithm to compute a plan of actions, and check the consistency of the first action w.r.t. to the default theory. We have illustrated the approach on a marine biology mission, and presented the results of simulations. This application and the associated results clearly show that the proposed method is a practicable approach to manage safety rules and goal reasoning for autonomous robots. Default logic is indeed very convenient and concise framework to model such behaviours, as it allows to define general defaults rules, and then only specify specific exceptions.

Based on this architecture, future work will address the implementation of an interactive decision process, to allow the biologists to modify the NMR rules *online* while the robot is doing a mission, in order to integrate new behaviours due to not modeled situations.

ACKNOWLEDGMENTS

This work has been partly funded by the I-Site MUSE of the Univ. of Montpellier through ANR (the French National Research Agency) under the "Investissements d'avenir" program (PIA) with reference ANR-16-IDEX-0006.

REFERENCES

[1] F. Ingrand and M. Ghallab, "Deliberation for autonomous robots: A survey," *Artificial Intelligence*, vol. 247, pp. 10–44, 2017.
[2] Z. Thanopoulou, M. Sini, K. Vatikiotis, C. Katsoupis, P. G. Dimitrakopoulos, and S. Katsanevakis, "How many fish? Comparison of two underwater visual sampling methods for monitoring fish communities," *PeerJ*, vol. 6, p. e5066, 2018.
[3] A. Hereau, K. Godary-Dejean, J. Guiochet, C. Robert, T. Claverie, and D. Crestani, "Testing an Underwater Robot Executing Transect Missions in Mayotte," in *TAROS*, Virtual, United Kingdom, 2020.
[4] J. Hoffmann and R. Brafman, "Contingent Planning via Heuristic Forward Search with Implicit Belief States," in *ICAPS*, Monterey, CA, USA, 2005.
[5] J.-A. Delamer, Y. Watanabe, and C. P. C. Chanel, "Safe path planning for UAV urban operation under GNSS signal occlusion risk," *Robotics and Autonomous Systems*, vol. 142, 2021.
[6] R. Reiter, "A logic for default reasoning," *Artificial intelligence*, vol. 13, no. 1-2, pp. 81–132, 1980.
[7] H. Bride, J. S. Dong, R. Green, Z. Hóu, B. P. Mahony, and M. Oxenham, "GRAVITAS: A model checking based planning and goal reasoning framework for autonomous systems," *Eng. Appl. Artif. Intell.*, vol. 97, p. 104091, 2021.
[8] S. Bensalem, K. Havelund, and A. Orlandini, "Verification and validation meet planning and scheduling," *Int. J. on Software Tools for Technology Transfer*, vol. 16, no. 1, pp. 1–12, 2014.
[9] Y. Shoukry, P. Nuzzo, A. Balkan, I. Saha, A. L. Sangiovanni-Vincentelli, S. A. Seshia, G. J. Pappas, and P. Tabuada, "Linear temporal logic motion planning for teams of underactuated robots using satisfiability modulo convex programming," in *CDC*, Melbourne, Australia, 2017.
[10] A. Bolotov, O. Grigoriev, and V. Shangin, "Automated natural deduction for propositional linear-time temporal logic," in *TIME*, Alicante, Spain, 2007.
[11] M. Gelfond and V. Lifschitz, "Action Languages," *Electronic Trans. on Artificial Intelligence*, vol. 2, pp. 193–210, 1998.
[12] H. J. Levesque, R. Reiter, Y. Lespérance, F. Lin, and R. B. Scherl, "GOLOG: A logic programming language for dynamic domains," *The Journal of Logic Programming*, vol. 31, no. 1-3, pp. 59–83, 1997.
[13] Y. Jin and M. Thielscher, "Representing beliefs in the fluent calculus," in *ECAI*, Valencia, Spain, 2004.
[14] M. Tenorth and M. Beetz, "KnowRob: A knowledge processing infrastructure for cognition-enabled robots," *IJRR*, vol. 32, no. 5, pp. 566–590, 2013.
[15] T. Kern, J. Kreijger, and L. Willcocks, "Exploring ASP as sourcing strategy: theoretical perspectives, propositions for practice," *J. of Strategic Information Systems*, vol. 11, no. 2, pp. 153–177, 2002.
[16] X. Chen, J. Ji, J. Jiang, G. Jin, F. Wang, and J. Xie, "Developing high-level cognitive functions for service robots." in *AAMAS*, Toronto, Canada, 2010.
[17] B. Schäpers, T. Niemueller, G. Lakemeyer, M. Gebser, and T. Schaub, "Asp-based time-bounded planning for logistics robots," in *ICAPS*, Delft, The Netherlands, 2018.
[18] I. Toulgoat, P. Siegel, and A. Doncescu, "Modelling of submarine navigation by nonmonotonic logic," in *Int. Conf. on Broadband and Wireless Computing, Communication and Applications*, Washington, DC, USA, 2011.
[19] J. L. V. Medina, P. Siegel, V. Risch, and A. Doncescu, "Intelligent and Adaptive System based on a Non-monotonic Logic for an Autonomous Motor-glider," in *ICARCV*, Singapore, 2018.
[20] V. W. Marek, A. Nerode, and J. B. Remmel, "Complexity of recursive normal default logic," *Fundamenta Informaticae*, vol. 32, no. 2, pp. 139–147, 1997.
[21] E. Dantsin, T. Eiter, G. Gottlob, and A. Voronkov, "Complexity and expressive power of logic programming," *ACM Computing Surveys (CSUR)*, vol. 33, no. 3, pp. 374–425, 2001.
[22] C. Lesire, D. Doose, and C. Grand, "Formalization of robot skills with descriptive and operational models," in *IROS*, Las Vegas, NV, USA (virtual), 2020.
[23] J. Hoffmann and B. Nebel, "The FF planning system: Fast plan generation through heuristic search," *JAIR*, vol. 14, pp. 253–302, 2001.
[24] A. Hereau, K. Godary-Dejean, J. Guiochet, , and D. Crestani, "A Fault Tolerant Control Architecture Based on Fault Trees for an Underwater Robot Executing Transect Missions," in *ICRA*, Xi'an, China, 2021.