Towards Teaching High-Level Behaviors to a Robotic Agent

Philippe Hérail, Arthur Bit-Monnot

Towards Teaching High-Level Behaviors to a Robotic Agent

Philippe Hérail
LAAS-CNRS, Université de Toulouse, INSA, CNRS
Toulouse, France
philippe.herail@laas.fr

Arthur Bit-Monnot
LAAS-CNRS, Université de Toulouse, INSA, CNRS
Toulouse, France
abitmonnot@laas.fr

***Abstract*—Developing robust autonomous agents require complex execution architectures operating at several levels of abstraction in order to keep the acting problem tractable. While there is a growing body of work focused on learning models at the sensory-motor level, the same cannot be said for high-level models enabling deliberative functions. In this paper, we identify the possibilities that can be offered by a learning system integrating human input for learning hierarchical operational models and present a learning algorithm that could provide a missing component for such a system.**

I. Introduction

For achieving complex and high-level tasks, autonomous robots typically need to combine a set of elementary skills, where each skill abstracts over motor primitives for elementary operations, such as picking up something. These elementary skills themselves may have to be slightly different depending on the context (e.g. the movement to open a door depends on the handle type). To alleviate the difficulty of designing such skills by hand, several approaches have focused on learning them [1–5].

The high level tasks require acting deliberately, considering the effects of individual actions on the global activity while unexpected events require flexibility and reactivity. Existing execution systems have addressed these compromise for several decades [6–10], leveraging planning techniques to handle new contexts or new tasks to achieve while still attempting to keep deliberation time reasonable.

However, most execution system still rely on manually written "programs" – called *operational models* – even though they are cumbersome to design, requiring human experts with application specific domain knowledge as well as some familiarity with planning languages. Thanks to the advances in action and activity recognition [11,12], we can envision a human tutor teaching an agent through a small set of demonstrations from which to generalize.

For learning hierarchical operational models, there has been approaches focusing on learning Behavior Trees (BTs) [13,14] and hierarchical policies [15]. However, these approaches are focusing on learning reactive models, providing limited lookahead capabilities compared to hierarchical planning models such as Hierarchical Task Networks (HTNs) [16]. While there has been attempts at learning planning models from demonstrations, many of these approaches are focused on non-hierarchical models, focusing on partial traces observability [17], non-deterministic environments [18,19] or even taking into account noisy observations [20].

In the context of hierarchical models, several approaches have been developed, such as HTN-MAKER for use in fully observable deterministic environments[21], and non-deterministic ones [22]. This technique relies on annotated tasks with pre- and post-conditions to extract sequences allowing to achieve the tasks but is limited in terms of abstraction capabilities. HTNLearn [23] also learns hierarchical models from similarly annotated tasks by converting the learning problem into one of constraint satisfaction, but is limited to deterministic environments.

Recently, unsupervised learning techniques have been used to learn Hierarchical Goal Networks (HGNs) [24]. Approaches have also been developed to learn Combinatory Categorial Grammar (CCG) [25,26], which share a similar structure, mainly in the context of plan recognition.

Most of these methods suffer from some limitation in the expressiveness of the learned models and most importantly, they do not offer a mechanism to reuse previously learned information for subsequent learning.

II. Integrating a Human in the Learning Process

We involve the human in the learning process as a tutor, in charge of designing and carrying out demonstrations of desirable behavior to achieve a set of tasks. In general, we assume that each skill exploited in the tutor's demonstration has a corresponding skill in the agent's capabilities, which may require the tutor to place its demonstration at the appropriate level of abstraction to smooth out any difference between their capabilities (e.g. wheels vs legs).

Once learned, a task could be leveraged in further demonstrations, exploiting this common vocabulary to decouple the learning of high-level behavior from the much more fine-grained details. Indeed, such tasks could then be viewed as a new skill shared by the agent and the tutor, allowing for more efficient demonstrations down the line.

A learner able to incorporate abstract tasks in its demonstrations would be able to incrementally learn complex tasks by reusing previously learned behaviors. This idea is similar

This work has been partially supported by AIPlan4EU, a project funded by EU Horizon 2020 research and innovation program under GA n.101016442.

to the notion of learning from a curriculum instead of arbitrary demonstrations, which has been explored for use in learning hierarchical models [27] and recently HTNs [28].

III. Learning Problem and Definitions

Considering a set of demonstrations from a tutor, we want to learn operational models able to reproduce and generalize the demonstrated behaviors.

A. Inputs to the Learning Problem

We consider as input to our learning problem the set of skills common to the tutor and the agent, each represented as a primitive action a composed of an identifier and a set of parameters, such as $a = \text{action\_name}(\text{arg}_1, \ldots, \text{arg}_n)$. We refer to the set of primitive actions as A.

The tutor and learner also share a vocabulary of non-primitive tasks T_I with known parameters, representing the high-level activities to be demonstrated.

For each task $t_I \in T_I$, the agent is given a set D_{t_I} of demonstration traces from the tutor. Each trace $d \in D_{t_I}$ is an alternating sequence of states and parameterized tasks $\{t_i \mid t_i \in \{T_I \cup A\}\}$ such as $d = \{s_0 \rightarrow t_0 \rightarrow s_1 \rightarrow \cdots \rightarrow t_{n-1} \rightarrow s_n\}$. States are a compact representation of the world at any given time.

B. Operational Model

Operational models represent the knowledge an agent has regarding *how* it may achieve some activity in its environment [29]. We define them as in [30], but will recall some definitions here for simplicity.

We define an operational model O as an HTN-like structure which can be written as a tuple $O = (T, A, M)$ where T is a set of abstract tasks, A a set of primitive actions and M a set of possible methods decomposing the tasks $t \in T$ into subtasks $\{t_d \mid t_d \in \{T \cup A\}\}$. An example of such structure is given in Figure 1. Using such a model, the task $\texttt{serve\_drink}(\text{cup}_1)$ could be decomposed as $\texttt{grab}(\text{cup}_1) \rightarrow \texttt{place}(\text{cup}_1) \rightarrow \texttt{pour\_coffee}(\text{cup}_1)$ while $\texttt{serve\_drink}(\text{cup}_2)$ could be decomposed as $\texttt{grab}(\text{cup}_2) \rightarrow \texttt{place}(\text{cup}_2) \rightarrow \texttt{pour\_orange\_juice}(\text{cup}_2)$.

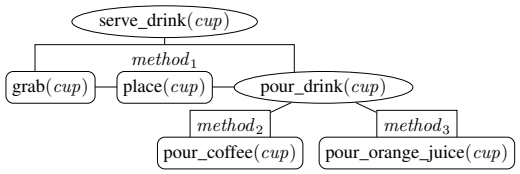

Fig. 1: Example of a simplified operational model structure.

More formally, a method $m \in M_t$ is a tuple $m = (Pre_m, N_m)$, where Pre_m are the preconditions of the method, and N_m is a sequence of tasks and actions defining a way to decompose t. A method is applicable in a given state s iff its preconditions hold in this state.

When considering an acting problem, these models are "executed" to generate a behavior leading to the achievement of a task. In a nutshell, this behavior is generated by systematically replacing a high level task with a method body, until reaching a sequence of primitive actions.

IV. Proposed Approach

In our approach, we focus on developing the learning algorithm which would be part of the learning system described in section II, that is, the generation of complete and efficient operational models from demonstration traces. Building a full pipeline – which would notably require the integration with existing methods for perception, action and intent recognition – is beyond the scope of this paper.

It is based on a learning procedure where an initial model (possibly the result from a previous learning experience) is iteratively refined through the exploration of similar models, extending the work presented in [30].

Each iteration of this procedure starts by generating similar structures from the initial model by adding, removing or reordering branches of the decomposition tree. Parameters are not considered during this first step of the process. Every generated structure is then parameterized, and the model quality is finally evaluated with regard to its ability to "simplify" the given demonstrations.

A. Model Structure Generation

To explore the space of possible model structures, we currently use a descent algorithm, the neighbors of the base model being generated through the changes made to the branches of the decomposition tree. To better escape local minima, we intend to replace the basic descent algorithm with a metaheuristic. In particular, genetic algorithms have been successfully used for learning hierarchical model structures [13,14].

This structure now needs to be parameterized. In the model presented in Figure 1, we would need to know that the parameter *cup* in the action $\texttt{pour\_coffee}(cup)$ is the same as the one in $\texttt{serve\_drink}(cup)$. In addition, we need to establish a correspondence between the parameters appearing in demonstrations (e.g. cup_1) and the ones appearing in the operational model (e.g. *cup*).

B. Model Parameterization

To parameterize the structure, we first consider an initial model where each method has exactly one parameter for each parameter in its underlying sub-actions and sub-tasks. This naïve guess obviously lead to many duplicated parameters that eventually need to be unified. We cast this problem as MAX-SMT [31] with equality logic, under the objective of minimizing the number of parameters in the model and requiring that each demonstration trace remains a possible output of the final operational model.

C. Parameterized Model Quality Metric

To evaluate the quality of the learned model, we use the metric described in [30], based on the Minimum Description Length (MDL) principle [32]. This metric exploits data compression as a way to drive the model search towards abstracting redundant parts in the demonstrations.

REFERENCES

[1] J. Kober and J. Peters, "Learning motor primitives for robotics," in *2009 IEEE International Conference on Robotics and Automation*, May 2009, pp. 2112–2118.

[2] S. Levine, N. Wagener, and P. Abbeel, "Learning Contact-Rich Manipulation Skills with Guided Policy Search," *arXiv:1501.05611 [cs]*, Feb. 2015.

[3] C. Finn, T. Yu, T. Zhang, P. Abbeel, and S. Levine, "One-Shot Visual Imitation Learning via Meta-Learning," *arXiv:1709.04905 [cs]*, Sep. 2017.

[4] L. Johannsmeier, M. Gerchow, and S. Haddadin, "A Framework for Robot Manipulation: Skill Formalism, Meta Learning and Adaptive Control," in *2019 International Conference on Robotics and Automation (ICRA)*, May 2019, pp. 5844–5850.

[5] O. Kroemer, S. Niekum, and G. Konidaris, "A review of robot learning for manipulation: Challenges, representations, and algorithms," *The Journal of Machine Learning Research*, vol. 22, no. 1, pp. 30:1395–30:1476, Jan. 2021.

[6] F. Ingrand, R. Chatila, R. Alami, and F. Robert, "PRS: A high level supervision and control language for autonomous mobile robots," in *Proceedings of IEEE International Conference on Robotics and Automation*, vol. 1, Apr. 1996, pp. 43–49 vol.1.

[7] O. Despouys and F. F. Ingrand, "Propice-Plan: Toward a Unified Framework for Planning and Execution," in *Recent Advances in AI Planning*, ser. Lecture Notes in Computer Science, S. Biundo and M. Fox, Eds. Berlin, Heidelberg: Springer, 2000, pp. 278–293.

[8] S. Sardina, L. de Silva, and L. Padgham, "Hierarchical planning in BDI agent programming languages: A formal approach," in *Proceedings of the Fifth International Joint Conference on Autonomous Agents and Multiagent Systems*, ser. AAMAS '06. New York, NY, USA: Association for Computing Machinery, May 2006, pp. 1001–1008.

[9] A. Mayima, A. Clodic, and R. Alami, "JAHRVIS, a Supervision System for Human-Robot Collaboration," in *2022 31st IEEE International Conference on Robot and Human Interactive Communication (RO-MAN)*, Aug. 2022, pp. 777–784.

[10] J. Turi and A. Bit-Monnot, "Guidance of a Refinement-based Acting Engine with a Hierarchical Temporal Planner," in *ICAPS Workshop on Integrated Planning, Acting, and Execution (IntEx)*, Jun. 2022.

[11] V. Krüger, D. Kragic, A. Ude, and C. Geib, "The meaning of action: A review on action recognition and mapping," *Advanced Robotics*, vol. 21, no. 13, pp. 1473–1501, Jan. 2007.

[12] D. Höller, G. Behnke, P. Bercher, and S. Biundo, "Plan and Goal Recognition as HTN Planning," in *2018 IEEE 30th International Conference on Tools with Artificial Intelligence (ICTAI)*, Nov. 2018, pp. 466–473.

[13] M. Colledanchise, R. Parasuraman, and P. Ögren, "Learning of Behavior Trees for Autonomous Agents," *IEEE Transactions on Games*, vol. 11, no. 2, pp. 183–189, Mar. 2018.

[14] Q. Zhang, J. Yao, Q. Yin, and Y. Zha, "Learning Behavior Trees for Autonomous Agents with Hybrid Constraints Evolution," *Applied Sciences*, vol. 8, no. 7, p. 1077, Jul. 2018.

[15] T. Silver, R. Chitnis, A. Ajay, J. Tenenbaum, and L. P. Kaelbling, "Learning skill hierarchies from predicate descriptions and self-supervision." in *AAAI GenPlan Workshop*, 2020.

[16] K. Erol, J. Hendler, and D. S. Nau, "Complexity results for HTN planning," *Annals of Mathematics and Artificial Intelligence*, vol. 18, no. 1, pp. 69–93, Mar. 1996.

[17] D. Aineto, S. Jiménez Celorrio, and E. Onaindia, "Learning action models with minimal observability," *Artificial Intelligence*, vol. 275, pp. 104–137, Oct. 2019.

[18] H. M. Pasula, L. S. Zettlemoyer, and L. P. Kaelbling, "Learning Symbolic Models of Stochastic Domains," *Journal of Artificial Intelligence Research*, vol. 29, pp. 309–352, Jul. 2007.

[19] B. Juba and R. Stern, "Learning Probably Approximately Complete and Safe Action Models for Stochastic Worlds," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 36, no. 9, pp. 9795–9804, Jun. 2022.

[20] H. H. Zhuo, J. Peng, and S. Kambhampati, "Learning Action Models from Disordered and Noisy Plan Traces," *arXiv:1908.09800 [cs]*, Sep. 2019.

[21] C. Hogg, H. Muñoz-Avila, and U. Kuter, "HTN-MAKER: Learning HTNs with minimal additional knowledge engineering required," in *Proceedings of the 23rd National Conference on Artificial Intelligence - Volume 2*, ser. AAAI'08. Chicago, Illinois: AAAI Press, Jul. 2008, pp. 950–956.

[22] C. Hogg, U. Kuter, and H. Muñoz-Avila, "Learning hierarchical task networks for nondeterministic planning domains," in *IJCAI 2009, Proceedings of the 21st International Joint Conference on Artificial Intelligence, Pasadena, California, USA, July 11-17, 2009*, C. Boutilier, Ed., 2009, pp. 1708–1714.

[23] H. H. Zhuo, H. Muñoz-Avila, and Q. Yang, "Learning hierarchical task network domains from partially observed plan traces," *Artificial Intelligence*, vol. 212, pp. 134–157, Jul. 2014.

[24] M. Fine-Morris, M. W. Floyd, B. Auslander, G. Pennisi, K. Gupta, M. Roberts, J. Heflin, and H. Muñoz-Avila, "Learning decomposition methods with numeric landmarks and numeric preconditions," in *Proceedings of the 5th ICAPS Workshop on Hierarchical Planning (HPlan 2022)*, 2022, pp. 29–37.

[25] C. W. Geib and R. P. Goldman, "Recognizing plans with loops represented in a lexicalized grammar," in *Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2011, San Francisco, California, USA, August 7-11, 2011*, W. Burgard and D. Roth, Eds. AAAI Press, 2011.

[26] P. Kantharaju, S. Ontañón, and C. W. Geib, "Extracting CCGs for plan recognition in RTS games," in *Proceedings of the 2nd Workshop on Knowledge Extraction from Games Co-Located with 33rd AAAI Conference on Artificial Intelligence, KEG@AAAI 2019, Honolulu, Hawaii, January 27th, 2019*, ser. CEUR Workshop Proceedings, M. Guzdial, J. C. Osborn, and S. Snodgrass, Eds., vol. 2313. CEUR-WS.org, 2019, pp. 9–16.

[27] P. Morere, L. Ott, and F. Ramos, "Learning to Plan Hierarchically From Curriculum," *IEEE Robotics and Automation Letters*, vol. 4, no. 3, pp. 2815–2822, Jul. 2019.

[28] R. Li, M. Roberts, M. Fine-Morris, and D. Nau, "Teaching an HTN learner," in *Proceedings of the 5th ICAPS Workshop on Hierarchical Planning (HPlan 2022)*, 2022, pp. 68–72.

[29] M. Ghallab, D. Nau, and P. Traverso, *Automated Planning and Acting*. Cambridge: Cambridge University Press, 2014.

[30] P. Hérail and A. Bit-Monnot, "Learning Operational Models from Demonstrations: Parameterization and Model Quality Evaluation," in *ICAPS Hierarchical Planning Workshop (HPlan)*, Singapore (virtual), Singapore, Jun. 2022.

[31] R. Nieuwenhuis and A. Oliveras, "On SAT Modulo Theories and Optimization Problems," in *Theory and Applications of Satisfiability Testing - SAT 2006*, ser. Lecture Notes in Computer Science, A. Biere and C. P. Gomes, Eds. Berlin, Heidelberg: Springer, 2006, pp. 156–169.

[32] P. Grünwald, "A minimum description length approach to grammar inference," in *Connectionist, Statistical and Symbolic Approaches to Learning for Natural Language Processing*, J. G. Carbonell, J. Siekmann, G. Goos, J. Hartmanis, J. Leeuwen, S. Wermter, E. Riloff, and G. Scheler, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 1996, vol. 1040, pp. 203–216.