A Hierarchical Control Architecture for Robust and Adaptive Collaborative Robot Task Execution

Luke Fraser*, Banafsheh Rekabdar†, Monica Nicolescu‡, Mircea Nicolescu§, and David Feil-Seifer¶
Department of Computer Science and Engineering, University of Nevada, Reno
Email: *fraser@nevada.unr.edu, †brekabdar@nevada.unr.edu, ‡monica@cse.unr.edu, §mircea@cse.unr.edu, ¶dave@cse.unr.edu

I. Introduction

Robot task representation typically involves a sequential set of steps. This methodology does not generalize to complex real-world tasks. Real-world tasks involve multiple paths of execution, where intra-task dependencies allow for different methods to complete the same task. Representing such complex tasks compactly poses a challenge as enumerating all possible paths results in combinatorial growth. As well, complex tasks benefit from the shared work of multiple agents. Tasks representations should provide a mechanism for multiple agents to communicate and self-organize as a team. We propose a control architecture to address these issues. The architecture 1) provides an efficient an compact encoding of complex tasks, 2) enables robots to infer what the other teammates are doing, 3) allows robots to dynamically decide which execution path to follow using an activation spreading mechanism.

II. Related Work

The focus of the the proposed work addresses challenges encountered working with service and assistive robots in real-world environments. Assistive and service robot applications have been investigated: search and rescue [1], space exploration [2], and medicine and health [3], [4]. The control structures presented perform specific tasks in a sequential manor. This methodology does not generalize well to complex tasks. As well performing in real-world environments requires a control architecture that can adaptively react chagnes in the environment. Furthermore such an architecture should extend to multiple agents. Large and complex tasks benefit from multiple agents participating collaboratively. The proposed architecture seamlessly integrates multiple agents in a distributed task representation.

The proposed task representation resembles a hierarchical task network (HTN) [5]. HTNs have been used effectively to encode simple tasks for automated robots [6], [7]. In [6] HTN's are used as a task representation for object manipulation planning. Using HTN's the planner is able to generate sequential plans for the robot to execute. In [7] HTN's are used to build a complete task plan for a given scenario. The HTN encodes several behaviors that are executed sequentially at runtime. The HTN provides a means for fast planning of a given task or subtask.

Acknowledgement: This material is based upon research supported by the Office of Naval Research under Award Number N000141612312.

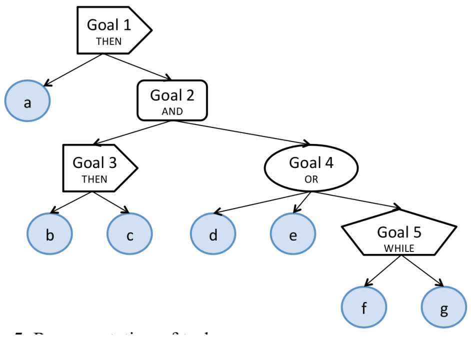

Fig. 1. Representation of task network for expression: "a THEN ((b OR c) AND (d THEN e THEN (f WHILE g)))"

Although, HTN's are similar, the method of task execution is vastly different. The proposed HTN does not separate the planning stage from the execution stage. In our approach both planning and execution are performed concurrently. As well our approach does not expand an HTN at runtime to all possible paths of execution. The proposed distributed HTN architecture allows for real-time planning and execution of a complex task. An activation spreading approach is used to execute and plan a task. As well, the activation spreading framework generalizes to multiple robot tasks.

The following sections will describe the proposed architecture. Section III will provide an overview of the task representation. Section IV will describe the multi-agent execution scheme.

III. Task Representation

We propose a task representation that can encode complex tasks structures and would enable a heterogeneous team of robots and people to self-organize in deciding how to achieve a task's goals. The architecture thus serves multiple roles: i) it provides an efficient, compact encoding of the structure of the task, ii) it enables robots to process which goals other agents are performing, iii) based on the previous information it allows robots to make decisions on what actions to do next and iv) it serves as the control architecture that the robots will use to achieve their goals.

A. Methodology

We propose a representation using a behavior-based paradigm, which provides modularity and ease in communication and connectivity between behaviors. As previously mentioned, our aim is to enable the system to encode tasks that involve temporal sequencing constraints, overlapping temporal constraints (some steps need to happen at the same time, e.g., hold cup while pouring), alternative ways of execution (any one of multiple options is acceptable), and no temporal constraints (some steps can be executed in any order). All these options could be a part of a single task representation, which could be seen from a task such as "a THEN ((b THEN c) AND (d OR e OR (f WHILE g)))". To encode such a task we will define two types of nodes in our behavior network. Behavior nodes encode a basic behavior that achieves a well-defined goal (such as a, b, c, d, e, f, g above). Goal nodes are N-ary trees (i.e., can have from 0 to N children) and encode the four different types of execution constraints mentioned above, as follows:

- THEN goal nodes encode sequencing constraints. For example, GoalSeq = a THEN b, implies that in order to achieve GoalSeq, the system should execute behavior a, followed by behavior b.
- OR goal nodes encode alternate paths of execution. For example GoalAlt = a OR b, implies that in order to achieve GoalAlt, the system can execute either behavior a or behavior b.
- AND goal nodes encode the option of having no ordering constraints. For example GoalNoOrd = a AND b, implies that in order to achieve GoalNoOrd, the system should execute both behaviors a and behavior b, but in any order. This also leads to alternative paths of task execution, but in which all the individual components must be performed at some point.
- WHILE goal nodes encode overlapping temporal constraints. For example GoalTmp = a WHILE b, implies that in order to achieve GoalTmp, the system must execute behavior a while maintaining the goals of behavior b. Implicitly this means that both behaviors a and b might need to be executed concurrently.

With these types of components, the above task will be represented as shown in Figure 1. It can be seen that this representation compactly encodes all the task constraints, including all possible paths of execution. This is especially important when there are multiple alternative paths, such as for Goal2: either goal3 or Goal4 could be performed first; to achieve Goal4 either one of d, e or Goal5 could be executed. Instead of explicitly enumerating all possibilities, we use the most compact form of the task representation. This representation would be more accurately represented by the following prefix encoding: (THEN a, (AND (THEN b c) (OR d e (WHILE f g)))).

Each goal node stores the following information: status (indicates whether the goal has been achieved, not achieved but currently being worked on, or not achieved and not actively being pursued), type (THEN, OR, AND, WHILE), activation level and collaborative type (indicates whether it is a joint goal, requiring or permitting that more than one agent performs the goal, or not). During task execution, each goal node continuously evaluates its status (computed from the status of its children) and this information is available for the higher-level parent nodes.

To perform a task, a robot uses the above task representation as follows: the root node of the task (typically a goal node) evaluates, if all its children goal/behavior nodes are done. If they are all finished, nothing needs to be done. Otherwise, the node will send activation messages to its child nodes in order to signal that they should become active and work. These goal nodes, in turn, send activation messages as needed to their children. Messages are also passed from children nodes that are currently active (basic behaviors that are running, or goal nodes that are active) upwards to the parents to indicate the viability of a given child. This is important for keeping track of which pathways of execution are preferred by parent nodes. This is a key feature that enables seamless self-organization within the robot team (Section IV). The different types of goal nodes send activation messages to their children as follows:

- THEN goal nodes evaluate the status of their children in the order given by the sequence and send activation messages with decreasing magnitudes, from the first goal in the list whose status is not done to the last. This ensures that the subsequent steps of the task are executed in the proper sequential ordering.
- OR goal nodes initially send activation messages to all their children. Once the activation messages reach a leaf node (basic behavior), the behavior with the highest activation level and whose own preconditions are met will start running. This allows for opportunistic task execution, in situations in which environmental conditions are met for just one (or some) of the alternative pathways. Once a pathway becomes active (detected through messages from the children), the goal node will stop sending activation messages to all the other children.
- AND goal nodes send activation messages to all their children. When a particular child node becomes active, the node will increase the activation it is sending to the active child, but continue sending activation to the other children nodes. This would ensure that the system would not oscillate between a child goal node to another and also allow for the possibility that another robot could work on one of the other child goal nodes, which is explained in Section IV.
- WHILE goal nodes first evaluate the status of their second child. If it is achieved, then activation is sent to the first child to begin execution. Otherwise, the node first sends activation to the second child, to ensure that its conditions are met before the first node can start running.

B. Outcomes

This work will develop an architecture that addresses significant challenges that arise in collaborative domains. First,

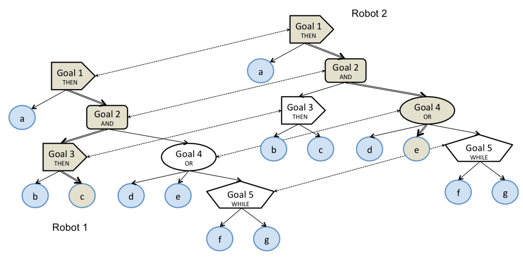

Fig. 2. Communications between nodes from representations on multiple robots. Gray nodes represents the current active path on each robot Connections between basic behaviors are not shown for clarity.

it provides a compact representation of complex tasks with multiple pathways of execution and it serves both as a decision making tool, as well as a control system that is directly employed by the robot for task execution.

IV. Task Execution

The task representation described in Section III enables a single robot to perform a complex task with multiple types of constraints on its own. As a member of a team, however, a robot's decision making needs to take into account its teammates. For this, in addition to the overall goals of the task, the robot needs to take into account which goals are already being worked on by other teammates and also which goals require additional assistance. We describe below how robots can acquire this information and make decisions regarding their own future actions for the benefit of the team.

A. Methodology

We propose that each robot have its own copy of the overall task representation, which uses the top-down and bottom-up activation spreading and message passing mechanism presented in Section III. In addition, each goal or basic behavior node continuously communicates with its identical (peer) nodes on the other robots and sends its current status and an activation message (possibly different from the activation level of the node) (Figure 2):

- If the node status is achieved, this information, along with a zero activation level, will be communicated to all corresponding nodes on the other robots. In this situation, the node will automatically be considered achieved by all the robots, thus preventing any top-down spreading of activation from the completed node (on all other robots) and re-execution of that goal by another teammate.
- If the node status is not achieved, and not active, there is no communication with corresponding nodes on other robots. This implicitly indicates that the goal is "available" for any other teammate to work on. All the robots in the team will thus use the activation levels within their own representations to decide what to work on.
- If the node status is not achieved, but active, this information will be communicated to all corresponding nodes on the other robots. Depending of the type of the node and its class (whether it is a joint goal or not), the other robots will use this information as described next. Goal nodes that are not joint are those that could be performed by a single robot. Therefore, if a robot is currently engaged in working on such a goal, the activation sent to sibling nodes on other robots will be zero, indicating that this goal is currently being tended to. For joint goals, the activation sent to sibling nodes on other robots will be identical to the activation level of the node, indicating to other robots that help is needed with this part of the task. This allows robots to infer when others need assistance and provides them with awareness of the current needs of the robot team.

This process seamlessly enables the self organization of the robot team: as one robot begins working on a part of the overall task, the messages between their representations allow all the robots in the team to know what parts of the task are being worked on by others and what parts are still left to be done. Within each robot's own representation, the activation messages from the un-achieved goals will lead the robot to select the appropriate sub-task to work on. The proposed approach works identically in situations in which the overall team task is composed of multiple root goal nodes, as different teammates can choose to work on different high-level goals.

V. Conclusion

We have proposed an control architecture and task representation that will address many collaborative robotics challenges. The task representation compactly encodes complex tasks. This representation allows for multiple paths of execution and creates a general description of tasks with different constraints. The use of this task representation allows for collaborative teams of robots to work on the same task together. The robots, using the activation spreading mechanism can communicate between active behaviors to allow for seamless and distributed self-organization in real-world environments. With the proposed architecture robots will be capable of working together to complete common goals.

References

[1] R. Murphy, "Human-Robot Interaction in Rescue Robotics," *IEEE Systems, Man and Cybernetics Part C: Applications and Reviews, special issue on Human-Robot Interaction*, vol. 34, no. 2, may 2004.

[2] R. Ambrose, H. Aldridge, R. Burridge, W. Bluethman, M. Diftler, C. Lovchik, D. Magruder, and F. Rehnmark, "ROBONAUT: NASA's Space Humanoid," *IEEE Intelligent Systems Journal*, aug 2000.

[3] H. Krebs, B. Volpe, M. Aisen, and N. Hogan, "Increasing productivity and quality of care: robot-aided neurorehabilitation," *Journal of Rehabilitation Research and Development*, vol. 37, no. 6, 2000.

[4] D. Wilkes, A. Alford, R. Pack, T. Rogers, R. Peters, and K. Kawamura, "Toward socially intelligent service robots," *Applied Artificial Intelligence*, vol. 12, no. 7–8, 1998.

[5] K. Erol, J. A. Hendler, and D. S. Nau, "UMCP: A Sound and Complete Procedure for Hierarchical Task-network Planning." in *AIPS*, vol. 94, 1994, pp. 249–254.

[6] M. Weser and D. Off, "HTN robot planning in partially observable dynamic environments," in *2010 IEEE International Conference on Robotics and Automation*. IEEE, may 2010, pp. 1505–1510. [Online]. Available: http://ieeexplore.ieee.org/articleDetails.jsp?arnumber=5509770

[7] L. Kaelbling and T. Lozano-Pérez, "Hierarchical task and motion planning in the now," *Robotics and Automation*, 2011. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5980391