

# Prioritizing Behaviors in a Fuzzy Behavior Hierarchy

Brent E. Eskridge  
School of Computer Science  
University of Oklahoma  
Norman, Oklahoma 73019-6151, USA  
eskridge@ou.edu

Dean F. Hougen  
School of Computer Science  
University of Oklahoma  
Norman, Oklahoma 73019-6151, USA  
hougen@ou.edu

## ABSTRACT

Multi-agent systems offer capabilities that single agent systems cannot. Chief among these capabilities are robustness, scalability, modularity, and the ability to distribute the agents throughout their environment. Along with these capabilities, however, comes the challenge of coordination the agents so they are able to achieve their goals effectively and efficiently. Our work focuses on the issues involved with the scalability of coordination between both individual agents and teams of agents. We are specifically interested in scaling up multi-agent systems in three ways. The first is to scale up the capabilities of individual agents. When using behavior-based systems, this generally equates to adding new behaviors, often in the form of composite behaviors. The second is to scale up the number of agents in a team. The third is to scale up to having teams of teams, or *meta-teams*, to arbitrary levels.

Having teams of teams is different than simply adding all of the agents to a single, larger team because it allows us to have a team hierarchy. This allows us to distribute subgoals to individual teams so the meta-team can collectively achieve its overall goal. With homogeneous teams, this distributed approach allows us to make coordination and cooperation tractable. With heterogeneous teams, this additionally allows us to assign distinct types of subgoals to appropriate teams.

With a hierarchical approach to teams, we should be able to arbitrarily extend the number of levels present in the hierarchy to such a degree that inter-team coordination is not only possible, but highly desirable. However, when scaling in any of these three ways, the complexity of the resulting interactions can quickly become prohibitive. As a result, we are interested in techniques that are able to mitigate this complexity. Since, at a conceptual level, the problems encountered are very similar for each type of scaling, we believe a parsimonious approach to all three is warranted. Our approach is to abstract sensor information through priorities. We demonstrate this approach using flocking at both the team and meta-team levels.

## I. RELATED WORK

Behavior-based architectures allow an agent to react in real time in dynamic environments [6]. Furthermore, the use of behaviors promotes the decomposition of the agent's functionality into small, encapsulated units. For example, Flocking as a result of the interaction between the simple behaviors **Cohesion**, **Separation**, and **Alignment** was first proposed by Reynolds [1]. The coordination of competing behaviors such as these is an area of much research [7]. Some use the approach of organizing the behaviors into a hierarchy to simplify individual behaviors [8]. Another approach is to use fuzzy logic rule sets for smooth transitions between behaviors and to improve the overall robustness of the system [9].

The approach used here is to combine these techniques into a fuzzy behavior hierarchy [10], [11]. Figure 1 shows one implementation of such a hierarchy, based on the work of Tunstel [11]. Rather than performing behavior arbitration in which a single behavior is selected for control, fuzzy behavior hierarchies typically use behavior fusion. Essentially, this means that more than one behavior participates in the control of the agent and each behavior is individually weighted according to its applicability to the current situation. Individual behaviors are weighted through the use of higher-level behaviors called *composite behaviors*. These behaviors are responsible for coordinating lower-level behaviors through weighting values called *degrees of applicability*. Furthermore, lower-level behaviors can be developed in isolation as they contain is no internal reference to their parent composite behaviors. The hierarchical nature of the set of behaviors, along with this encapsulation, minimizes many of the negative effects encountered when scaling up the number of behaviors.

One of the most significant remaining problems to scaling these fuzzy behavior hierarchies is that composite behaviors still require access to the all of the sensor information provided to lower-level behaviors in order to

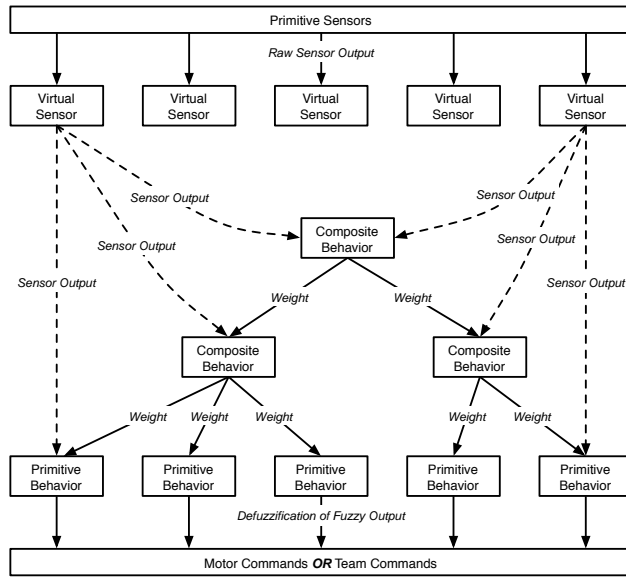


Fig. 1. A sample fuzzy behavior hierarchy, as used by Tunstel [11]. Behaviors are implemented as fuzzy rule sets and output either weighting values for other behaviors or control values to be defuzzified. In our work, these control values can either be motor commands or commands sent to members of a team. For simplicity, not all virtual sensor output links are shown.

weight them appropriately. As a result, the number of inputs to these composite behaviors explode as the hierarchy is scaled. This area of concern has been largely overlooked in previous research. This is most likely due to the fact that most behavior hierarchies described in the literature are relatively shallow.

## II. PRIORITIZING BEHAVIORS

To address this coupling between behaviors and their context, we add an abstraction layer between composite behaviors and the sensor information used in the lower-level behaviors. Therefore, high-level composite behaviors use high-level information to coordinate behaviors, thus reducing the amount and granularity of the information required. A *virtual* sensor is used to not only provide output for use by lower-level behaviors, but to also provide an abstracted view of this output. This virtual sensor is tied to a behavior present in the behavior hierarchy and indicates the current importance of the behavior, given the information available. This *priority* value is then used by higher level behaviors, instead of the normal output provided by the sensor. We see examples of this approach in our everyday life. For example, the engine warning light in a car does not indicate the exact problem, but rather the general fact that attention is required.

Note that although each virtual sensor is capable of sending its output, such as the distance to an obstacle, to its corresponding behavior, this is only necessary for behaviors that directly affect the control commands (i.e., are not composite behaviors). This is due to the fact that the priorities provide enough information for a composite behavior to coordinate the execution of the sub-behaviors. Furthermore, the abstraction of this output means that composite behaviors need not know the nature of that output. For example, fuzzy behaviors can make use of fuzzified sensor output values but, since behaviors are not restricted to fuzzy implementations, there is no limit imposed on the nature or volume of information provided to behaviors by the sensors.

### A. Using Priorities at the Individual Level

Priority is used in two ways. First, the virtual sensor related to a higher-level behavior uses this priority, in conjunction with other priorities, to determine the current importance of the higher-level behavior. Second, the priority is used by composite behaviors in the behavior hierarchy to determine the weighting, or applicability, of the corresponding behavior. For example, if an **Avoid Obstacle** behavior's priority is high, a parent behavior might give greater weight to its output. This weighting is done without the information on *why* the behavior has a high priority. However, it is possible that, despite a high priority, **Avoid Obstacle** might be given a relatively low weight. This is a result of the fact that priority only indicates the importance of a behavior in isolation. This priority is only used as a recommendation and can be ignored if the situation warrants. For example, while a car's engine warning light is generally a cause for concern, if the car is currently sliding on ice the **Pull To Shoulder** behavior should probably be ignored for the present.

By adding these virtual sensors to abstract the sensor data to higher and higher levels, we have effectively created a sensor hierarchy that parallels the existing behavior hierarchy (Fig. 2). Not only has the sensor data been abstracted for high-level behaviors, it has also been abstracted for high-level virtual sensors. As a result, the high-level sensors benefit from the abstraction and can be conceptually simple.

Using priorities and virtual sensors, the system has four distinct stages (Fig. 3). First, the primitive sensors sense the environment and generate output. Second, virtual sensors process these outputs so they are usable by behaviors and then generate priorities which give overall indications of the importance of the current output. These sensor outputs and priorities are then used by the behavior hierarchy to produce control commands. Lastly, these control commands are sent to motor controllers or the corresponding team for execution. The effectiveness of this system is dependent on the process used to generate the priorities and their quality. The amount of effort is similar to that of the standard approach but the responsibility has been shifted from the behaviors to the virtual sensors. In doing so, we have reduced the composite behavior to its essence.

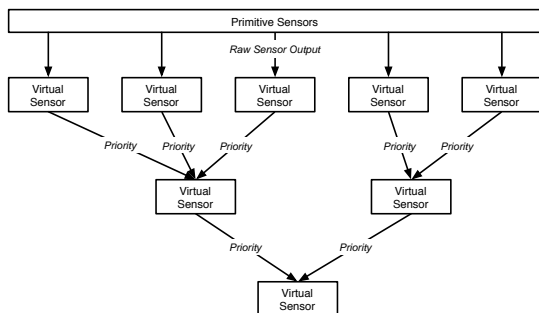


Fig. 2. A sample sensor hierarchy that could be used in conjunction with a behavior hierarchy. Virtual sensors produce a priority for a corresponding behavior that is then used by virtual sensors at a higher level to produce further abstracted priorities.

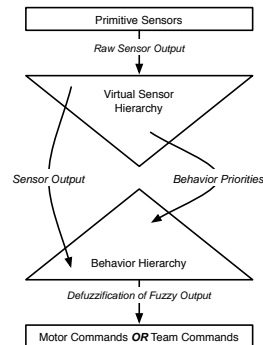


Fig. 3. The full system comprised of a sensor and behavior hierarchy. While a specific sequence is depicted, there is no requirement for the individual stages to be performed synchronously. In fact, each stage could be evaluated at different rates.

### B. Using Priorities at the Team Level

The priority abstraction also means that a composite behavior is no longer tied to executing in a specific environment. This is due to the fact that composite behaviors no longer rely on low-level information for their decision-making. For example, it no longer matters whether or not the obstacle to be avoided is a rock in the path of an individual agent or a dense forest in the path of a team of agents. Both are obstacles that need to be avoided and the response should be the same. What it means to avoid an obstacle is the same, regardless of the situation. The same behavior can now be used at different levels in the agent hierarchy without modification.

## III. IMPLEMENTATION

To evaluate the effectiveness of this sensor abstraction approach, we applied it to the problem of coordinating multi-team behaviors. While this will not explicitly test the scaling of the architecture to large numbers of behaviors, it does test the effects of the process of abstracting sensor information into priorities. This is fundamental to the scaling of the architecture since it is dependent on the quality and effectiveness of the abstraction of low-level information. Furthermore, the use of fewer behaviors means simpler agents and allows us to focus on the abstraction process.

### A. Flocking Behaviors

For this experiment, we chose to implement the Flocking behavior described by Reynolds [1]. Flocking is the combination of the three sub-behaviors Cohesion, Separation, and Alignment. Cohesion steers the agent towards the average position of all the agents in the neighborhood. Separation steers the agent away from other agents in the neighborhood while giving greater weight to closer agents. Alignment steers the agent such that it is heading in the same direction and at the same speed as nearby agents. Each of these behaviors are conceptually simple and simple to implement. Furthermore, they require different processing to be performed on the raw sensor data. Since Cohesion and Separation are constantly competing for control, careful coordination of these behaviors is required. As a consequence, the calculation of accurate priorities is imperative.

## B. Team Flocking

Once these behaviors were constructed, the sensor and behavior hierarchies were copied, without modification, to another level of the team hierarchy. This gave the team itself the ability to flock with other teams. In order for the individual agents to act using these team behaviors, another primitive behavior *Team Cmd.* was added to the hierarchy along with a new composite behavior *Team Flock* that coordinated this new behavior with the existing *Flocking* behavior.

## IV. DISCUSSION

Development of the composite behaviors was relatively simple due to the use of priorities. It was no longer necessary to fully analyze the current situation when determining the weighting of the *Cohesion* and *Separation* behaviors. All that was required was to compare their relative priorities. If both behaviors were at a high priority, preference was given to the *Separation* behavior in order to prevent collisions between agents.

The only difference between the behavior hierarchies at different team levels was the addition of the previously mentioned *Team Cmd.* behavior that integrates the recommendation provided by a higher level. As a result, the same behaviors and virtual sensors were used at each level of the team hierarchy. This reuse allowed improvements and bug fixes applied to a behavior to be immediately reflected at all levels without extra effort.

The abstraction of sensor output into priorities showed no evidence of negatively impacting the ability of the fuzzy behavior hierarchy to coordinate multiple behaviors. The same sort of behavior seen in other flocking systems was readily visible. Although the definitions for behaviors remain fairly constant between implementations, it is the method of coordination of the behaviors which varies. The same behavior was evident at all levels. For instance, *Cohesion* was easily visible at both the individual and team levels.

## V. CONCLUSIONS

The use of priorities allowed us to abstract the sensor output without losing the fidelity required to effectively coordinate behaviors. The resulting behavior hierarchy was successfully reused in multiple levels within the team hierarchy. This abstraction and reuse allowed us to more quickly and easily develop the system.

The integration of team commands into the behavior hierarchy is difficult, but it does not appear that there is anything fundamentally different between these commands and the results of other behaviors. They are just another instance of the behavior coordination problem. This is a situation where machine learning could be used to great effectiveness.

## VI. ACKNOWLEDGMENTS

We would like to thank all the members of the AI Research group at the University of Oklahoma for their contributions. This material is based upon work supported by the U. S. Army Research Laboratory and the U. S. Army Research Office under grant number DAAD19-03-1-0142.

## REFERENCES

- [1] C. W. Reynolds, "Flocks, herds, and schools: A distributed behavioral model," *Computer Graphics*, vol. 21, no. 4, pp. 25–34, July 1987.
- [2] G. W. Flake, *The Computational Beauty of Nature: Computer Explorations of Fractals, Chaos, Complex Systems, and Adaptation*. MIT Press, 1998.
- [3] O. Soysal and E. Sahin, "Probabilistic aggregation strategies in swarm robotic systems," in *Proc. of the IEEE Swarm Intelligence Symposium*, Pasadena, California, June 2005.
- [4] E. Bahceci and E. Sahin, "Evolving aggregation behaviors for swarm robotic systems: A systematic case study," in *Proc. of the IEEE Swarm Intelligence Symposium*, Pasadena, California, June 2005.
- [5] Y. Altshuler, A. Bruckstein, and I. A. Wagner, "Swarm robotics for a dynamic cleaning problem," in *Proc. of the IEEE Swarm Intelligence Symposium*, Pasadena, California, June 2005.
- [6] R. A. Brooks, "A robust layered control system for a mobile robot," *IEEE Journal of Robotics and Automation*, vol. 2, no. 1, pp. 14–23, March 1986. [Online]. Available: <http://jmvidal.cse.sc.edu/library/brooks86a.pdf>
- [7] P. Pirjanian, "Behavior coordination mechanisms – state-of-the-art," Institute for Robotics and Intelligent Systems, University of Southern California, Tech. Rep. IRIS-99-375, October 1999.
- [8] M. Nicolescu and M. J. Matarić, "A hierarchical architecture for behavior-based robots," in *International Joint Conference on Autonomous Agents and Multiagent Systems*, Bologna, Italy, July 2002.
- [9] A. Saffiotti, "Fuzzy logic in autonomous robotics: behavior coordination," in *Proc of the 6th IEEE Intl Conf on Fuzzy System*, Barcelona, Spain, 1997, pp. 573–578, online at <http://www.aass.oru.se/~asaffio/>.
- [10] A. Saffiotti and Z. Wasik, "Using hierarchical fuzzy behaviors in the robocup domain," in *Autonomous Robotic Systems*, D. M. C. Zhou and D. Ruan, Eds. Berlin, DE: Springer-Verlag, 2003, pp. 235–262.
- [11] E. Tunstel, "Fuzzy-behavior synthesis, coordination, and evolution in an adaptive behavior hierarchy," in *Fuzzy Logic Techniques for Autonomous Vehicle Navigation*, ser. Studies in Fuzziness and Soft Computing Series, A. Saffiotti and D. Driankov, Eds. Physica-Verlag, 2001, vol. 61, ch. 9.