Proceedings of the 5th WSEAS Int. Conf. on Artificial Intelligence, Knowledge Engineering and Data Bases, Madrid, Spain, February 15-17, 2006 (pp31-35)

A Petri Net Simulator for Self-organizing Systems

DIANXIANG XU[1], PRITI BORSE[1], KARL ALTENBURG[2], KENDALL NYGARD[1]

[1]Department of Computer Science
[2]Department of Accounting and Information Systems
North Dakota State University
Fargo, ND 58105
United States of America

Abstract: - Self-organizing systems have the general property that the primitive elements that comprise the system interact in order to achieve emergent global properties. We describe an extended Petri net simulation architecture for such systems that is based on interacting agents that carry out autonomous actions, yet collaborate with each other in structured ways. There is no central controlling entity. The approach was motivated by an application to mission planning for unmanned air vehicles (UAVs), but is applicable to more general agent-oriented systems.

Key-Words: - Agent, Petri nets, Simulation, UAV, Self-organizing, swarm

1 Introduction

Considerable research has been directed at the implementation of self-organizing systems [2,10]. We consider systems that have constituent units that respond to stimuli from each other and their environment over time to induce a unified functionality. These properties are not directly imposed by one or even a few controlling elements, but rather are epiphenomena that emerge naturally through their interactions. In the worlds of insects, birds, mammals, and other animals, emergent properties are commonly observed in the form of flocks, swarms, herds, schools, and packs, all of which are considered to be organized by low-level actions and behaviors. We report on a architecture designed to simulate systems comprised of heterogeneous individuals with multiple sensors that detect signals which trigger reactive actions that produce structured activity for the group as a whole.

The work is primarily motivated by military projects that have the need for rapid prototyping of simulations of systems of cooperating Unmanned Air Vehicles (UAVs) [1,4,8,11]. This requires that the simulator support higher-level constructs that can be easily assembled to specify a newly devised mission scenario. A related motivation is connected with the needs of NASA projects involving space missions that would use small swarming space vehicles with solar sails to catalog the asteroid belt [7].

Validation of self-organizing systems is notoriously difficult, which motivates the development of a simulator design aimed at realizing the benefits of a software engineering formal method. Benefits include the ability to keep track of individual behaviors, track mission goals, prove the correctness of system properties, and ensure that unforeseen outcomes will not happen, and check for errors. We demonstrate that a suitably extended Petri net based architecture can provide the desired properties.

The architecture that we present is based upon Petri nets. Using high-level Colored Petri nets, we are able to formulate the low-levels controls associated with sense/act cycles at the individual vehicle level. To realize the effects of UAV behaviors on the simulation environment, we also extend the basic Petri net operations by associating computational procedures with transitions.

2 Motivation

There are two high-level needs that motivate the design of the simulator. First, it is desirable for the design to be flexible enough to conveniently support a variety of UAV missions in which the cooperative control of the vehicles is accomplished through a local sensing/reaction paradigm. Second, the approach should support the advantages of a formal model.

2.1 A Flexible Simulator for UAV Missions

Unmanned Aerial Vehicles (UAVs) have become very important systems for modern warfare. Modern UAVs potentially provide a cost effective means of conducting a variety of military missions, including basic needs such as reconnaissance and suppression of enemy air defenses. Several simulators have been developed to evaluate the performance of conducting UAV missions with a reactive, behavior-based approach to individual and collective intelligence [1,4,7,10]. Central to these simulations is the design and implementation of the control structure for the UAVs. It is highly desirable for the simulation system to easily support a wide variety of simulation scenarios of the emergent intelligence type. Example scenarios that have been evaluated thus far include the following: i) sweep searching of an area of interest by multiple UAVs for reconnaissance purposes, ii) persistent patrolling by UAVs of a high-value asset to help protect it from an enemy strike, iii) forward air controller searching of an area with a vehicle available for striking as needed, and iv) sweep searching of a geographical area with striking actions decided by an explicit decision support model such as a partially observable Markovian decision process (POMDP). These scenarios, while exhibiting important differences, do share a significant number of primitive task, such as following waypoints, entering into orbits around objects of interest, avoiding collision, and maintaining offset distances from neighbors. The need to provide the flexibility to accommodate scenario differences while still providing infrastructure that makes the simulator easy to use is a basic design challenge. The approach using extended Petri nets allows generic types of tasks to be developed and implemented, essentially forming a library of low-level primitives that can be used as components for assembling a specific mission. This maximizes reuse, and allows for rapid prototyping of different UAV simulations, which is very cost-effective.

2.2 Advantages of a Formal Method

There are several motivations for developing the simulation with a formal approach. A major issue is that by their nature, the emergent properties of self-organizing systems may not be known or understood and therefore difficult to test. The highly distributed nature and large number of interacting agents in self-organizing systems present other testing complications. Finally, testing is significantly complicated if the agents are imbued with machine learning features that modify their capabilities over time.

Petri nets are an established formalism for modeling concurrency, synchronization, and non-determinism in distributed systems [5]. They provide a mathematical tool for reasoning about system behaviors, as well as a graphical notation for visualization of system modeling and analysis. From the outset Petri nets were designed to support testing for functional correctness. A Petri net specification can readily be tested for deadlock, reachability of states, persistence, liveness and boundedness. In the context of simulation of self-organizing UAV systems, these formal capabilities of the Petri net representation support the following specific kinds of functionalities for our application domain:

- Understanding and predicting emergent behaviors
- Tracking of specific sequences of individual UAV actions
- Tracking mission goals (e.g., destroying targets, populating battlefield intelligence maps)
- Proving correctness of system properties
- checking for concurrent system execution errors, such as race conditions

2 The Simulation Architecture

A Petri net-based architecture addresses the need for generic modeling of UAV controls for task executions that follow the reactive, behavior-based approach to intelligence. Following Petri net conventions, *places* are nodes that hold tokens that represent system state. *Arcs* connect places to *transitions* which have both input and output places. When a transition *fires*, tokens are moved from all of its input places to all of its output places. Arcs can have a *guard*, which is an attached Boolean expression which allows a token to travel only when the guard evaluates to true.

We utilize the RENEW [6] software as the Petri net tool for implementing the UAV simulator. The key feature of RENEW that motivates this choice is seamless integration of high-level Petri nets with the object-oriented programming language Java. In RENEW, an arc label (a tuple of variables and expressions), is written in a general inscription language. A transition can fire only if there is a substitution of variables in the labels of all of its input arcs that evaluates their guard condition to true. Firing

a transition removes the tokens that are unified with the arc labels and adds new tokens to the output places of the transition. The new tokens are constructed in terms of the arc labels and the variable substitution. Transition firing also executes Java code associated with the transition. RENEW also supports colored tokens with a data type that can be tested and manipulating.

Figure 1 illustrates a low level task net for synchronously controlling speed adjustments of multiple UAVs. Associated with place p_1 are distinguished colored tokens uav1, uav2,… representing individual uavs whose speed is to be adjusted from a current value to a value that exceeds the value of variable max in increments of y units over a simulation time step of x units. There is a need to enforce round robin synchronization of the speed adjustment steps, to ensure that no uav is allowed to repeatedly adjust its own speed to the max value before the others get a chance to adjust theirs. The arc that goes from p_1 to t_3 is an inhibitor that ensures that all of the uav tokens in p_1 will fire one by one in a given time step.

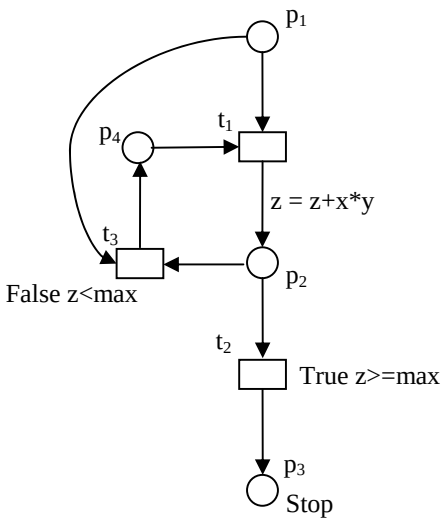

Figure 1. Basic task net for UAV speed control

An important feature of the simulator is that the visual animation for a scenario is directly controlled by the Petri net. This is possible through the association of Java code segments with transitions in the Petri nets. Figure 2 illustrates a more complex task: controlling orbits around a center point at a specified radius in preparation for aligning UAV for a subsequent searching sweep.

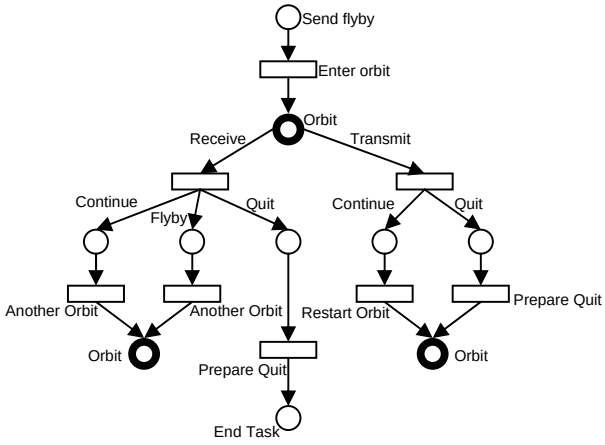

Figure 2. Task net for Orbiting

This task net applies at the point where a set of UAVs are transitioning from maintaining alignment for searching purposes to configuring turns and loiter orbits at the end of the search area in preparation for another search sweep. Tracing from the top of the figure, the first UAV in the group issues a flyby signal to alert the follower UAV that the current orbiting position is occupied, then begins a loiter orbit. Follower UAVs assume their orbits in turn; until the last one senses that it has no follower and sends a signal that the formation should commence the next sweep. After completing their orbit in process, each UAV begins sweeping. All of the behaviors are entirely local sensing and reacting pairs. By design the model is entirely indifferent to the numbers of UAVs in the mission, and requires no global communication. Each UAV in the mission is directly controlled by its set of inherent Petri task nets. Task nets have been developed for several types of common mission tasks, including basic waypoint following, tracking a target, releasing a chosen weapon, serving as a communication relay, and maintaining a patrol posture around an asset. The task nets can also incorporate intelligent decision logic, such as choosing between searching and striking, and soliciting information from neighboring UAVs before choosing a movement action.

4 Benefits of a Formal Method

The Petri net approach provides a mathematically rigorous model for complex self-organizing systems. The immediate benefits of the approach concern the ability to explore the state space of the model in detail through reachability analysis. Through reachability graphs, it is possible to track individual behaviors in detail for any given scenario. Natural extensions provide the ability to track larger goals that are specified at levels above the task nets themselves. These capabilities provide rigorous verification of the models in a way that would be impossible with empirical testing. Reachability analysis is often characterized as having the disadvantage of being susceptible to explosive growth of the state space. We deliberately configure our scenarios so that the missions they model are completed in a bounded time frame, thus limiting the state space.

Another benefit realized from the approach is that the simulation designs have a precision and discipline that is typically absent from more ad hoc approach. The graphical nature of the nets makes a scenario easy to build as well as supporting the associated precision and discipline. Finally, the Petri net approach provides built-in mechanisms for checking for certain types of errors that can easily occur in concurrent systems, such as race conditions.

3 Conclusions

A multi-agent simulation architecture for self-organizing systems was developed using high level Petri nets. The application is motivated by systems of cooperative unmanned air vehicles, but readily applies to other systems as well. The low-level task nets that are developed facilitate rapid prototyping of alternative simulations. The framework provides a rigid formalism of visualization for designing and tuning control structures of UAV simulations.

The formalism provides several significant advantages. Model verification and testing is possible with considerable rigor. Details of agent behaviors and goals reached are available through reachability analysis. Debugging of simulated scenarios is facilitated through the use of tools that can detect deadlock, liveness, and undesired states.

References:

[1] K. Altenburg, J. Schlecht, and K. Nygard. An Agent-based Simulation for Modeling Intelligent Munitions. *In Proc. of the Second WSEAS Int. Conf. on Simulation, Modeling and Optimization*, Skiathos, Greece, 2002.

[2] E. Bonabeau, M. Dorigo, and G. Theraulaz, *Swarm Intelligence: From Natural to Artificial Systems.* Santa Fe Institute Studies in the Sciences of Complexity, New York: Oxford University Press, 1999.

[3] R.A. Brooks. A Layered Control System for a Mobile Robot. *IEEE Journal of Robotics and Automation*. 2(1): 14-23, 1986.

[4] N. Huff, A. Kamel, and K. Nygard, An Agent Based Framework for Modeling UAVs. In Proc. of the 16th International Conference on Computer Applications in Industry and Engineering (CAINE03), 2003.

[5] K. Jensen. *Coloured Petri Nets: Basic Concepts, Analysis Methods and Practical Use*. Springer-Verlag, 1992.

[6] O. Kummer, F. Wienberg, and M. Duvigneau. *Renew - User Guide (Release 1.6)*. University of Hamburg, September 2002. Available at http://www.informatik.uni-hamburg.de/TGI/renew/

[7] Lua, Chin A., Altenburg, Karl, and Nygard, Kendall E., ANTS with Firefly Communication, in Proceedings of the 2005 International Conference on Artificial Intelligence (ICAI), Las Vegas, 2005

[8] C. Lua, K. Altenburg, and K. Nygard. Synchronized Multi-Point Attack by Autonomous Reactive Vehicles with Simple Local Communication. *In Proc. of the IEEE Swarm Intelligence Symposium*, Indianapolis, 2003.

[9] T. Murata. Petri Nets: Properties, Analysis and Applications. *Proc. of the IEEE*, vol.77, no.4, pp. 541-580, April 1989.

[10] G. Di Marzo Serugendo, A. Karageorgos, O. F. Rana, and F. Zambonelli, eds., *Engineering Self-Organising Systems, Nature-Inspired Approaches to Software Engineering*, vol. 2977 of Lecture Notes in Computer Science, Springer, 2004.

[11] J. Schlecht, Joseph, K. Altenburg, B. Ahmed, and K. Nygard. Decentralized Search by Unmanned Air Vehicles using Local

Communication. *In Proc. of the International Conference on Artificial Intelligence,* Volume II, Las Vegas, 757-762, 2003.