Verifying Large Numbers of Cooperating Adaptive Agents

Christopher Rouff
SAIC
703-676-6184
rouffc@saic.com

Mike Hinchey
NASA GSFC
Code 581
michael.g.hinchey@nasa.gov

Walt Truszkowski, James Rash
NASA GSFC
Code 588
walter.f.truszkowski@nasa.gov
james.l.rash@nasa.gov

Abstract

NASA is proposing to use large numbers of cooperating spacecraft in future exploration missions. These swarms will exhibit complex behaviors and interactions that can result in unintended emergent properties. Verifying the proper behavior of these new types of missions will be critical to their success. This paper gives the results of research into the use of formal methods techniques for verification of a large number of cooperating agents. Multiple formal methods were evaluated to determine their effectiveness in assuring the behavior of swarms of spacecraft. The NASA ANTS mission was used as an example of swarm intelligence against which to apply the formal methods. This paper discusses the evaluation of these formal methods, provides a partial specification of ANTS using four selected methods, and an outline of an integrated formal method for verifying future NASA swarm missions.

Key Words: Verification, formal methods, swarms, autonomy, emergent behavior, spacecraft.

1. Introduction

Intelligent swarms of autonomous spacecraft are being proposed for missions that have complex behaviors and interactions. Bonabeau et al. [1] who studied self-organization in social insects state that "complex collective behaviors may emerge from interactions among individuals that exhibit simple behaviors" and describe emergent behavior as "a set of dynamical mechanisms whereby structures appear at the global level of a system from interactions among its lower-level components." The Autonomous Nano Technology Swarm (ANTS) mission is an example of one of the types of swarm missions NASA is considering. Since ANTS and other similar missions will consist of autonomous spacecraft that may be out of contact with the Earth for extended periods of time, and have low communications bandwidths due to weight constraints, it will be difficult for ground controllers on Earth to observe improper behavior and correct any errors after launch.

Swarm technologies, whereby federated systems of spacecraft or rovers (of varying degrees of collective intelligence) mimic the societal behaviors of swarms, colonies, or flocks in nature (such as of bees, ants, or geese) appear to offer great potential, and are becoming a major focus for future NASA missions. These types of missions provide greater flexibility and the opportunity to gather more science data than traditional single vehicle missions [4]. The emergent and autonomic properties of these missions make them powerful, but at the same time more difficult to design and verify. These missions are also more complex than previous types of missions, and neither NASA nor anyone else has much experience in developing, verifying, and validating them.

Swarm-based systems will naturally bear all the hallmarks of a complex system – perhaps millions of lines of code, complex hardware-software interactions, real-time behavior, continual updates from environmental sensors, and a domain that is not fully understood. More importantly, such a system can never be exhaustively tested. With the large number of parallel and distributed swarm members, the state space is extremely large and it is impossible to test every path through the state space.

One of the highest possible levels of assurance comes from formal methods [5]. Once written, a formal specification can be used to prove properties of a system (e.g., the underlying system will go from one state to another, or not into a specific state) and check for particular types of errors (e.g., race or livelock conditions). A formal specification can also be used as input to a model checker for further validation.

The authors have investigated a collection of formal methods techniques for verification and validation of spacecraft using swarm technology. This paper discusses the evaluation of these formal methods and the partial specifications of the ANTS mission that were undertaken using four selected methods. We then provide an evaluation of the methods and outline a possible new integrated formal method that could be used to verify future NASA swarm missions.

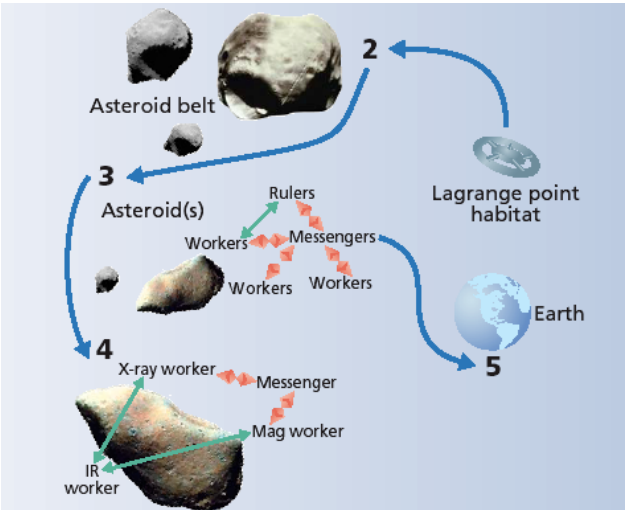

Figure 1: ANTS Mission Concept.

2. ANTS Mission Overview

The Autonomous Nano-Technology Swarm (ANTS) mission [3] will comprise approximately 1000 autonomous pico-class (approximately 1kg) spacecraft that will search the asteroid belt for asteroids that have specific characteristics (Figure 1). Under present thinking, the swarm comprises three distinct classes: workers, leaders, and messengers. Workers will carry high-end miniature instruments. Leaders will be goal-oriented and will direct the workers. Messengers will route communications between leaders, workers, and Earth. To examine an asteroid, the spacecraft will have to cooperate with each other, since they each only have a single instrument on board. Exploration activities of the swarm will involve a hierarchical social behavior where some spacecraft will have the role of directing others. Sub-swarms will act as teams that explore a particular asteroid based on the asteroid's properties and share resources (instruments) between them.

This mission will involve high levels of autonomy as well as autonomic properties. Artificial intelligence technologies such as genetic algorithms, neural nets, fuzzy logic and on-board planners are being investigated to assist the mission in maintaining high levels of autonomy. Crucial to the mission will be the ability to modify its operations autonomously to reflect the changing nature of the mission, and the restrictions of distance and low bandwidth communications back to Earth.

3. Approaches and Assurance

As mission software becomes increasingly complex, testing and error discovery become more difficult. This is especially true of highly parallel processes and distributed computing, such as swarms and distributed systems with autonomic properties. Race conditions in these systems can rarely be found by inputting sample data and checking whether the results are correct. These types of errors are time-based and only occur when processes send or receive data at particular times, or in a particular sequence, or after learning occurs. To find these errors, the software processes involved have to be executed in all possible combinations of states (state space) that the processes could collectively be in. Because the state space is exponential to the number of states, exhaustive testing can become impossible with a relatively small number of processes. Traditionally, to get around the state explosion problem, testers have artificially reduced the number of states of the system and approximated the underlying software using models.

Formal methods are proven approaches for assuring the correct operation of complex interacting systems [5, 8, 9]. They are particularly useful for specifying complex parallel and distributed systems where more than one person was involved in the development. Once written, a formal specification can be used to prove properties of a system correct, check for particular types of errors (e.g., race conditions), and can be used as input to a model checker. Verifying emergent behavior, however, is one area that most formal methods have not addressed to required levels of satisfaction.

We have surveyed formal methods techniques to determine whether there exist formal methods that would be suitable for verifying swarm-based systems and their emergent behavior. A number of formal methods support the specification of either concurrency or algorithms [10]. Though there are a small number of formal methods that have been used to specify swarm-based systems, only two formal approaches have been used to analyze the emergent behavior of swarms, according to our investigations. Weighted Synchronous Calculus of Communicating Systems (WSCCS), a process algebra, was used by Tofts to model social insects [13], and to analyze the non-linear aspects of social insects [12]. X-Machines have been used to model cell biology [7], and modifications could increase the potential for specifying swarms. Simulation approaches are being investigated to determine emergent behavior. These approaches do not predict emergent behavior from the model, but rather model the emergent behavior after the fact.

4. Specifications and Evaluation

In the initial evaluation of specification techniques for swarm-based systems [11], specifications of the NASA ANTS mission employed Communicating Sequential Processes (CSP) [6], WSCCS, Unity Logic [2] and X-Machines. Partial specifications of ANTS using the four methods are presented here, along with an evaluation of

these methods and their potential for analyzing emergent behavior. In each case, only a sufficient amount of the ANTS mission to gather enough information to evaluate the method for specifying swarm-based systems was specified.

4.1. CSP

The combination of the goals of the individual ANTS spacecraft should equal the goals of the mission. The following is the top-level CSP specification of the ANTS mission.

Table 1: Leader States and Actions

State	Action	f	p
	Identity		
Commun-icating	SendMessageWorker	50	2
	SendMessageLeader	50	2
	SendMessageError	1	1
	ReceiveMessageWorker	50	2
	ReceiveMessageLeader	50	2
	ReceiveMessageError	1	1
Reasoning	ReasoningDeliberatve	50	2
	ReasoningReactive	50	2
Processing	ProcessingSortingandStorage	17	2
	ProcessingGeneration	17	2
	ProcessingPrediction	17	2
	ProcessingDiagnosis	16	2
	ProcessingRecovery	16	2
	ProcessingRemediation	17	2

$$ANTS_{goals} = Leader_{i,i_goals} \parallel Messenger_{j,m_goals} \parallel Worker_{k,w_goals} \bullet 1 \le i \le m, 1 \le j \le n, 1 \le k \le p$$

where m is the number of leader spacecraft, n the number of messenger spacecraft, and p the number of worker spacecraft. The ANTS mission is initialized, with a set of goals given to it by the principal investigator. Some of these goals are given to the leader (some of the goals may not be given to the leader if, for example, they are ground based or not applicable to the leader). The leader spacecraft specification consists of two processes:

$$Leader_i = LEADER_COM_{i,\{\}} \parallel LEADER_INTELLIGENCE_{i,goals,model}$$

The communication process, *LEADER_COM*, specifies the behavior of the spacecraft as it relates to communicating with the other spacecraft and Earth, and specifies a protocol between the spacecraft. The second process, *LEADER_INTELLIGENCE*, is the specification of the intelligence of the leader. This is where the deliberative and reactive parts of the intelligence are implemented, including leader goal maintenance. In addition to the goals, this process maintains the spacecraft and environment models and specifies how they are modified during operations. The following is a partial specification of LEADER_COM:

$$\begin{aligned} LEADER_COM_{i,conv} &= leader.in?msg \rightarrow \\ &case\ LEADER_MESSAGE_{i,conv,msg}\ if \\ &\quad sender(msg) = LEADER \\ &MESSENGER_MESSAGE_{i,conv,msg}\ if \\ &\quad sender(msg) = MESSENGER \\ &WORDKER_MESSAGE_{i,conv,msg}\ if \\ &\quad sender(msg) = WORKER \\ &EARTH_MESSAGE_{i,conv,msg}\ if \\ &\quad send(msg) = EARTH \\ &ERROR_MESSAGE_{i,conv,msg}\ otherwise \end{aligned}$$

4.2. WSCCS

To model the ANTS Leader spacecraft, WSCCS (Weighted Synchronous Calculus of Communicating Systems), a process algebra, takes into account

- The possible states (agents) of the Leader,
- Actions each agent-state may perform that would qualify them to be in those states, and
- The relative frequency and priority of each action.

Agent states and a view of priority (p) and frequency (f) of the actions of the Leader are given in Table 1. The states of the Leader can now be defined by definition statements such as the following.

$$\begin{aligned} &Communicating \equiv \\ &50\omega^2 : ReasoningDeliberative.Reasoning + \\ &50\omega^2 : ReasoningReactive.Reasoning + \\ &17\omega^2 : ProcesingSortingAndStorage.Processing + \\ &17\omega^2 : ProcessingGeneration.Processing + \\ &17\omega^2 : ProcessingPrediction.Processing + \\ &16\grave{u}^2 : ProcessingDiagnosis.Processing + \\ &16\omega^2 : ProcessingRecovery.Processing + \\ &17\omega^2 : ProcessingRemediation.Processing \end{aligned}$$

This statement provides that a Leader, when in a Communicating state, has the option (is allowed) to perform any action from the set given by:

$$\left\{\begin{array}{l} ReasoningDeliverative, ReasoningReactive, \\ ProcessingSortingAndStorage, ProcessingGeneration, \\ ProcessingPrediction, ProcessingDiagnosis, \\ ProcessingRecovery, ProcessingRemediation \end{array}\right\}$$

and that the Communicating Leader will perform *ReasoningDeliberatve* with a probability of 25% and will give that action the same priority as the others. The second term in the statements specifies that the Communicating Leader will perform *ReasoningReactive* with the same 25% probability and priority of 2. The symbol + in this notation denotes a choice between the allowed actions, and the choice will be made based on the frequencies and priorities of each allowable action.

The single Leader by itself shows the following example emergent behavior. The Communicating Leader will choose to transition to a Processing state with a probability of 50% by choosing to process by one of the six available processing types. It will choose from the six types with equal probability.

To study the emergent behavior of a swarm of Leaders, we begin by considering a swarm of only two Leader spacecraft, L1 and L2. Both leaders tick forward by performing one action per time step. Thus the two Leaders perform a composition of two actions, denoted $m1\omega^{k1} * m2\omega^{k2}$, on each time step. When this happens, the pair of leaders behave according to the rules for composition:

$$n\omega^{k+1} * m\omega^{k} = (nm)\omega^{k+(k+1)} = m\omega^{k} * n\omega^{k+1}$$

$$n\omega^{k} * m\omega^{k} = (nm)\omega^{k+k} = m\omega^{k} * n\omega^{k}$$

This gives the Leader pair their own set of relative frequencies and priorities. Since there are two Leaders and each has three states and 14 possible actions, the pair of leaders has 9 possible state pairs and 196 possible action compositions. The two-Leader swarm will have a much higher probability of having both leaders communicating or reasoning, rather than processing. Processing will be performed by the swarm, but with much less frequency than communicating or reasoning. These features can be extrapolated to a swarm of *n* leaders as follows.

Given a swarm of *n* Leader Spacecraft, the *n*-leader swarm will tick forward in time by performing simultaneous actions – one action per leader per time step. Thus the *n*-leader swarm will perform (on each time step) a composition of *n* actions, denoted with weight $m_1\omega^{k_1} * m_2\omega^{k_2} * \ldots * m_n\omega^{k_n}$. When this happens, the *n*-leader swarm must still behave according to the rules for composition seen before.

This gives the *n*-leader swarm its own set of relative frequencies and priorities. Since there are *n* Leaders and each has three states and 14 possible actions, the swarm of *n* leaders has 3^n possible state sets and 14^n possible action compositions. There are only two possible priority values and four possible relative frequency values available. Thus we can narrow down that each priority k_i must be either 1 or 2 and each relative frequency m_i must be either 1 (if the priority is 1) or one of 16, 17 or 50 (if the priority is 2). Thus the remaining options for leaders in the swarm will include communicating, reasoning, and processing (either by prediction or recovery, or otherwise). Let N_{comm} be the number of leaders in the swarm who choose to communicate (not in error) on a given time step. Let $\boldsymbol{N_{reason}}$ be the number of leaders in the swarm who choose to reason on that time step. Let $\boldsymbol{N_{process16}}$ be the number of leaders in the swarm who choose to process (by prediction or recovery) on that time step. Lastly, let $\boldsymbol{N_{process17}}$ be the number of leaders in the swarm who choose to process (by other means) on that time step. Then, each action by each leader will have priority 2 and relative frequency 16, 17 or 50. Thus, the composition of their actions will have weight:

$$m_1\omega^{k_1} * m_2\omega^{k_2} * \ldots * m_n\omega^{k_n} =$$
$$\left(50^{N_{comm}+N_{reason}}\right)\left(16^{N_{process16}}\right)\left(17^{N_{process17}}\right)\omega^{2n}$$

From this weighting, we can see that drastically higher frequencies exist when larger numbers of the leaders in the swarm choose to communicate or to engage in reasoning. Much lower frequencies exist when larger numbers of leaders choose to undertake processing. Thus the swarm will be communicating and reasoning much

Table 2. Leader States and Transitions

Q	Φ	$Q' = F(Q,\Phi)$
Start	SendMessage	Commun.
	ReceiveMessage	Commun.
	Reason	Reasoning
	Process	Processing
Commun.	SendMessage	Commun.
	ReceiveMessage	Commun.
	Reason	Reasoning
	Process	Processing
Reasoning	SendMessage	Commun.
	ReceiveMessage	Commun.
	Reason	Reasoning
	Process	Processing
Processing	SendMessage	Commun.
	ReceiveMessage	Commun.
	Reason	Reasoning
	Process	Processing

more often than processing, although processing will take place.

4.3. Unity Logic

To model the ANTS Leader spacecraft with Unity Logic, we consider states of the Leader. In Unity Logic, we will consider the states of the Leader, and the actions taken to cause the Leader to be in those states, but the notation will appear much closer to classical logic. Predicates will be defined to represent the actions that would put the Leader into its various states. Those predicates then become statements which, if true, would mean that the Leader had performed an action that put itself into the corresponding state. The Leader program would then be specified using ***assertions*** such as the following for Communication:

[Communicating]ReasoningDeliberatve(Leader)[Reasoning]
[Communicating]ProcessingGeneration(Leader)[Processing]

4.4. X-Machines

To model the ANTS Leader spacecraft as an X-Machine we must view the Leader as a tuple:

$$L = \{Input, Memory, Output, Q, \Phi, F, start, m_0\}$$

where the components of the tuple are defined as

$$Input = \begin{Bmatrix} worker, messenger, leader, error, \\ Deliberative, Reactive, \\ SortAndStore, \\ Generate, Predict, Diagnose, \\ Recover, Remediate \end{Bmatrix}$$

$Memory$ is written as a tuple $m = (Goals, Model)$ where $Goals$ describes the goals of the mission and $Model$ describes the model of the universe maintained by the Leader. The initial memory is denoted by $(Goals_0, Model_0)$. When the goals and/or model change, the new tuple is denoted as $m' = (Goals', Model')$.

$$Output = \begin{Bmatrix} SentMessageWorker, \\ SentMessageMessenger, SentMessageLeader, \\ SentMessageError, ReceivedMessageWorker, \\ ReceivedMessageMessenger, \\ ReceivedMessageLeader, \\ ReceivedMessageError, \\ ReasonedDelibartively, ReasonedReactively, \\ ProcessedSortingAndStoring, \\ ProcessedGeneration, ProcessedPrediction, \\ ProcessedDiagnosis, ProcessedRecovery, \\ ProcessedRemediation \end{Bmatrix}$$

$Q = \begin{Bmatrix} Start, Communicating, \\ Reasoning, Processing \end{Bmatrix}$ is a set of states.

$\Phi = \begin{Bmatrix} SendMessage, ReceiveMessage, \\ Reason, Process \end{Bmatrix}$ is a set of (partial) transition functions where each transition function maps $Memory \times Input \rightarrow Output \times Memory$ as in the following:

$$\Phi(m, Worker) = (m', SentMessageWorker)$$
$$\Phi(m, Generate) = (m', ProcessedGeneration)$$

Then $F : Q \times \Phi \rightarrow Q$ is defined according to definitions such as in Table 2.

5. Evaluation of Methods

CSP is a process algebra and is very good at specifying the process protocols between and within the spacecraft, and at analyzing the result for race conditions. Being able to evaluate a system for race conditions is particularly useful in swarm-based systems which are highly parallel. From a CSP specification, reasoning about the specification can be undertaken to determine race conditions. The specification may also be converted into a model checking language and analyzed with a model checker.

WSCCS provides a process algebra that takes into account the priorities and probabilities of actions performed by the leader and other ANTS spacecraft. It further provides a syntax and large set of rules for predicting and specifying the choices and behaviors of the Leader, as well as a congruence and syntax for determining if two automata are equivalent. All of this in hand, WSCCS can be used to specify the ANTS spacecraft and to reason about, and even predict, the behavior of one or more spacecraft. This robustness affords WSCCS the greatest potential for specifying emergent behavior in the ANTS swarm. What it lacks towards that end is an ability to track the goals and model of the ANTS mission in memory. This may be achieved by blending the WSCCS methods with the memory aspects of X-Machines.

Unity Logic provides a logical syntax equivalent to simple Propositional Logic for reasoning about these predicates and the states they imply, as well as for defining specific mathematical, statistical, and other simple calculations to be performed. However, it does not appear to be rich enough to allow ease of specification and validation of more abstract concepts such as mission goals. This same simplicity, however, may make it a good tool for specifying and validating the actual Reasoning programming (as opposed to Reasoning process) portion of the ANTS Leader spacecraft, when the

need arises. In short, specifying emergent behavior in the ANTS swarm will not be accomplished well using Unity Logic.

X-Machines provide a highly executable environment for specifying the ANTS spacecraft. It allows for a memory to be kept and it allows for transitions between states to be seen as functions involving inputs and outputs. This allows us to track the actions of the ANTS spacecraft as well as write to memory any aspect of the goals and model. This ability makes X-Machines highly effective for tracking and effecting changes in the goals and model. However, X-Machines does not provide any robust means for reasoning about or predicting behaviors of one or more spacecraft, beyond standard propositional logic. This will make specifying emergent behavior difficult.

The following table summarizes these properties:

Properties of Current Formal Methods	
Method	**Useful Properties and Difficulties**
CSP	♦ Ability to model check ♦ Case-based reasoning approach ♦ Not able to specify algorithms and data manipulation
WSCCS	♦ Actions are given priorities and frequencies ♦ Defined algebra for extrapolation of how the agent will choose from various actions ♦ Probability used with action frequencies for predicting emergent behavior ♦ Allows for only a single state per space-craft (the craft may be in several concurrent states based on 2 or more sets of states) ♦ Actions with lower priorities will not actually occur ♦ There are no effective tools to aid calculation and interpretation of the emergent behavior of more than 2 agents ♦ No visualization capabilities exist to aid in the study of the emergent behavior
X-Machines	♦ Ability to store Goals and Model in memory (to maintain and update the goals of the mission and the model of the universe with each action taken) ♦ Uses a combination of current goals, model and current state to trigger an appropriate transition (this makes it adaptive to the current situation) ♦ Transition functions are very programmable ♦ Concepts of Input and Output can be used for verification and storage of the results of agent actions or processes ♦ Has few predictive qualities for emergent behavior of multiple agents.
Unity Logic	♦ Actions are seen as predicates (this allows for a more logic-based structure that can be easily programmed and allows the agent to be self-aware and track its own actions) ♦ Proof of correctness ♦ Has no sense of how or why an agent would choose to perform a given action and thus no ability to predict emergent behavior ♦ Needs a predictive quality for the agent's actions over time

6. An Integrated Swarm Formal Method

An effective formal method for use in the ANTS mission must be able to predict the emergent behavior of 1000 agents operating as a swarm, as well as the behavior of the individual agents. Crucial to the mission will be autonomic properties and the ability to modify operations autonomously to reflect the changing nature of the mission. For this, the formal specification will need to be able to track the goals of the mission as they change and to modify the model of the universe as new data comes in. The formal specification will also need to allow for specification of the decision-making process to aid in the decision of which instruments will be needed, at what location, with what goals, etc.

Once written, the formal specification must be able to be used to prove properties of the system, check for particular types of errors, as well as be used as input to a model checker. The formal method must also be able to track the models of the leaders and it must allow for decisions to be made as to when the data collected have met the goals.

To accomplish all of this, integrating the above methods seems to be the best approach for verifying cooperating swarm-based systems. Integrating the memory and transition function aspects of X-Machines with the priority and probability aspects of WSCCS and other methods may produce a specification method that will allow all the necessary aspects for specifying emergent behavior in the ANTS mission and other swarm-based systems.

The merging of these formal methods is currently being performed. The approach being taken is to use a

conserving integration [12] of the methods. In this type of formal methods integration, the base formalisms of the methods are maintained and relationships between the formalisms are developed to reflect the new formal method. This approach will preserve the strength of the underlying methods and allow a seamless specification of the ANTS mission, and the development of support tools using existing semantics of the methods.

7. Conclusion

Future NASA missions will increasingly use cooperative intelligent swarm-based systems to conduct new science and perform unmanned exploration. These new missions will be highly autonomous and will be out of touch with NASA ground stations for extended periods of time. In addition, due to the nature of swarm-based systems, they may be designed with, or unintentionally exhibit, emergent behaviors. These missions, consequently, will pose an even greater challenge for verification than has been the case in past missions.

To verify NASA swarm-based (autonomous) missions, an effective formal method must be able to predict the emergent behavior of 1000 agents functioning as a swarm, as well as the behavior of individual agents. Crucial to the mission will be the ability to modify operations autonomously to reflect the changing goals and data gathering strategies of the mission.

We have identified several important attributes needed in a formal approach for verifying swarm-based systems and potential formal methods that support these attributes. From this potential list we have undertaken sample formal specifications of part of the NASA ANTS mission using four of these methods and have determined that they each have appropriate attributes, but no single method alone is sufficient. We are currently integrating these methods to develop a new formal method for swarm-based systems and will test this new formal method through a formal specification of the NASA ANTS mission.

Acknowledgements

This work was supported by the NASA Office of Safety and Mission Assurance (OSMA) Software Assurance Research Program (SARP) and managed by the NASA Independent Verification and Validation (IV&V) Facility.

References

[1] Bonabeau, E., G. Theraulaz, et al. Self-organization in Social Insects, *Trends in Ecology and Evolution*, 1997, vol. 12, pp. 188-193.

[2] Chandy, K. M. and Misra, J. *Parallel Program Design: A Foundation*. Addison-Wesley. 1988.

[3] Curtis, S. A., J. Mica, J. Nuth, G. Marr, M. Rilee, and M. Bhat. ANTS (Autonomous Nano-Technology Swarm): An Artificial Intelligence Approach to Asteroid Belt Resource Exploration. *Proc. International Astronautical Federation, 51st Congress*, Oct. 2000.

[4] Clark, P. E., Curtis, S. A. and Rilee, M. L. ANTS: Applying a New Paradigm to Lunar and Planetary Exploration. *Proc. Solar System Remote Sensing Symposium*, Pittsburg, 2002.

[5] Hinchey, M. and Bowen, J. *Industrial-Strength Formal Methods in Practice*. Springer. 1999.

[6] Hoare, C.A.R. Communicating Sequential Processes. Communications of the ACM, 21(8):666-677, Aug., 1978.

[7] Holcombe, M. Mathematical models of cell biochemistry. Technical Report CS-86-4. 1986. Dept of Computer Science, Sheffield University, UK.

[8] Nayak, P. Pandurang, et al. Validating the DS1 Remote Agent Experiment. *Proc. 5th International Symposium on Artificial Intelligence, Robotics and Automation in Space (iSAIRAS-99).*

[9] Rouff, C., Rash, J., Hinchey, M. Experience Using Formal Methods for Specifying a Multi-Agent System. *Proc. 6th IEEE International Conference on Engineering of Complex Computer Systems (ICECCS 2000)*, Sept. 11-15, 2000.

[10] Rouff, C., Vanderbilt, A., Truszkowski, W., Rash, J. and Hinchey, M. Verification of NASA Emergent Systems. *Proc. 9th International Conference on Engineering. of Complex Computer Systems (ICECCS 2004*), Florence, Italy, April 14-16, 2004.

[11] Rouff, C., Vanderbilt, A., Hinchey, M. Truszkowski, W., and Rash, J. Properties of a Formal Method for Prediction of Emergent Behaviors in Swarm-based Systems. *Proc. 2nd IEEE International Conference on Software Engineering and Formal Methods*. Beijing, China, 26-30 September, 2004.

[12] Suhl, C. RT-Z: An Integration of Z and Timed CSP. *Proc. 1st International Conference on Integrated Formal Methods (IFM99*). York, UK, June 28-29, 1999.

[13] Sumpter, D.J.T., Blanchard, G.B. and Broomhead, D.S. Ants and Agents: A Process Algebra Approach to Modelling Ant Colony Behaviour. *Bulletin of Mathematical Biology*, 63, 951-980, 2001

[14] Tofts, C. Describing social insect behaviour using process algebra. *Transactions on Social Computing Simulation*, 227-283, 1991.