Properties of a Formal Method for Prediction of Emergent Behaviors in Swarm-based Systems

Christopher Rouff,
Amy Vanderbilt
SAIC
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

Abstract

Autonomous intelligent swarms of satellites are being proposed for NASA missions that have complex behaviors and interactions. The emergent properties of swarms make these missions powerful, but at the same time more difficult to design and assure that proper behaviors will emerge. This paper gives the results of research into formal methods techniques for verification and validation of NASA swarm-based missions. Multiple formal methods were evaluated to determine their effectiveness in modeling and assuring the behavior of swarms of spacecraft. The NASA ANTS mission was used as an example of swarm intelligence for which to apply the formal methods. This paper will give the evaluation of these formal methods and give partial specifications of the ANTS mission using four selected methods. We then give an evaluation of the methods and the needed properties of a formal method for effective specification and prediction of emergent behavior in swarm-based systems.

1. Introduction

Autonomous intelligent swarms of satellites are being proposed for missions that have complex behaviors and interactions. The emergent properties of swarms also make these missions powerful, but at the same time more difficult to design and assure that the proper behaviors will emerge. A significant challenge when verifying and validating swarms of intelligent interacting agents is how to determine that the possible exponential interactions and emergent behaviors are producing the desired results. Assuring correct behavior and interactions of swarms will be critical to mission success.

The Autonomous Nano Technology Swarm (ANTS) mission is an example of one of the swarm types of missions NASA is considering. Since the ANTS and other similar missions are going to consist of autonomous spacecraft who may be out of contact with the earth for extended periods of time, and have low bandwidths due to weight constraints, it will be difficult to observe improper behavior and to correct any errors after launch. One of the highest possible levels of assurance comes from formal methods [9]. 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. Multiple formal methods were evaluated to determine their effectiveness in modeling and assuring the behavior of swarms of spacecraft. The ANTS mission was used as an example of swarm intelligence for which to apply the formal methods.

In swarm simulations, a group of interacting agents [21] (often heterogeneous or near heterogeneous agents) are studied for their emergent behavior. In swarm simulations; each of the agents is given certain parameters that it tries to maximize. Bonabeau et al. [5, 6] who studied self-organization in social insects stated "that complex collective behaviors may emerge from interactions among individuals that exhibit simple behaviors" and described 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."

Intelligent swarms [3, 4, 7] are the use of swarms of simple intelligent agents using local interactions (interactions between agents and the environment). There is no central controller directing the swarm, they are self-organizing based on the emergent behaviors of the simple interactions. These types of swarms exhibit self-organization since there is no external force directing their behavior and no one agent has a global view of the intended macroscopic behavior.

One of the most challenging aspects of using swarms is how to verify that the emergent behavior of

such systems will be proper and that no undesirable behaviors will occur. In addition to emergent behavior in swarms, there are also a large number of concurrent interactions going on between the agents that make up the swarms. These interactions can also contain errors, such as race conditions, that are difficult to detect until they occur. Once they do occur, it can be difficult to recreate the errors since they are usually data and time dependent.

Verifying intelligent swarms are even more difficult since the swarms are no longer made up of homogeneous members with limited intelligence and communications. Intelligent swarms may from the beginning be made up of heterogeneous elements, reflecting different capabilities as well as a possible social structure. Verifying such swarms will be difficult due to the complexity of each member but also due to the complex interaction of a large number of intelligent elements. This will create a huge state space, and since the elements may be learning, the behavior of the individual elements and the emergent behavior of the swarm will be constantly changing and may be difficult to predict.

2. ANTS mission overview

The NASA Autonomous Nano-Technology Swarm (ANTS) mission [1, 2, 8, 10, 11] will be made up of swarms of autonomous pico-class (approximately 1kg) satellites that will explore the asteroid belt. There will be approximately 1,000 spacecraft involved in the mission consisting of several types (Figure 1). Approximately 80 percent of the spacecraft will be workers which will have a single specialized instrument onboard (e.g., a magnetometer, x-ray, gamma-ray, visible/IR, neutral mass spectrometer) and will obtain specific types of data.

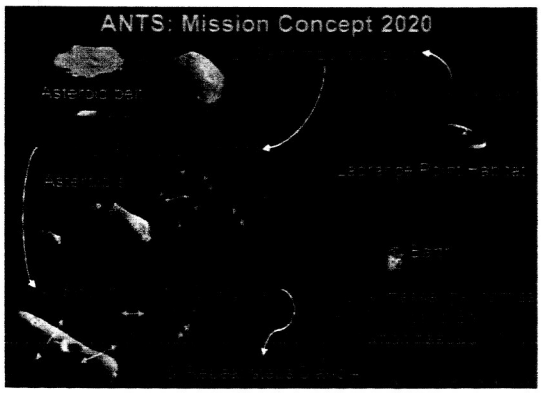


Figure 1. ANTS Mission Concept.

Some will be coordinators (called leaders) that have rules that decided the types of asteroids and data the mission is interested in and will coordinate the efforts of the workers. The third type of spacecraft are the messengers that will coordinate communications between the workers, leaders and Earth. Each worker spacecraft will examine asteroids they encounter and send messages back to a coordinator that will then evaluate the data and send other appropriate satellites with specialized instruments to the asteroid to gather further information.

To implement this mission a heuristic approach is being considered that provides for a social structure to the spacecraft based on the above hierarchy. Crucial to the mission will be the ability to modify its operations autonomously to reflect the changing nature of the mission and the distance and low bandwidth communications back to Earth.

Finding errors in missions like ANTS that contain large amount of parallel processes and distributed computing can be very difficult. Errors in these systems can rarely be found by inputting sample data into the system and checking if the results are correct. Errors in these systems tend to be time-based and only occur when processes send or receive data at particular times or in a particular sequence. 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, the state space grows extremely fast with the number of states in the processes, and becomes untestable with a relatively small number of processes. Traditionally, to get around the state explosion problem, testers have artificially reduced the number of states and approximated the underlying software using models.

To be able to effectively verify missions such as ANTS, new verification methods need to be developed [17]. To determine the properties needed for a formal method several current formal methods were surveyed and four were selected to specify part of the ANTS mission. The part of the ANTS mission that was specified is a virtual experiment that is conducted by a subset of the Leader spacecraft in the ANTS mission as well as the operation of an individual ANT Leader spacecraft. Supporting documentation for the specification is given in the documents titled "Protocol for ANTS Encounters" [1] and "Prospecting ANTS Missions: Applying a New Paradigm to Lunar and Planetary Exploration" [2] as well as papers describing the mission that are freely available on the ANTS mission web site at http://www.ants.gsfc.nasa.gov. These papers include "ANTS (Autonomous Nano Technology Swarm): An Artificial Intelligence Approach to Asteroid Belt Resource Exploration" [10], "Onboard Science Software Enabling Future Space Science and Space Weather Missions" [15],

"ANTS for the Human Exploration and Development of Space" [11], and "Describing Intelligent Agent Behavior" [14].

3. Comparison of formal methods

An initial survey of formal methods was done to determine methods that would be appropriate for specifying swarm-based systems. Based on the results of a survey [16, 17, 18], four methods were selected to do the partial specification of the ANTS mission (a previous specification of part of the ANTS mission was also done with BDI logic [14]). The four methods selected were CSP, WSCCS, Unity Logic and X-Machines and evaluated for effectiveness in analyzing emergent behavior in the ANTS swarm. Each of the methods was evaluated for its effectiveness for specifying swarm-based systems and analyzing any emergent behavior of the mission.

The following first gives the partial specifications of the ANTS mission using CSP, WSCCS, Unity Logic and X-Machines. We then offer an evaluation of the methods and conclusions concerning the properties of a formal method needed for effective specification and prediction of emergent behavior in swarm-based systems. Due to space requirements only samples of the specifications are given and the reader is referred to [16] for the entire specifications.

3.1. CSP specification of ANTS

The following is a specification of the behavior of the NASA ANTS mission using Communicating Sequential Processes (CSP) [12, 13]. In the specification, each of the spacecraft has goals to fulfill their mission. The aggregate or emergent behavior of all these goals should equal the goals of the mission. The following is the top-level specification of the ANTS mission:

$$ANTS_{goals} = Leader_{i,l\_goals} \parallel$$
$$Messenger_{j,m\_goals} \parallel Worker_{k,w\_goals}$$
$$\bullet\, 1 \le i \le m,\ 1 \le j \le n,\ 1 \le k \le p$$

where m is the number of leader spacecraft, n the number of messenger spacecraft and p the number of worker spacecraft. The ANTS mission starts, or is initialized, with a set of goals given to it by the principal investigator and part of these goals are given to the leader (some of these goals may not be given to the leader because the goals are ground based or not applicable to the leader). In addition to goals, each of the spacecraft is given a name (in this case in the form of a number) so that it can identify itself when communicating with other ANTS spacecraft and the Earth. The following gives a partial specification for a leader.

The leader spacecraft specification consists of two processes, the communications process and the intelligence process:

$$Leader_i = LEADER\_COM_{i,l,l} \parallel$$
$$LEADER\_INTELLIGENCE_{i,goals,model}$$

The communication process, *LEADER_COM*, specifies the behavior of the spacecraft as it relates to communicating with the other spacecraft and Earth. 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 and the maintenance of the goals for the leader is done. In addition to the goals, the *LEADER_INTELLIGENCE* process also maintains the models of the spacecraft and its environment and specifies how it is modified during operations. Each of the above processes has parameters that have an identifying number that indicates which spacecraft of a group it is, as well as other parameters that are sets that store conversations, goals and models. Since at startup there have been no conversations, the conversation set in the LEADER_COM process is empty. Since leaders are given initial goals and models, these sets are non-empty at start up. The following is the top level specification of the leader communication.

$$LEADER\_COM_{i,conv} = leader.in?msg \rightarrow$$
$$\text{case } LEADER\_MESSAGE_{i,conv,msg}$$
$$\quad if \text{ sender}(msg) = LEADER$$
$$\quad MESSENGER\_MESSAGE_{i,conv,msg}$$
$$\quad if \text{ sender}(msg) = MESSENGER$$
$$\quad WORKER\_MESSAGE_{i,conv,msg}$$
$$\quad if \text{ sender}(msg) = WORKER$$
$$\quad EARTH\_MESSAGE_{i,conv,msg}$$
$$\quad if \text{ sender}(msg) = EARTH$$
$$\quad ERROR\_MESSAGE_{i,conv,msg}$$
$$\quad otherwise$$

The above shows the messages from other spacecraft types that a leader may receive. Messages sent from another leader may be one of two types: requests or informational. For requests, the requests may be for such things as information on the leader's model or goals, for resources (e.g., more workers), or for status. Messages may also be informational and contain data containing new goals or new information for the agent's model (due to a new discovery or a message from Earth). This information needs to be

examined by the intelligence process and the model process to determine if any updates to the goals or model needs to be made. The following processes further describe the messages that may be received from other leaders.

$$
\begin{aligned}
&LEADER\_MESSAGE_{i,conv,msg} = \\
&\quad \text{case } LEADER\_INFORMATION_{i,conv,msg} \\
&\qquad \textit{if } \text{content}(msg) = information \\
&\qquad LEADER\_REQUESTS_{i,conv,msg} \\
&\qquad \textit{if } \text{content}(msg) = request \\
&\qquad LEADER\_RECEIVE_{i,conv,msg} \\
&\qquad \textit{if } \text{content}(msg) = reply\_to\_request \\
&\qquad ERROR\_MESSAGE_{i,conv,msg} \\
&\qquad \text{otherwise}
\end{aligned}
$$

The following gives additional information on the leader information messages.

$$
\begin{aligned}
&LEADER\_INFORMATION_{i,conv,msg} = \\
&\quad leader\_model_i!(NEW\_INFO, msg) \\
&\quad \rightarrow goals\_channel_i!(NEW\_INFO, msg) \\
&\quad \rightarrow LEADER\_COM_{i,conv}
\end{aligned}
$$

If the message is new information, then that information has to be sent to the deliberative part of the agent to check if the goals should be updated as well as the model part to check if any of the information requires updates to the model.

$$
\begin{aligned}
&LEADER\_REQUESTS_{i,conv,msg} = \\
&\quad \text{case } LEADER\_STATUS\_REQ \\
&\qquad \textit{if } \text{content}(msg) = status\_request \\
&\qquad LEADER\_INFO\_REQ_{i,conv,msg} \\
&\qquad \textit{if } \text{content}(msg) = info\_request \\
&\qquad LEADER\_RESOURCE\_REQ_{i,\,conv,\,msg} \\
&\qquad \textit{if } \text{content}(msg) = resource\_request \\
&\qquad ERROR\_MESSAGE_{i,conv,msg} \\
&\qquad \text{otherwise}
\end{aligned}
$$

If the message is a request, then depending on the type of request different processes are executed. Requests from others may be for status of the spacecraft, requests for information on the leader's goals or model, or it could be a request for resources, such as some workers under the leader's direction to form a sub-team to investigate a particular asteroid or the need for a messenger to be relocated to perform communication functions.

3.2. WSCCS

To model the ANTS Leader spacecraft, WSCCS [19, 20], 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
- The relative frequency of each action for the agent
- The priority of each action for that agent

Consider the following actions, agent states and view of frequency, f, and priority, p, on the actions of the Leader as seen in the table below:

Agent State	Actions leading to the agent state	f	p
	Identity		
Communicating	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

Based on this information, WSCCS provides an algebra by which the behavior of the Leader can be studied and verified [20, 19]. Given the information from the table above, we define the agent-states as

$$
\begin{aligned}
Communicating \equiv\ & 50\omega^2 : \text{SendMessageWorker}.Communicating \\
& + 50\omega^2 : \text{SendMessageLeader}.Communicating \\
& + 1\omega^1 : \text{SendMessageError}.Communicating \\
& + 50\omega^2 : \text{ReceiveMessageWorker}.Communicating \\
& + 50\omega^2 : \text{ReceiveMessageLeader}.Communicating \\
& + 1\omega^1 : \text{ReceiveMessageError}.Communicating \\
& + 50\omega^2 : \text{ReasoningDeliberatve}.Reasoning \\
& + 50\omega^2 : \text{ReasoningReactive}.Reasoning \\
& + 17\omega^2 : \text{ProcessingSortingAndStorage}.Processing \\
& + 17\omega^2 : \text{ProcessingGeneration}.Processing \\
& + 17\omega^2 : \text{ProcessingPrediction}.Processing \\
& + 16\omega^2 : \text{ProcessingDiagnosis}.Processing \\
& + 16\omega^2 : \text{ProcessingRecovery}.Processing \\
& + 17\omega^2 : \text{ProcessingRemediation}.Processing
\end{aligned}
$$

The symbol + in this notation denotes that the Communicating Leader will make a choice between

the various allowed actions, and that that choice will be made based on the frequencies and priorities of each allowable action. For example, the Communicating leader may choose to remain in the Communicating state by choosing to send a message to a worker. It would do so with a frequency of 50 and a priority of 2 which tells us that it will make this choice with a probability of 12.5%. The Communicating Leader may instead choose to transition to a Processing state by processing for Recovery. There is a 4% chance that the Leader will make this choice. What follows are similar statements for the Reasoning Leader and the Processing Leader:

$$\begin{aligned}
Reasoning \equiv\ & 50\omega^3 : \text{ReasoningDeliberatve.}Reasoning \\
& + 50\omega^3 : \text{ReasoningReactive.}Reasoning \\
& + 50\omega^2 : \text{SendMessageWorker.}Communicating \\
& + 50\omega^2 : \text{SendMessageLeader.}Communicating \\
& + 1\omega^1 : \text{SendMessageError.}Communicating \\
& + 50\omega^2 : \text{ReceiveMessageWorker.}Communicating \\
& + 50\omega^2 : \text{ReceiveMessageLeader.}Communicating \\
& + 1\omega^1 : \text{ReceiveMessageError.}Communicating \\
& + 17\omega^2 : \text{ProcessingSortingAndStorage.}Processing \\
& + 17\omega^2 : \text{ProcessingGeneration.}Processing \\
& + 17\omega^2 : \text{ProcessingPrediction.}Processing \\
& + 16\omega^2 : \text{ProcessingDiagnosis.}Processing \\
& + 16\omega^2 : \text{ProcessingRecovery.}Processing \\
& + 17\omega^2 : \text{ProcessingRemediation.}Processing
\end{aligned}$$

In the above definition of the Reasoning Leader, we see that the Leader will not choose to send or receive a message in error since the priorities of these actions are lower than the priorities of other actions.

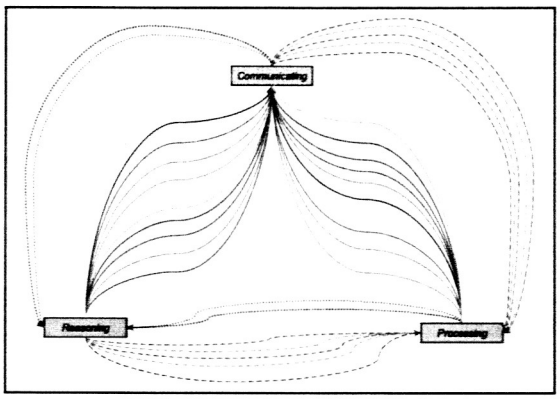
$$\begin{aligned}
Processing \equiv\ & 17\omega^2 : \text{ProcessingSortingAndStorage.}Processing \\
& + 17\omega^2 : \text{ProcessingGeneration.}Processing \\
& + 17\omega^2 : \text{ProcessingPrediction.}Processing \\
& + 16\omega^2 : \text{ProcessingDiagnosis.}Processing \\
& + 16\omega^2 : \text{ProcessingRecovery.}Processing \\
& + 17\omega^2 : \text{ProcessingRemediation.}Processing \\
& + 50\omega^3 : \text{ReasoningDeliberatve.}Reasoning \\
& + 50\omega^3 : \text{ReasoningReactive.}Reasoning
\end{aligned}$$

This statement shows that the Processing Leader is forced to go into the Reasoning state prior to entering the Communication State to ensure that the Leader has reasoned about its mission goals and model after processing and before communicating to other members of the swarm.

The operations of choice (+) and composition of actions (*) are then defined by the following rules:

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

A transitional semantics defines what series of actions are valid for a given agent, and allows us to interpret agents as finite state automations represented by a transition graph. A transition graph derived from these transitions for the ANTS Leader Spacecraft is shown below. (Nodes represent the agents and the edges between (color-coded to save space) represent the weights and actions.)



3.2.1. Emergent Behavior of a Swarm of Leaders Using Probability. 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 a composition of n actions, denoted with weight $m_1\omega^{k_1} * m_2\omega^{k_2} * \ldots * m_n\omega^{k_n}$, on each time step. When this happens, the n-leader swarm still must behave according to the rules for composition seen earlier. 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 and thus we can narrow down that each priority k_i must be either 1 or 2 with each relative frequency m_i either 1 (if the priority is 1) or one of 16, 17 or 50 (if the priority is 2).

Any composition which includes any leader communicating in error will have a priority less than the priority of not sending any messages in error and thus the swarm will not choose to send or receive a message in error. Thus the remaining options for leaders in the swarm will include communicating (not in error), 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 N_{reason} be the number of leaders in the swarm who choose to reason on that time step. Let $N_{process16}$ be the number of leaders in the swarm who choose to process (by prediction or recovery) on that time step. Lastly, let $N_{process17}$ be the number of leaders in the swarm who choose to process (by other means) on that time step.

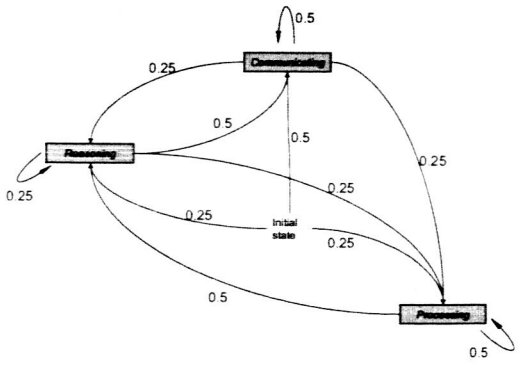
Then, each action by each leader will have priority 2 and relative frequency 16, 17 or 50. Thus, the composition of their actions will have weight

$$m_1\omega^{k_1} * m_2\omega^{k_2} * \ldots * m_n\omega^{k_n} =$$

$$(50^{N_{comm}+N_{reason}})(16^{N_{process16}})(17^{N_{process17}})\omega^{2n}$$

From this weighting, we can see that drastically higher frequencies exist when a larger number of leaders in the swarm choose to communicate or reason. Much lower frequencies exist when larger numbers of leaders choose to process. Thus the swarm will be communicating and reasoning much more often than processing, although processing will take place.

3.2.2. Emergent Behavior of a Leader Using Markov Chains. Using Markov Chains to get a different view of the Leader's emergent behavior, we find the following diagram and results:



Based on these statements and the previous frequencies and priorities, we can calculate the probabilities for the Leader choosing each action and therefore the probabilities that the Leader will transition to one state or another. From these probabilities we can construct the following matrix, P, which for each entry p_{ij} shows the probability of the Leader choosing to transition from state i to state j. For example p_{13} = 0.25 which means that the probability of transitioning from state 1 (Initial state or Identity State) to state 3 (Processing) is 25%.

$$P = \begin{pmatrix} 0 & .5 & .25 & .25 \\ 0 & .5 & .25 & .25 \\ 0 & .5 & .25 & .25 \\ 0 & 0 & .5 & .5 \end{pmatrix}$$

Given this matrix, we can calculate the various powers, P^n, of the matrix. The n^{th} power of the matrix P will tell us the probabilities which state the Leader will be in on the n^{th} time step. For example, considering the following results:

> P^2;

0.	.375000000000000000	.312500000000000000	.312500000000000000
0.	.375000000000000000	.312500000000000000	.312500000000000000
0.	.375000000000000000	.312500000000000000	.312500000000000000
0.	.250000000000000000	.375000000000000000	.375000000000000000

We see in the matrix for P^2 that the entry P^2_{42} is 0.25. This tells us that if the Leader begins in the fourth state (Processing), it has a probability of 25% of being in the second state (Communicating) on the second time step. Observe the convergence of these matrices at higher powers – i.e. as time goes on:

> P^8;

0.	.333343505859375000	.333328247070312500	.333328247070312500
0.	.333343505859375000	.333328247070312500	.333328247070312500
0.	.333343505859375000	.333328247070312500	.333328247070312500
0.	.333312988281250000	.333343505859375000	.333343505859375000

> P^10000000000;

0.	.333333333333333370	.333333333333333370	.333333333333333370
0.	.333333333333333370	.333333333333333370	.333333333333333370
0.	.333333333333333370	.333333333333333370	.333333333333333370
0.	.333333333333333370	.333333333333333370	.333333333333333370

>P^10000000000000000000000000000000000000000000000000000000000000000000000000000;

0.	.333333333333333370	.333333333333333370	.333333333333333370
0.	.333333333333333370	.333333333333333370	.333333333333333370
0.	.333333333333333370	.333333333333333370	.333333333333333370
0.	.333333333333333370	.333333333333333370	.333333333333333370

We see the powers of P converging to the matrix

$$P^n = \begin{pmatrix} 0 & 1/3 & 1/3 & 1/3 \\ 0 & 1/3 & 1/3 & 1/3 \\ 0 & 1/3 & 1/3 & 1/3 \\ 0 & 1/3 & 1/3 & 1/3 \end{pmatrix}$$

where the Leader will not return to the initial state but will have equal probability of being in any of the three other states given a starting point of any of the four states. This is just an example of the type of prediction that Markov Chains may be able to deliver. These concepts are currently being further studied.

3.3. Unity Logic

To model the ANTS Leader spacecraft with Unity Logic [23], we consider states of the Leader just as in

Table 1. Leader States and Transitions

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

WSCCS and other state - machine based specification languages. In Unity Logic, we will consider the states of the Leader and the actions taken to make the Leader be in those states, but the notation will appear much closer to classical logic. Predicates are 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. This allows us to formally specify the Leader using ***assertions*** such as the following:

[Communicating]ReasoningDeliberatve(Leader)[Reasoning]

[Reasoning] SendMessage (Leader,Worker)[Communicating]

[Processing] SendMessage (Leader, Worker) [Communicating]

Unity Logic then provides a logical syntax equivalent to 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.

3.4. X-Machines

To model the ANTS Leader spacecraft as an X-Machine [22] we must define it as a tuple:

$L = \{Input, Memory, Output, Q, \Phi, F, start, m_0\}$ where the components of the tuple are defined as:

$Input =$

$$\left\{ \begin{array}{l} worker, messenger, leader, error, \\ Deliberative, Reactive, SortAndStore, \\ Generate, Predict, Diagnose, Recover, \\ Remediate \end{array} \right\}$$

is a set of data. $Memory$ will be 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 will be denoted by $(Goals_0, Model_0)$. When the goals and/or model changes, the new tuple will be denoted as $m' = (Goals', Model')$.

$Output =$

$$\left\{ \begin{array}{l} SentMessageWorker, \\ SentMessageMessenger, \\ SentMessageLeader, SentMessageError, \\ ReceivedMessageWorker, \\ ReceivedMessageMessenger, \\ ReceivedMessageLeader, \\ ReceivedMessageError, \\ ReasonedDeliberatively, \\ Reasoned Reactively, \\ ProcessedSortingAndStoring, \\ ProcessedGeneration, Processed Prediction, \\ ProcessedDiagnosis, Processed Recovery, \\ Processed Remediation \end{array} \right\}$$

is another set of data.

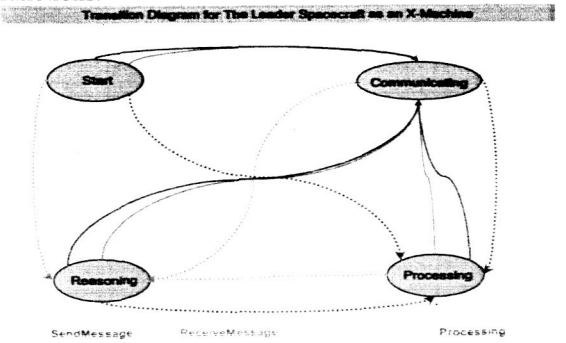
$Q = \left\{ \begin{array}{l} Start, Communicating, Reasoning, \\ Processing \end{array} \right\}$ is a set of states.

$\Phi = \left\{ \begin{array}{l} SendMessage, ReceiveMessage, \\ Reason, Process \end{array} \right\}$ 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 a next-state partial function defined according to definitions such as in Table 1.

A transition diagram for the ANTS Leader Spacecraft is shown below. (Nodes represent the states and the edges between represent the transition functions.



3.5. Evaluation of methods

CSP is a process algebra and is very good at specifying the process protocols between and within the spacecraft and analyzing the result for race conditions. Being able to evaluate a system for race conditions is very important in systems, particularly swarm-based systems which are highly parallel. From a CSP specification reasoning, about the specification can be done to determine race conditions as well as converted into a model checking language for running on 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 a 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 affecting 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	◆ Actions are seen as predicates

Logic	(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

4. Conclusion

Based on these properties, the experiences of creating partial specifications for the ANTS Leader Spacecraft, and the needs of the ANTS mission, we draw the following conclusions about the properties needed for effective specification and emergent behavior prediction of the ANTS mission.

An effective formal method must be able to predict the emergent behavior of 1000 agents as a swarm as well as the behavior of the individual agent. Crucial to the mission will be the ability to modify operations autonomously to reflect the changing nature of the mission and the distance and low bandwidth communications back to Earth. 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 to be developed must be able to be used to prove properties of the system correct (e.g., the underlying system will go from one state to another or not into a specific state), check for particular types of errors (e.g. race conditions), as well as be used as input to a model checker.

From this we can see that the formal method must be able to track the models of the leaders and it must allow for decisions to be made as to when the data collected has met the goals. The ANTS mission details are still being determined and are changing as more research is done. Therefore, the formal method must be flexible enough to allow for efficient changes and re-prediction of emergent behavior.

Bearing all of this in mind, the following table summarizes the properties necessary for effective specification and emergent behavior prediction of the ANTS swarm and other swarms, and looks to the existing formal methods to provide some of the desired properties.

Property (Existing Method?) - Notes
Specify processes (X-Machines, CSP) - Processes can be specified using the various manifestations of transition functions. This property could also be more robust.
Specify reasoning (Unity Logic) - Unity Logic provides only limited capability in this area. Other forms of possibly non-standard logics may need to be employed here to allow for intelligent reasoning with uncertain and possibly conflicting information.
Specify how an agent will choose between action alternatives (WSCCS) - A modified version of this ability from WSCCS may be used to supply an algebra for choosing between possible actions.
Support asynchronous messaging (CSP Variant) - Messaging may not be synchronized upon or after implementation. There are variants of CSP that support asynchronous messaging.
Support message buffering (CSP Variant) - Message buffering may be needed due to the possibly asynchronous nature of messaging between members of the swarm. There are variants of CSP that support buffering.
Specify concurrent agent states for each spacecraft (WSCCS) - This ability is solidly in place and will require only an augmentation of notation.
Specify communication protocols between agents (CSP) - CSP allows for this as it stands.
Adaptable to programming (X-Machines, Unity Logic) - Any formal specification languages that are created will need to keep in mind the ease of converting the formal specification to programs and model checkers.
Provide a method for determining if the goals have been met (None) - The goals of each spacecraft are constantly under review. We will need to be able to specify a method by which the spacecraft will know when the goals have been met. A modification to X-Machines may be able to solve this since the goals could be tracked using X-Machines.
Provide a method for determining new goals (None) - Once goals are met, new goals must be formed. We need to be able to specify a method for forming these goals. Again, a modification to X-Machines may be best since X-Machines could be used to track the goals.
Ability to model check (CSP) - Model checking will prevent semantic inconsistencies in the specifications.

Track Models (X-Machines) - X-Machines have the ability to track the universe model in memory but need a more robust way to detail what the model is, how it is created and how it is modified.
Associate agent actions with priorities (WSCCS) - This ability is firmly in place.
Associate agent actions with expected frequencies (WSCCS) - This ability is firmly in place.
Ability to predict emergent behavior at individual and swarm levels (WSCCS) - Current WSCCS abilities are not robust enough for these purposes and will need to be enhanced by greater use of Probability, Markov Chains and/or Chaos Theory.

A blending of the above methods seems to be the best approach for specifying swarm-based systems. Blending 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 methods is currently being performed.

5. Acknowledgments

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

6. References

[1] ANTS team. Protocol for ANTS Encounters. NASA GSFC, Code 695.

[2] ANTS Prospecting Asteroids Mission (ANTS/PAM). NASA Goddard Space Flight Center. http://ants.gsfc.nasa.gov/.

[3] Beni, G. The Concept of Cellular Robotics. In Proceedings of the 1988 IEEE International Symposium on Intelligent Control, pp 57-62, Los Alamitos, CA 1988. IEEE Computer Society Press.

[4] Beni, G. and Want, J. Swarm Intelligence. In Proceedings of the Seventh Annual Meeting of the Robotics Society of Japan, pp 425-428, Tokyo, Japan, 1989, RSJ Press.

[5] Bonabeau, E. and Theraulaz, G., Swarm smarts. Scientific American, pp. 72-79, March 2000.

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

[7] Bonabeau, E., Dorigo, M. and Theraulaz, G. Swarm Intelligence: From Natural to Artificial Systems. Oxford University Press, New York, 1999.

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

[9] Clare, E. and Wing, J. Formal Methods: State of the Art and Future Directions. Report by the Working Group on Formal Methods for the ACM Workshop on Strategic Directions in Computing Research, ACM Computing Surveys, vol. 28, no. 4, December 1996, pp. 626-643.

[10] 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. International Astronautical Federation, 51st Congress, October 2000.

[11] Curtis, S., Truszkowski, W., Rilee, M., and Clark, P. ANTS for the Human Exploration and Development of Space. IEEE Aerospace Conference, 2003.

[12] Hinchey, M. Jarvis, S. Concurrent Systems: Formal Development in CSP. McGraw-Hill. 1995.

[13] Hoare, C.A.R. Communicating Sequential Processes. Prentice Hall, 1985.

[14] Iyengar, J., Truszkowski, W and Mills, F. Describing Intelligent Agent Behaviors. Journal of International Information Management, 10(2), 99-77. 2002.

[15] Rilee, M.L., Boardsen, S.A., Bhat, M.K. and Curtis, S.A. Onboard Science Software Enabling Future Space Science and Space Weather Missions. 2002 IEEE Aerospace Conference Big Sky, Montana, March 2002.

[16] Rouff, C., Vanderbilt, A., Truszkowski, W., Rash, J., and Hinchey, M. Formal Approaches to Intelligent Swarm Technology, A Survey of Formal Methods for Intelligent Swarms. Technical Report. 2003. http://ants.gsfc.nasa.gov.

[17] Rouff, C., Truszkowski, W., Rash, J, and Hinchey, M. Formal Approaches to Intelligent Swarms. 28th NASA/ IEEE Software Engineering Workshop, Dec. 2003.

[18] Rouff, C., Vanderbilt, A., Truszkowski, W., Rash, J, and Hinchey, M. Verification of NASA Emergent Systems. The 9th IEEE International Conference on Engineering of Complex Computer Systems, April 2004.

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

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

[21] Weiss, G. Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence. The MIT Press. 1999.

[22] Holcombe, W.M.L. X-machines as a Basis for System Specification. Software Engineering Journal, 3(2), 1988, 69-76.

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