International Journal on Software Tools Technology Transfer manuscript No.
(will be inserted by the editor)

Christopher A. Rouff · Michael G. Hinchey · Walter F. Truszkowski · James L. Rash

Experiences applying Formal Approaches in the Development of Swarm-Based Space Exploration Systems

Abstract NASA is researching advanced technologies for future exploration missions using intelligent swarms of robotic vehicles. One of these missions is the Autonomous Nano Technology Swarm (ANTS) mission that will explore the asteroid belt using 1,000 cooperative autonomous spacecraft. The emergent properties of intelligent swarms make it a potentially powerful concept, but at the same time more difficult to design and ensure that the proper behaviors will emerge. NASA is investigating formal methods and techniques for verification of such missions. The advantage of using formal methods is the ability to mathematically verify the behavior of a swarm, emergent or otherwise. Using the ANTS mission as a case study, we have evaluated multiple formal methods to determine their effectiveness in modeling and ensuring desired swarm behavior. This paper discusses the results of this evaluation and proposes an integrated formal method for ensuring correct behavior of future NASA intelligent swarms.

Keywords Validation · Verification · Formal Methods · Swarm

1 Introduction

NASA is investigating new paradigms for future space exploration, heavily focused on the (still) emerging technologies of autonomous and autonomic systems [45,46]. Traditional missions, reliant on one large spacecraft, are being replaced with missions that involve smaller collaborating spacecraft, analogous to swarms in nature [13]. This approach offers several advantages: the ability to send spacecraft to explore regions of space where traditional craft simply would be impractical, greater redundancy (and, consequently, greater protection of assets), and reduced costs and risk, to name but a few. Concept swarm missions entail the use of unmanned autonomous vehicles (UAVs) flying approximately one meter above the surface of Mars, which will cover as much of the surface of Mars in three seconds as the now famous Mars rovers did in their entire time on the planet; the use of armies of tetrahedral walkers to explore the Mars and Lunar surface [12]; constellations of satellites flying in formation; and, the use of miniaturized spacecraft to explore the asteroid belt, where heretofore it has been impossible to send exploration craft without the high likelihood of loss [13].

These new approaches to exploration simultaneously pose many challenges. The missions will be unmanned and highly autonomous. They will also exhibit the properties of autonomic systems, being self-protecting, self-healing, self-configuring, and self-optimizing. Many of these missions will be sent to parts of the solar system where manned missions are simply not possible, and to where the round-trip delay for communications to spacecraft exceeds 40 minutes, meaning that the decisions on responses to exploration opportunities as well as problems and undesirable situations must be made *in situ* rather than from ground control on Earth.

The degree of autonomy and intelligence that such missions will possess would require a prohibitive amount of testing. Furthermore, with learning and autonomic properties, such as self-optimizing and self-healing, emergent behavior patterns simply cannot be fully predicted. The authors believe that formal specification techniques and formal verification will play important roles in the future development of NASA space exploration missions. Formal methods will likely play a major role in the specification and analysis of forthcoming missions, enabling software assurance and proof of correctness of the be-

Christopher A. Rouff
SAIC
Intelligent Systems Division
McLean, VA USA
E-mail: rouffc@saic.com

Michael G. Hinchey, Walter F. Truszkowski, James L. Rash
NASA Goddard Space Flight Center
Information Systems Division
Greenbelt, MD, USA
E-mail: {michael.g.hinchey, walter.f.truszkowski, james.l.rash}@nasa.gov

havior of the swarm, even (it is projected) when this behavior is emergent (as a result of composing a large number of interacting entities, producing behavior that was not foreseen). Formal models derived may also be used as the basis for automating the generation of much of the code for the mission [20].

To address the challenge in verifying the above missions, a NASA project, Formal Approaches to Swarm Technology (FAST), is investigating appropriate formal methods for use in such missions, and is beginning to apply these techniques to specifying and verifying parts of a NASA swarm-based concept mission. The remainder of this paper gives an overview of swarm technologies and the ANTS swarm-based mission, presents the results of evaluation of a number of formal methods for verifying swarm-based missions, and proposes an integrated formal method for verifying swarm-based systems.

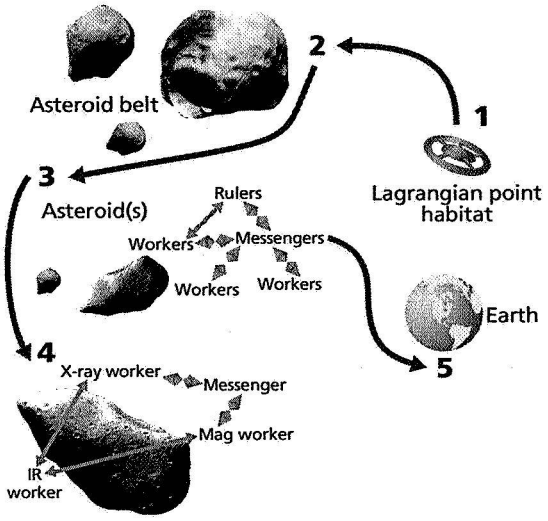

Fig. 1 ANTS Mission Concept

2 Swarm Technologies

Swarms [1,2] consist of a large number of simple agents that have local interactions (between each other and the environment). There is no central controller directing the swarm and no one agent has a global view; they are self-organizing based on the emergent behaviors of the simple interactions. This type of behavior is observed in insects and flocks of birds. Bonabeau et al. [6], 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 emergent behavior is sometimes referred to as the macroscopic behavior and the individual behavior and local interactions as the microscopic behavior.

Agent swarms are often used as a modeling technique and as a tool to study complex systems [16]. In swarm simulations, a group of interacting agents [48] (often homogeneous or near homogeneous agents) is studied for their emergent behavior. Examples of the use of swarm simulations includes studying flocks of birds [9,30], business and economics [27], and ecological systems [37]. In swarm simulations, each of the agents is given certain parameters that it tries to maximize. In terms of bird swarms, each bird tries to find another bird to fly with, and then fly off to one side and slightly higher to reduce its drag. Eventually the birds form flocks. Swarms are also being investigated for use in applications such as telephone switching, network routing, data categorizing, and shortest path optimizations [5].

Intelligent swarm technology is where the individual members of a swarm also exhibit intelligence [4,5]. With intelligent swarms, members may be heterogeneous or homogeneous. Even if members start out as homogeneous, due to their differing environments they may learn different things, develop different goals, and thereby become a heterogeneous swarm. Intelligent swarms may also from the beginning be made up of heterogeneous elements, such as the NASA concept mission described below, reflecting different capabilities as well as a possible social structure. This makes verifying such systems even more difficult since the swarms are no longer made up of homogeneous members with limited intelligence and communications.

The remainder of this section gives an overview of the NASA ANTS concept swarm-based mission. We are using the ANTS mission as an example test-bed and case study, for the purpose of evaluating multiple formal methods in the specification, validation, and verification of swarm-based missions.

2.1 ANTS Mission Overview

The Autonomous Nano-Technology Swarm (ANTS) concept mission [12–14] will involve the launch of a swarm of autonomous pico-class (approximately 1kg) spacecraft that will explore the asteroid belt for asteroids with certain scientific characteristics. Figure 1 gives an overview of the ANTS mission [45]. In this mission, a transport ship, launched from Earth, will travel to a point in space where net gravitational forces on small objects (such as pico-class spacecraft) are negligible (a Lagrangian point). From this point, 1000 spacecraft, that have been manufactured en route from Earth, will be launched into the asteroid belt. The asteroid belt presents a large risk of destruction for large (traditional) spacecraft. Even with pico-class spacecraft, 60 to 70 percent of them are expected to be lost. Because of their small size, each spacecraft will carry just one specialized instrument for collecting a specific type of data from asteroids in the belt.

To implement this mission, a heuristic approach is being considered that provides for a social structure to the spacecraft that uses a hierarchical behavior analogous to colonies or swarms of insects, with some spacecraft directing others. Artificial intelligence technologies such as genetic algorithms, neural nets, fuzzy logic, and on-board planners are being investigated to assist the mission to maintain a high level 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 distance and low bandwidth communications back to Earth. Approximately 80 percent of the spacecraft will be workers that will carry the specialized instruments (e.g., a magnetometer, x-ray, gamma-ray, visible/IR, neutral mass spectrometer) and will obtain specific types of data. Some will be coordinators (called rulers or leaders) that have rules that decided the types of asteroids and data the mission is interested in and that will coordinate the efforts of the workers. The third type of spacecraft are messengers that will coordinate communication between the rulers and workers, and communications with the mission control center on Earth.

The swarm will form sub-swarms, each under the control of a ruler, which contains models of the types of science that are to be pursued. The ruler will coordinate workers, each of which uses its individual instrument to collect data on specific asteroids and feeds this information back to the ruler, who will determine which asteroids are worth examining further. If the data matches the profile of a type of asteroid that is of interest, an imaging spacecraft will be sent to the asteroid to ascertain the exact location and to create a rough model to be used by other spacecraft for maneuvering around the asteroid. Other teams of spacecraft will then coordinate to finish mapping the asteroid to form a complete model.

2.2 Autonomic Properties of ANTS

The ANTS mission will exhibit almost total autonomy. The mission will also exhibit all of the properties required of an autonomic system [25,40,42]:

- Self-Configuring: ANTS resources must be fully configurable to support concurrent exploration and examination of hundreds of asteroids. Resources must be configured at both the swarm and team (sub-swarm) levels, in order to coordinate science operations while simultaneously maximizing resource utilization.
- Self-Optimizing: Rulers self-optimize primarily through learning and improving their ability to identify asteroids that will be of interest. Messengers self-optimize through positioning themselves appropriately. Workers self-optimize through learning and experience. Self-optimization at the system level propagates up from the self-optimization of individuals.
- Self-Healing: ANTS must self-heal to recover from damage due either to solar storms or to collision with asteroids or (possibly) other ANTS spacecraft. Loss of a particular type of instrument may require the launch of a replacement from Earth; loss of a ruler or messenger may involve a worker being "upgraded" to fulfill that role. Additionally, loss of power may require a worker to be killed off [41].
- Self-Protecting: In addition to protection from collision with asteroids and other spacecraft, ANTS teams must protect themselves from solar storms, where charged particles can degrade sensors and electronic components, and destroy solar sails (the ANTS spacecrafts' sole source of power). ANTS teams must replan their trajectories, or, in worst-case scenarios, must go into "sleep" mode to protect their sails.

2.3 Specifying and Verifying ANTS

The above is a very simplified description of the ANTS mission. For a more detailed exposition, the interested reader is directed to [34,45], or to the ANTS web-site. But, as can be seen from the brief exposition above, ANTS is a highly complex system that poses many significant challenges. Not least amongst these are the complex interactions between heterogeneous components, the need for continuous re-planning, re-configuration and re-optimization, the need for autonomous operation without intervention from Earth, and the need for assurance of the correct operation of the mission.

In missions such as ANTS that will be highly autonomous and out of contact with ground control for extended periods of time, errors in the software may not be observable or correctable after launch. Because of this, a high level of assurance is necessary for these missions before they are launched. Testing of space exploration systems is done through simulations since it would be impractical or impossible to test them in their end environment. Although these simulations are of very high quality, often very small errors get through and can result in the loss of an entire mission, as is thought to have happened with the Mars Polar Lander Mission [3,10], where the absence of one line of code may have resulted in the loss of the entire mission. In the report on the loss of the Mars Polar Lander [10], it is stated that

> it is important to recognize that space missions are a "one strike and you are out" activity. Thousands of functions can be correctly performed and one mistake can be mission catastrophic.

and

> A thorough test and verification program is essential for mission success.

Complex missions such as ANTS exacerbate the difficulty of finding errors, and will require new mission verification methods to provide the level of software assurance that NASA requires to reduce risks to an acceptable

level. Many of the ANTS behaviors, including those that produce race conditions, for example, are time-based and only occur when processes send or receive data at particular times or in a particular sequence, or after learning occurs. Errors under such conditions can rarely be found by inputting sample data and checking for correct results. To find these errors through testing, the software processes involved would 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 (and sometimes factorial) to the number of states, it 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 of the system and approximated the underlying software using models. This reduces the fidelity of the model and can mask potential errors.

A significant issue for specifying (and verifying) swarm behavior is support for analysis of and identification of emergent behavior. The idea of emergence is well known from biology, economics, and other scientific areas. It is also prominent in computer science and engineering, but the concept is not so well understood by computer scientists and engineers, although they encounter it regularly. Emergent behavior has been described as "system behavior that is more complex than the behavior of the individual components ... often in ways not intended by the original designers [29]." This means that when interacting components of a system whose individual behavior is well understood are combined within a single environment, they can demonstrate behavior that can be unforeseen or not explained from the behavior of the individual components. The corollary of this is that making changes to components of a system of systems, or replacing a sub-system within a system of systems, may often have unforeseen, unexpected, and completely unexplained ramifications for both overall system behavior and the behavior of other subsystems.

3 Formal Methods

Formal methods are proven approaches for assuring the correct operation of complex interacting systems [17,18, 28,31]. Formal methods are mathematically-based tools and techniques for specifying and verifying systems. They are particularly useful for specifying complex parallel and distributed systems where the entire system is difficult for a single person to fully understand and when more than one person was involved in the development. With formal methods, we may propose that certain properties hold, and prove that they hold. In particular this is invaluable for properties that we cannot test on Earth. By its nature, a good formal specification can guide us to propose and verify certain behaviors (or lack of certain behaviors) that we would often not think of when using regular testing techniques. Moreover, if properly applied, and properly used in the development process, a good formal specification can guarantee the presence or absence of particular properties in the overall system well in advance of mission launch, or even implementation. Indeed, various formal methods offer the additional advantage of support for simulation, model checking and automatic code generation, making the initial investment well worth while.

It has been stated that formal analysis is not feasible for emergent systems due to the complexity and intractability of these systems, and that simulation is the only viable approach for analyzing emergence of a systems [7]. For NASA missions, relying on simulations and testing alone are not sufficient even for systems that are much simpler than the ANTS mission, as noted above. The use of formal analysis would complement the simulation and testing of these complex systems and would give additional assurance of their correct operation. Given that one mistake can be catastrophic to a system and result in the loss of hundreds of millions of dollars and years of work, development of a formal analysis tool, even at a great cost, could have huge returns even if only one mission is kept from failing.

Verifying emergent behavior is an area that has been addressed very little by formal methods, though there has been some work done in this area by computer scientist analyzing biological systems [22,43,44]. However, formal methods may provide guidance in determining possible emergent behaviors that must be considered. Formal methods have been widely used for test case generation to develop effective test cases. Similar techniques may be used with formal methods, not to generate a test plan, but to propose certain properties that might or might not hold, or certain emergent behaviors that might arise.

4 Formal Methods for Intelligent Swarms

The *Formal Approaches to Swarm Technologies*, or *FAST*, project has surveyed formal methods and formal techniques to determine whether existing formal methods, or a combination of existing methods, could be suitable for specifying and verifying swarm-based missions and their emergent behavior [32,36]. Various methods were surveyed based on a small number of criteria that were determined to be important in their application to intelligent swarms. These include:

- support for concurrency and real-time constraints;
- formal basis;
- (existing) tool support;
- past experience in application to agent-based and/or swarm-based systems;
- algorithm support.

A large number of formal methods that support the specification of one of, but not both, concurrent behavior and algorithmic behavior were identified. In addition,

there were a large number of integrated or combination formal methods that have been developed over recent years with the goal of supporting the specification of both concurrency and algorithms.

Table 1 illustrates part of the results of the survey for mainstream formal methods. Table 2 compares a number of the integrated or combination formal methods surveyed. Table 3 compares methods that, as reported in the literature, have been applied to modeling or specifying swarm-based systems (whether computer-based swarms, or real swarms in nature).

Although the survey identified a few formal methods that have been used to specify swarm-based systems, initially only two formal approaches were found that had been used to analyze the emergent behavior of swarms, namely Weighted Synchronous Calculus of Communicating Systems (WSCCS) [44] and Artificial Physics [38,39]. Since the survey was completed, two other approaches that may prove valuable in analyzing emergent behavior—CommUnity [15] and CSP2B [8]—have been brought to our attention, although we have not as yet identified their use with swarm technologies *per se*.

4.1 Experience specifying swarm behaviors

There has not been significant work on specifying swarm behavior. Interestingly, most of the work that has been reported in the literature has been related to specifying the behavior of swarms or colonies of insects, and has been performed by biologists with the assistance of computer scientists using modified formal methods. The following is a brief description of some specification techniques that have been used for specifying social, swarm, and emergent behavior:

- Weighted Synchronous Calculus of Communicating Systems (WSCCS), a process algebra, was used by Tofts to model social insects [44]. WSCCS was also used in conjunction with a dynamical systems approach for analyzing the non-linear aspects of social insects [43].
- X-Machines have been used to model cell biology [22, 23], and modifications, such as Communicating Stream X-Machines [24] also seem to have potential for specifying swarms.
- Dynamic Emergent System Modeling Language (DESML) [26], a variant of UML, has been suggested for use in modeling emergent systems.
- Cellular automata [47] have been used to model systems that exhibit emergent behavior (e.g., land use).
- Artificial Physics [38,39], which uses physics-based modeling to gauge emergent behavior, has been used to provide assurance for formation flying as well as other constraints on swarms.

Simulation approaches have also been investigated for determining emergent behavior, after which a modeling technique is used to model that behavior. Such approaches do not model emergent behavior *a priori*, instead only after the fact, and were not considered.

4.2 Evaluation of Specification Methods

Based on the results of the survey, four formal methods were selected to be used for sample specification of part of the ANTS mission. These methods were: the process algebras CSP [19,21] and WSCCS [43,44], X-Machines [24], and Unity Logic [11]. Table 4 describes some of the properties, advantages and disadvantages of the four selected methods, which were used to describe an ANTS virtual experiment, described in Section 5. CSP was chosen as a baseline specification method because the team has had significant experience and success [33,35] in specifying agent-based systems with CSP. WSCCS and X-Machines were chosen because they have already been used for specifying emergent behavior by others, apparently with some success. Unity Logic was also chosen because it had been successfully used for specifying concurrent systems and was a logic-based specification, which offered a contrast to the other methods.

DESML, Cellular Automata, Artificial Physics, and simulation approaches were not used even though they had been used for specifying or evaluating emergent behavior. DESML, though very interesting, was not used because it had not been used or evaluated outside of the thesis it was developed under (though we may be revisiting it at a future time). Cellular Automata were not selected because they did not have any built in analysis properties for emergent behavior and because they have been primarily used for simulating emergent systems (as described in the previous section). Though not used for the specification, it too may be revisited to examine its strengths. Artificial physics, which is very promising, was not selected because of the newness of the approach and because of the translation that must be done between physics and software behavior. Lastly, simulation techniques were not used due to the fact that verification cannot be undertaken using simulation. This is because there could be emergent or other undesirable behaviors occurring that are not visible or do not become apparent during a simulation, but may be there nonetheless. A formal technique is designed to find exactly these kinds of errors.

5 Specifications of the Virtual Experiment

A virtual experiment is conducted in the ANTS mission by an ANT subset consisting of a Leader spacecraft and individual worker spacecraft. Details of the operations of the ANTS mission can be found on the ANTS web page.

A scenario for the ANTS mission is based on the ANTS spacecraft targeting an asteroid on which to do an

Table 1 Comparison of candidate formal methods for intelligent swarms

Name	Concurrency Support	Algorithm Support	Tool Support	Formal Basis	Used in Agent-Based Specs.	Used in Swarm-Based Specs.
Artificial Physics	Yes	Yes	Yes	Yes (Mathematical)	Yes	Yes (limited)
B	No	Yes	Yes	Yes (Set Theory & Pred. Logic)	Yes	No
BDI Logic	Yes	No	Yes	Yes (Logic)	Yes	Yes (limited)
CSP	Yes	No	Yes	Yes (Algebraic)	Yes	No
Finite State Machines	No	Yes	Yes	Yes (Formal Lang.)	Yes	No
Game Theory	Yes	No	Yes	Yes (Mathematical)	Yes	Yes
I/O Automata	Yes	Yes	Yes	Yes (Formal Lang.)	Yes	No
KARO	Yes	No	Yes (limited)	Yes (Logic)	Yes	No
Mathematical Analysis	Yes	No	Yes	Yes (Mathematical)	Yes	Yes
Petri Nets	Yes	No	Yes	Yes	Yes	No
Pi Calculus	Yes	No	Yes	Yes (Algebraic)	Yes	No
Real Time Logic	Yes	No	Yes	Yes (Logic)	No	No
SCR	No	Yes	Yes	Yes (Formal Lang.)	No	No
Statecharts	Yes	No	Yes	No (Formal Lang.)	Yes	No
UML	Yes	Yes	Yes	No	Yes	No
X-Machines	No	Yes	No (limited)	Yes (Formal Lang.)	Yes	No
Z	No	Yes	Yes	Yes (Set Theory/ Pred. Calc.)	Yes	No

Table 2 Comparison of integrated formal methods

Name	Concurrency Support	Algorithm Support	Tool Support	Formal Basis	Used in Agent-Based Specs.	Used in Swarm-Based Specs.
Communicating X-Machines	Yes	Yes	No	Yes	Yes	Yes
CSP-OZ	Yes	Yes	No	Yes	Yes	No
Object-Z and Statecharts	Yes	Yes	No	Yes	Yes	No
Temporal B	Yes	Yes	No	Yes	Yes	No
Temporal Petri Nets	Yes	No	No	Yes	Yes	No
Timed Communicating Object Z	Yes	Yes	No	Yes	Yes	No
Timed CSP	Yes	No	Yes	Yes	Yes	No
ZCCS	Yes	Yes	No	Yes	Yes	No

Table 3 Comparison of formal methods used for swarm specifications

Name	Concurrency Support	Algorithm Support	Tool Support	Formal Basis	Emergent Behavior Analysis	Used in Swarm-Based Specs.
Cellular Automaton	Yes	Yes	Yes	Yes (FSM)	No	Yes
Com. X-Machines	Yes	Yes	No	Yes (Formal Lang.)	No	Yes
Unity Logic	Yes	No	Yes (limited)	Yes (Logic)	No	Yes
WSCCS	Yes	No	Some (Prob. Workbench)	Yes (Process Alg.)	Yes (Markov Chain)	Yes
Artificial Physics	Yes	Yes	Yes	Yes (Mathematical)	Yes	Yes (limited)

experiment and then forming a team to carry out that experiment. The following is a brief description of the scenario:

Team leaders contain models of the types of science they want to perform. Parts of this model are communicated to the messenger spacecraft that then relay it on to the worker spacecraft. The worker spacecraft then take measurements of asteroids using whatever kind of instrument they have until something matches the goal that was sent down by the leader.

The data will then be sent to a messenger to be sent back to the leader. If the data matches the profile of the type of asteroid that is being searched for, an imaging spacecraft will be sent to the asteroid to ascertain the exact location and to create a rough model

Table 4 Properties and Constraints of Existing Methods

Method	Useful Properties and Difficulties
CSP	Ability to model check Case-based reasoning approach Not able to specify algorithms and data manipulation (without very artificial manipulations)
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 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 two 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 viewed 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

prior to the arrival of other spacecraft so they have a model to use for maneuvering around the asteroid.

Other spacecraft that would then work together to finish the model and mapping of the asteroid would include:

- an asteroid detector/stereo mapper team that would consist of two spacecraft with field imaging spectrometers and a dynamic modeler with an enhanced radio science instrument for measuring dynamic properties (such as spin, density and mass distribution)
- a petrologist team that would consist of X-ray, Near Infrared, Gamma-ray, Thermal IR and wide fieled imager to determine the distribution of elements, minerals and rocks present
- a photogeologist team that would consist of Narrow Field and Wide Field Imagers and Altimeter to determine the nature and distribution of geological units based on texture, albedo, color, and apparent stratigraphy
- a prospector team consisting of an altimeter, magnetometer, near infrared, infrared, and X-ray spectrometers to determine the distribution of resources

The above teams would work together to form a model of asteroids as well as form virtual instruments.

Many things can happen when an ANTS team encounters an asteroid (Figure 2). A spacecraft can perform a flyby and make opportunistic observations. The flyby can be used to first determine if the asteroid is of interest before sending an entire team to the asteroid, or to determine that, due to the nature of the instrument on the spacecraft, only a flyby is necessary. If the asteroid is of interest, a mapping spacecraft will map the asteroid and determine its size, rate and axis of rotation, whether the asteroids have any satellites/moons, etc. This information is passed on to other spacecraft that will be making observations and need to perform a flyby, enter an orbit around the asteroid, enter a hovering point, etc. As more data is obtained about the asteroid, other ANTS spacecraft maybe sent to the asteroid for further data gathering.

The following first gives the partial specifications of the ANTS mission using CSP, WSCCS, Unity Logic, and X-Machines. Due to space requirements only samples of the specifications are given.

5.1 CSP specification of ANTS

The following is a specification of the behavior of the NASA ANTS mission using Communicating Sequential Processes. 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:

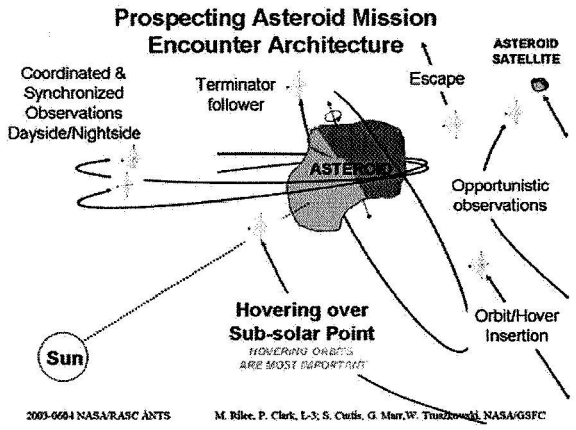

Fig. 2 ANTS encounter with an asteroid

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

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:

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

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 to store conversations, goals and models. Since at startup there have been no conversations, the conversation set ($conv$) 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.

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

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.

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

The following gives additional information on the leader information messages.

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

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.

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

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.

5.2 WSCCS Specification of ANTS

To model the ANTS Leader spacecraft, WSCCS, 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:

Table 5 Agent state and actions

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. Given the information from the table above, we define the agent-states as

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

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:

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

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.

$Processing \equiv$
$17\omega^2 : ProcessingSortingAndStorage.Processing$
$+17\omega^2 : ProcessingGeneration.Processing$
$+17\omega^2 : ProcessingPrediction.Processing$
$+16\omega^2 : ProcessingDiagnosis.Processing$
$+16\omega^2 : ProcessingRecovery.Processing$
$+17\omega^2 : ProcessingRemediation.Processing$
$+50\omega^3 : ReasoningDeliberative.Reasoning$
$+50\omega^3 : ReasoningReactive.Reasoning$

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 automatons represented by a transition graph. A transition graph derived from these transitions for the ANTS Leader Spacecraft is shown in Figure 3. (Nodes represent the agents and the edges between represent the weights and actions.)

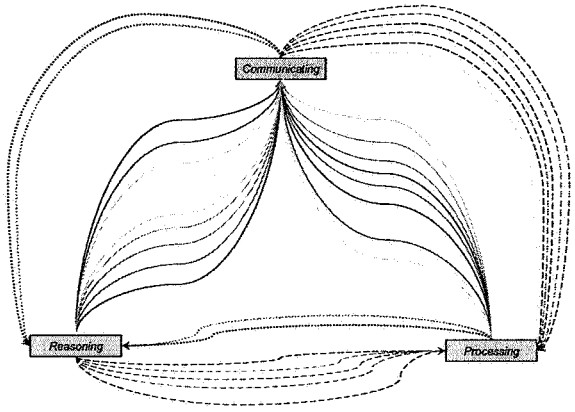

Fig. 3 Transition graph for WSCCS approach

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 $m_1\omega^{k_1} * m_2\omega^{k_2} * \cdots * m_n\omega^{k_n}$, 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 that 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

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

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.

Emergent Behavior of a Leader Using Markov Chains Figure 4 gives an overview of the states and the probability of going from one state to another using Markov Chains, which gives a different view of the Leader's emergent behavior.

Based on these statements and the previous frequencies and priorities, we can calculate the probabilities for the Leader choosing each action and therefore the prob-

abilities 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 percent.

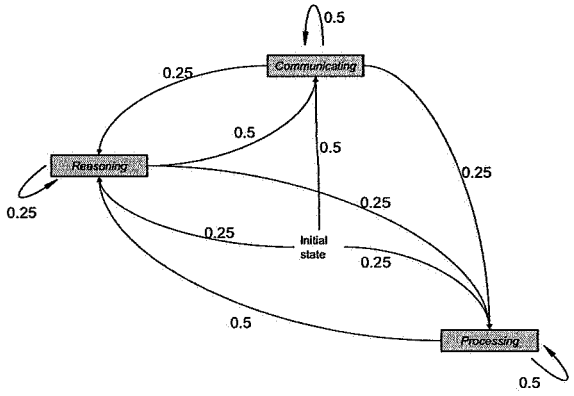

Fig. 4 Probabilities of going from one Leader state to another

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

Given this matrix, we can calculate the various powers, P^n, of the matrix. The nth power of the matrix P will tell us the probabilities which state the Leader will be in on the nth time step. For example, consider the results showing the calculated matrix P^2 in Figure 5.

We see in the matrix for P^2 that the entry P^2_{242} is 0.25. This tells us that if the Leader begins in the fourth state (Processing), it has a probability of 25 percent of being in the second state (Communicating) on the second time step. Observe in Figures 6 and 7 the convergence of these matrices at higher powers — i.e., as time goes on.

We see the powers of P converging to the matrix

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

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.

5.3 Unity Logic Specification of ANTS

To model the ANTS Leader spacecraft with Unity Logic, we consider states of the Leader just as in 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 is 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 in Table 6.

Table 6 Leader States and Transitions

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

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.

5.4 X-Machines Specification of ANTS

To model the ANTS Leader spacecraft as an X-Machine, 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:

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

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

$$P^2 = \begin{pmatrix} 0 & .375000000000000000 & .312500000000000000 & .312500000000000000 \\ 0 & .375000000000000000 & .312500000000000000 & .312500000000000000 \\ 0 & .375000000000000000 & .312500000000000000 & .312500000000000000 \\ 0 & .250000000000000000 & .375000000000000000 & .375000000000000000 \end{pmatrix}$$

Fig. 5 Calculated matrix P^2

$$P^8 = \begin{pmatrix} 0 & .333343505859375000 & .333328247070312500 & .312500000000000000 \\ 0 & .333343505859375000 & .333328247070312500 & .333328247070312500 \\ 0 & .333343505859375000 & .333328247070312500 & .333328247070312500 \\ 0 & .333312988281250000 & .333343505859375000 & .333343505859375000 \end{pmatrix}$$

Fig. 6 Calculated matrix P^8

$$P^{10000000000} = \begin{pmatrix} 0 & .333333333333333370 & .333333333333333370 & .333333333333333370 \\ 0 & .333333333333333370 & .333333333333333370 & .333333333333333370 \\ 0 & .333333333333333370 & .333333333333333370 & .333333333333333370 \\ 0 & .333333333333333370 & .333333333333333370 & .333333333333333370 \end{pmatrix}$$

Fig. 7 Calculated matrix $P^{10000000000}$

$m' = (Goals', Model')$

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

is another set of data.

$Q = \{Start, Communicating, Reasoning,$
$Processing\}$

is a set of states.

$\Phi = \{SendMessage, ReceiveMessage, Reason,$
$Process\}$

is a set of (partial) transition functions where each transition function maps $Memory \times Input \rightarrow Output \times Memory$ as in the following:

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

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 in Figure 8. Nodes represent the states and the edges between represent the transition functions.

X-Machines better reflect executable systems than standard finite state machines and would allow for a more natural specification of the ANTS spacecraft. It allows for a memory to be kept and transitions between states to be seen as functions involving inputs and outputs. This would allow specifications to track the actions of the ANTS spacecraft as well as write to memory any aspect of the goals and model of the mission as they change. This ability makes X-Machines highly effective for tracking and effecting changes in the goals and model. However, X-Machines do not provide any robust means for reasoning about or predicting behaviors of one or more spacecraft, beyond standard propositional logic. This would make specifying emergent behavior difficult. The table in the following section summarizes these properties.

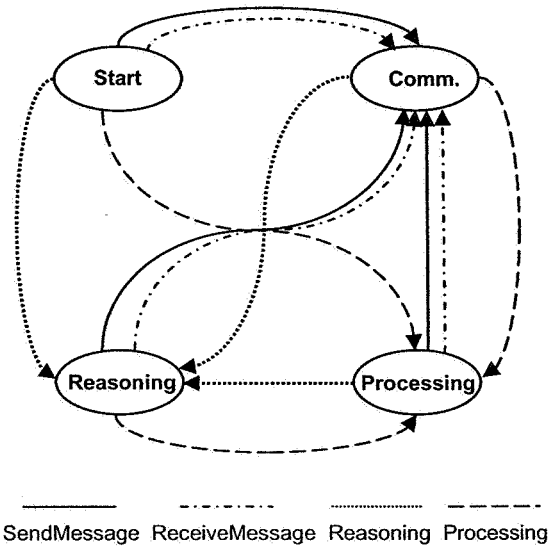

Fig. 8 Transition diagram of Leader for X-Machines

5.5 An Appraisal of Approaches

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 undertaken. Therefore, the formal method must be flexible enough to allow for efficient changes and re-prediction of emergent behavior.

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

Processes (X-Machines, CSP) - Processes can be specified using the various manifestations of transition functions.

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.

Choosing action alternatives (WSCCS) - A modified version of this ability from WSCCS may be used to supply an algebra for choosing between possible actions.

Asynchronous messaging (CSP) - Messaging may not be synchronized upon or after implementation. There are variants of CSP that support asynchronous messaging.

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.

Concurrent states for each spacecraft (WSCCS) - This ability is solidly in place and will require only an augmentation of notation.

Communication protocols between agents (CSP) – CSP allows for this as it stands.

Conversion to code (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.

Determining whether 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.

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.

Model checking (CSP) - Model checking will prevent semantic inconsistencies in the specifications.

Tracking 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.

Associating actions with priorities (WSCCS) - This ability is firmly in place.

Associating actions with frequencies (WSCCS) - This ability is firmly in place.

Predicting emergent behavior (WSCCS) - Current WSCCS abilities are not robust enough for the purpose of predicting individual and swarm emergent behavior and will need to be enhanced by greater use of Probability, Markov Chains, and/or Chaos Theory.

6 Evaluation of Methods

The following describes the results of the sample specifications and an evaluation of the methods used.

6.1 CSP

CSP 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, particularly in swarm-based systems which are inherently highly parallel. From a CSP specification, reasoning about the specification can be undertaken to determine race conditions, and the specification can be easily converted into a model checking language for running through a model checker. Although these are important issues and process algebras

have been widely and successfully used for specifying agent-based systems, there is no facility for evaluating emergent behavior of the end system.

6.2 WSCCS

WSCCS provides a process algebra that takes into account the priorities and probabilities of actions performed. Further, it provides a syntax and a large set of rules for predicting and specifying choices and behaviors, as well as a congruence and syntax for determining whether 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 when combined in a mission. 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 persistent memory.

6.3 Unity Logic

Unity Logic provides a logical syntax equivalent to simple Propositional Logic for reasoning about predicates and the states they define, 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, although the logic does provide many useful properties for reasoning about systems. CommUnity [15], an extension of Unity, may address these limitations, but has not yet been examined by the FAST project.

6.4 X-Machines

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 do not provide any robust means for reasoning about or predicting behaviors of one or more spacecraft, beyond standard propositional logic. This will make specifying or analyzing emergent behavior difficult or impossible.

6.5 Results of Comparison

An integration of the above methods seems to be the best approach for specifying swarm-based systems and analyzing emergent behavior of these systems. Figure 9 gives a high level overview of the resulting formal method. Blending the memory and transition function aspects of X-Machines with the priority and probability aspects of WSCCS may produce a specification method that will allow us to address all of the necessary aspects for specifying emergent behavior in the ANTS mission, and other swarm-based systems.

7 Future Work

Currently the project is working on integrating the above formal methods into a single formal method. After integrating the formal methods, an in-depth specification of the ANTS concept mission, and possibly a second NASA swarm mission, will be developed to give examples of the use of the new formal method as well as to determine if there are any modifications that need to be made to the new method. In addition to the integration of the above formal methods, tools for supporting the formal method are being developed. Potential tools are being considered that can be modified or developed as well as translators from the new formal method to these tools. Examples of some of the tools that are being examined include editors, syntax checkers, theorem provers and model checkers.

8 Conclusion

Swarms are being proposed and investigated for a number of applications. NASA is studying swarm-based systems to use on future missions to conduct new science and support unmanned exploration that is currently not possible. 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. Because of this, these missions must have an even higher level of verification performed on them than has been done on past missions that have been in constant contact with mission control.

To verify NASA swarm-based missions 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 autonomic properties and the ability to modify operations autonomously to reflect the changing nature of the mission. For this, a 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

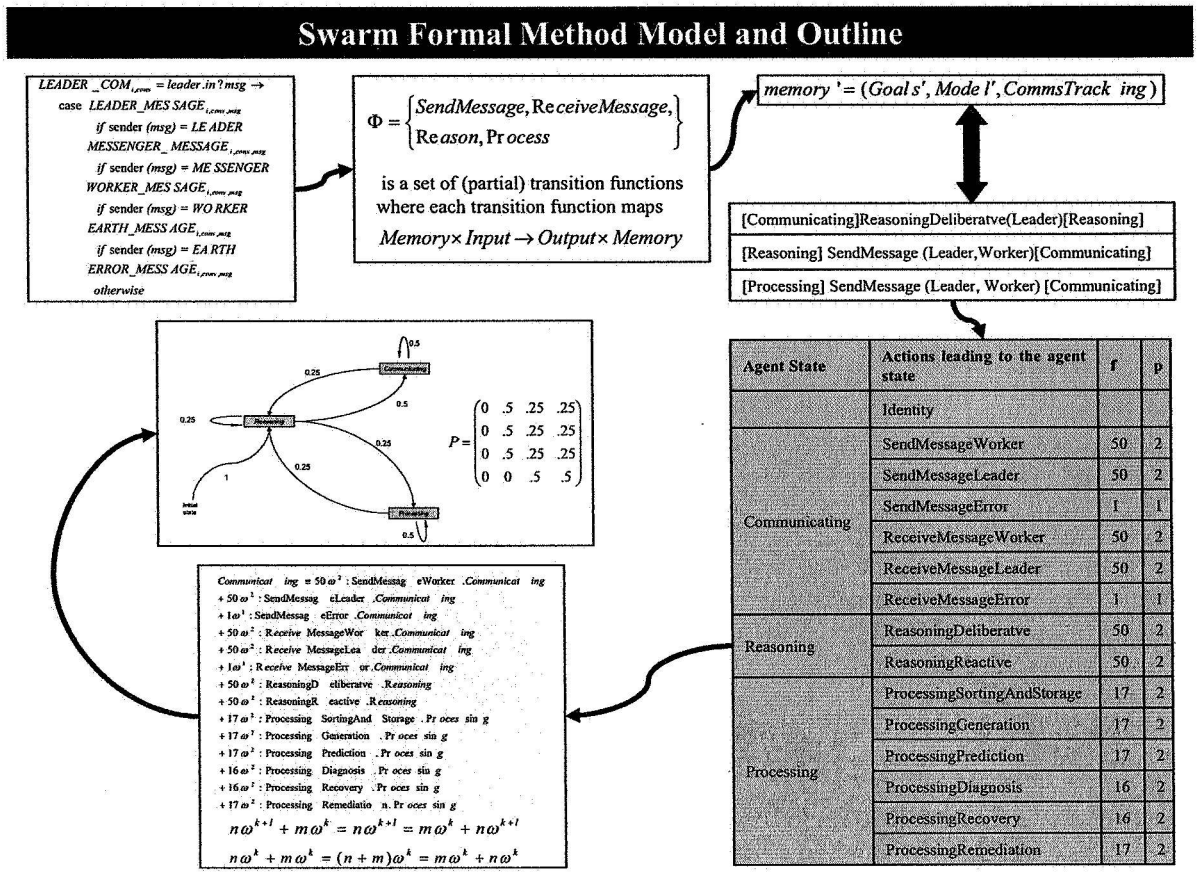

Fig. 9 Integrated formal method

in. The formal specification will also need to allow for specification of the decision-making process to aid in the decision as to which instruments will be needed, at what location, with what goals, etc.

We have identified several important attributes needed in a formal approach for verifying swarm-based systems. We have also surveyed a wide number of formal methods and approaches for verification of swarm-based systems and have identified several formal methods that have been used for modeling swarms or have the appropriate attributes. From this potential list we have done 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 alone are not sufficient. We are currently integrating these methods to develop a new formal method for swarm-based systems and will test this new formal method by developing 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. This paper is substantially based on [32].

References

1. Beni, G.: The concept of cellular robotics. In: Proc. 1988 IEEE International Symposium on Intelligent Control, pp. 57–62. IEEE Computer Society Press, Los Alamitos, Calif. (1988)
2. Beni, G., Want, J.: Swarm intelligence. In: Proc. Seventh Annual Meeting of the Robotics Society of Japan, pp. 425–428. RSJ Press, Tokyo, Japan (1989)
3. Blackburn, M., Busser, R., Nauman, A., Knickerbocker, R., Kasuda, R.: Mars polar lander fault identification using model-based testing. In: Proceedings of the 26th Annual IEEE/NASA Software Engineering Workshop (SEW). Greenbelt, MD (2001)
4. Bonabeau, E., Dorigo, M., Théraulaz, G.: Swarm Intelligence: From Natural to Artificial Systems. Oxford University Press, New York (1999)
5. Bonabeau, E., Théraulaz, G.: Swarm smarts. Scientific American pp. 72–79 (2000)

6. Bonabeau, E., Théraulaz, G., Deneubourg, J.L., Aron, S., Camazine, S.: Self-organization in social insects. Trends in Ecology and Evolution **12**, 188–193 (1997)
7. Brueckner, S., Parunak, H.V.D.: Resource-aware exploration of the emergent dynamics of simulated systems. In: Proceedings of Autonomous Agents and Multi Agent Systems (AAMAS), pp. 781–788 (2003)
8. Butler, M.J.: csp2B : A Practical Approach To Combining CSP and B. Declarative Systems and Software Engineering Group, Department of Electronics and Computer Science, University of Southampton (1999)
9. Carlson, S.: Artificial life: Boids of a feather flock together. Scientific American (2000)
10. Casani, J., Whetsler, C., Albee, A., Battel, S., Brace, R., Burdick, G., Burr, P., Dippoey, D., Lavell, J., Leising, C., MacPherson, D., Menard, W., Rose, R., Sackheim, R., Schallenmuller, A.: Report on the loss of the mars polar lander and deep space 2 missions. Tech. Rep. JPL D-18709, Jet Propulsion Laboratory, California Institute of Technology (2000)
11. Chandy, K.M., Misra, J.: Parallel Program Design: A Foundation. Addison-Wesley Publishing Company (1988)
12. Clark, P.E., Curtis, S.A., Rilee, M.L.: ANTS: Applying a new paradigm to Lunar and planetary exploration. In: Proc. Solar System Remote Sensing Symposium. Pittsburgh, Pennsylvania, USA (2002)
13. Curtis, S.A., Mica, J., Nuth, J., Marr, G., Rilee, M.L., Bhat, M.K.: ANTS (Autonomous Nano-Technology Swarm): An artificial intelligence approach to Asteroid Belt resource exploration. In: Proc. Int'l Astronautical Federation, 51st Congress (2000)
14. Curtis, S.A., Truszkowski, W.F., Rilee, M.L., Clark, P.E.: ANTS for the human exploration and development of space. In: Proc. IEEE Aerospace Conference. Big Sky, Montana, USA (2003)
15. Fiadeiro, J.L.: Categories for Software Engineering. Springer-Verlag, London (2004)
16. Hiebeler, D.E.: The swarm simulation system and individual-based modeling. In: Proc. Decision Support 2001: Advanced Technology for Natural Resource Management. Toronto, Canada (1994)
17. Hinchey, M., Rash, J., Rouff, C.: Verification and validation of autonomous systems. In: Proc. SEW-26, 26th Annual NASA/IEEE Software Engineering Workshop, pp. 136–144. NASA Goddard Space Flight Center, IEEE Computer Society Press, Los Alamitos, Calif., Greenbelt, MD (2001)
18. Hinchey, M.G., Bowen, J.P. (eds.): Industrial-Strength Formal Methods in Practice. FACIT Series. Springer-Verlag, London, UK (1999). URL http://www.springer.co.uk
19. Hinchey, M.G., Jarvis, S.A.: Concurrent Systems: Formal Development in CSP. International Series in Software Engineering. McGraw-Hill International, London, UK (1995)
20. Hinchey, M.G., Rash, J.L., Rouff, C.A.: Towards an automated development methodology for dependable systems with application to sensor networks. In: Proc. IEEE Workshop on Information Assurance in Wireless Sensor Networks (WSNIA 2005), Proc. International Performance Computing and Communications Conference (IPCCC-05). IEEE Computer Society Press, Los Alamitos, Calif., Phoenix, Arizona (2005)
21. Hoare, C.A.R.: Communicating Sequential Processes. Prentice Hall International Series in Computer Science. Prentice Hall International, Englewood Cliffs, NJ (1985)
22. Holcombe, W.M.L.: Mathematical models of cell biochemistry. Tech. Rep. CS-86-4, Sheffield University, UK (1986)
23. Holcombe, W.M.L.: Towards a formal description of intracellular biochemical organization. Tech. Rep. CS-86-1, Sheffield University, UK (1986)
24. Holcombe, W.M.L.: X-Machines as a basis for system specification. Software Engineering **3**(2), 69–76 (1988)
25. Horn, P.: Autonomic computing: IBM's perspective on the state of information technology. Tech. rep., IBM T. J. Watson Laboratory (October 15, 2001). URL http://www.research.ibm.com/autonomic/
26. Kiniry, J.R.: The specification of dynamic distributed component systems. Master's thesis, California Institute of Technology (1998)
27. Luna, F., Stefansson, B.: Economic Simulations in Swarm: Agent-Based Modelling and Object Oriented Programming. Kluwer Academic Publishers (2000)
28. Nayak, P.P., Bernard, D.E., Dorais, G., Jr., E.B.G., Kanefsky, B., Kurien, J., Millar, W., Muscettola, N., Rajan, K., Rouquette, N., Smith, B.D., Taylor, W., wen Tung, Y.: Validating the DS1 remote agent experiment. In: Proc. of the 5th International Symposium on Artificial Intelligence, Robotics and Automation in Space (iSAIRAS-99) (1999)
29. Parunak, H.V.D., Vanderbok, R.: Managing emergent behaviour in distributed control systems. In: Proceedings of ISA-Tech'97. Anaheim, CA (1997)
30. Reynolds, C.W.: Flocks, herds, and schools: A distributed behavioral model. Computer Graphics **21**(4), 25–34 (1987)
31. Rouff, C., Rash, J., Hinchey, M., Truszkwoski, W.: Formal methods at NASA Goddard Space Flight Center. In: Agent Technology from a Formal Perspective, NASA Monographs in Systems and Software Engineering, pp. 287–310. Springer-Verlag, London, UK (2005)
32. Rouff, C., Vanderbilt, A., Hinchey, M., Truszkowski, W., Rash, J.: Formal methods for swarm and autonomic systems. In: Proc. 1st International Symposium on Leveraging Applications of Formal Methods (ISoLA). Cyprus (2004)
33. Rouff, C.A., Rash, J.L., Hinchey, M.G.: Experience using formal methods for specifying a multi-agent system. In: Proc. Sixth IEEE International Conference on Engineering of Complex Computer Systems (ICECCS 2000). IEEE Computer Society Press, Los Alamitos, Calif., Tokyo, Japan (2000)
34. Rouff, C.A., Truszkowski, W.F., Hinchey, M.G., Rash, J.L.: Verification of emergent behaviors in swarm based systems. In: Proc. 11th IEEE International Conference on Engineering Computer-Based Systems (ECBS), Workshop on Engineering Autonomic Systems (EASe), pp. 443–448. IEEE Computer Society Press, Los Alamitos, Calif., Brno, Czech Republic (2004)
35. Rouff, C.A., Truszkowski, W.F., Hinchey, M.G., Rash, J.L.: Verification of NASA emergent systems. In: Proc. 9th IEEE International Conference on Engineering of Complex Computer Systems. IEEE Computer Society Press, Los Alamitos, Calif., Florence, Italy (2004)
36. Rouff, C.A., Truszkowski, W.F., Rash, J.L., Hinchey, M.G.: A survey of formal methods for intelligent swarms. Tech. Rep. TM-2005-212779, NASA Goddard Space Flight Center, Greenbelt, Maryland (2005)
37. Savage, M., Askenazi, M.: Arborscapes: A swarm-based multi-agent ecological disturbance model. Working paper 98-06-056, Santa Fe Institute, Santa Fe, New Mexico (1998)
38. Shehory, O., Kraus, S., Yadgar, O.: Emergent cooperative goal-satisfaction in large-scale automated-agent systems. Artif. Intell. **110**(1), 1–55 (1999)
39. Spears, W.M., Gordon, D.F.: Using artificial physics to control agents. In: Proc. IEEE International Conference on Information, Intelligence, and Systems. Charlotte, North Carolina (1999)

40. Sterrit, R., Hinchey, M.G.: SPAACE :: Self- properties for an autonomous & autonomic computing environment. In: Proc. The 2005 International Conference on Software Engineering Research and Practice (SERP'05), pp. 9–15. CSREA Press, Las Vegas, Nevada, USA (2005)
41. Sterritt, R., Hinchey, M.G.: Apoptosis and self-destruct: A contribution to autonomic agents? In: Proc. FAABS-III, 3rd NASA/IEEE Workshop on Formal Approaches to Agent-Based Systems, pp. 269–278. Springer-Verlag (2004)
42. Sterritt, R., Rouff, C.A., Rash, J.L., Truszkowski, W.F., Hinchey, M.G.: Self-* properties in NASA missions. In: 4th International Workshop on System/Software Architectures (IWSSA'05) in Proc. 2005 International Conference on Software Engineering Research and Practice (SERP'05), pp. 66–72. CSREA Press, Las Vegas, Nevada, USA (2005)
43. Sumpter, D.J.T., Blanchard, G.B., Broomhead, D.S.: Ants and agents: a process algebra approach to modelling ant colony behaviour. Bulletin of Mathematical Biology **63**(5), 951–980 (2001)
44. Tofts, C.: Describing social insect behavior using process algebra. Transactions on Social Computing Simulation pp. 227–283 (1991)
45. Truszkowski, W., Hinchey, M., Rash, J., Rouff, C.: NASA's swarm missions: The challenge of building autonomous software. IEEE IT Professional **6**(5), 47–52 (2004)
46. Truszkowski, W.F., Hinchey, M.G., Rash, J.L., Rouff, C.A.: Autonomous and autonomic systems: A paradigm for future space exploration missions. IEEE Transactions on Systems, Man and Cybernetics, Part C (2006 (to appear))
47. von Neumann, J.: Theory of Self-Reproducing Automata. University of Illinois Press, Urbana, Illinois (1996)
48. Weiss, G. (ed.): Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence. MIT Press, Cambridge, Massachusetts (1999)