Formal Verification of Astronaut-Rover Teams for Planetary Surface Operations

Matt Webster
Department of Computer Science
University of Liverpool
Ashton Street, Liverpool, L69 3BX, UK
matt@liverpool.ac.uk

Louise A. Dennis
Department of Computer Science
University of Liverpool
Ashton Street, Liverpool, L69 3BX, UK
L.A.Dennis@liverpool.ac.uk

Clare Dixon
Department of Computer Science
University of Liverpool
Ashton Street, Liverpool, L69 3BX, UK
cldixon@liverpool.ac.uk

Michael Fisher
Department of Computer Science
University of Liverpool
Ashton Street, Liverpool, L69 3BX, UK
mfisher@liverpool.ac.uk

Richard Stocker
Department of Computer Science
University of Chester
Pool Lane, Chester, CH2 4NU, UK
r.stocker@chester.ac.uk

Maarten Sierhuis
Ejenta, Inc.
181 2nd Street,
San Francisco, CA 94105, USA
sierhuis@ejenta.com

Abstract— **This paper describes an approach to assuring the reliability of autonomous systems for Astronaut-Rover (ASRO) teams using the formal verification of models in the Brahms multi-agent modelling language. Planetary surface rovers have proven essential to several manned and unmanned missions to the moon and Mars. The first rovers were tele- or manually-operated, but autonomous systems are increasingly being used to increase the effectiveness and range of rover operations on missions such as the NASA Mars Science Laboratory. It is anticipated that future manned missions to the moon and Mars will use autonomous rovers to assist astronauts during extra-vehicular activity (EVA), including science, technical and construction operations. These ASRO teams have the potential to significantly increase the safety and efficiency of surface operations. We describe a new Brahms model in which an autonomous rover may perform several different activities including assisting an astronaut during EVA. These activities compete for the autonomous rover's "attention" and therefore the rover must decide which activity is currently the most important and engage in that activity. The Brahms model also includes an astronaut agent, which models an astronaut's predicted behaviour during an EVA. The rover must also respond to the astronauts activities. We show how this Brahms model can be simulated using the Brahms integrated development environment. The model can then also be formally verified with respect to system requirements using the SPIN model checker, through automatic translation from Brahms to PROMELA (the input language for SPIN). We show that such formal verification can be used to determine that mission- and safety-critical operations are conducted correctly, and therefore increase the reliability of autonomous systems for planetary rovers in ASRO teams.**

TABLE OF CONTENTS

978-1-7281-2734-7/20/$31.00 ©2020 IEEE

1. INTRODUCTION

Planetary surface rovers have proven essential to several manned and unmanned missions to the moon and Mars. The first rovers were tele- or manually-operated, but autonomous systems are increasingly being used to increase the effectiveness and range of rover operations on missions such as the NASA Mars Science Laboratory [1]. It is anticipated that future manned missions to the moon and Mars will use autonomous rovers to assist astronauts during extra-vehicular activity (EVA), including science, technical and construction operations. These ASRO teams have the potential to significantly increase the safety and efficiency of surface operations [2], [3], [4].

In this paper we examine an ASRO team scenario in which an autonomous rover performs (i) teamwork operations to directly and indirectly assist an astronaut in a variety of activities, and (ii) solo operations when the astronaut is otherwise engaged, e.g., during rest periods. Our analysis is based on a formal model of the scenario written in Brahms [5], [6], a general-purpose multiagent workflow language developed at NASA Ames Research Center, and used previously for ASRO team modelling as part of the NASA Mobile Agents Architecture [7]. We describe a new Brahms model in which an autonomous rover may perform several different activities including assisting an astronaut in geological surveys, carrying tools and materials for an astronaut during habitat maintenance and construction, solo geological surveying, habitat integrity monitoring, and recording video to document astronaut EVA. These activities compete for the autonomous rovers "attention" and therefore the rover must decide which activity is currently the most important and engage in that activity. The Brahms model also includes an astronaut agent, which models an astronaut's predicted behaviour during an EVA. The rover must also respond to the astronaut's activities.

We show how this Brahms model can be simulated using the Brahms integrated development environment. The model can then also be formally verified with respect to system requirements using the SPIN model checker [8], through automatic translation from Brahms to PROMELA (the input language for SPIN). The automatic translation is based on

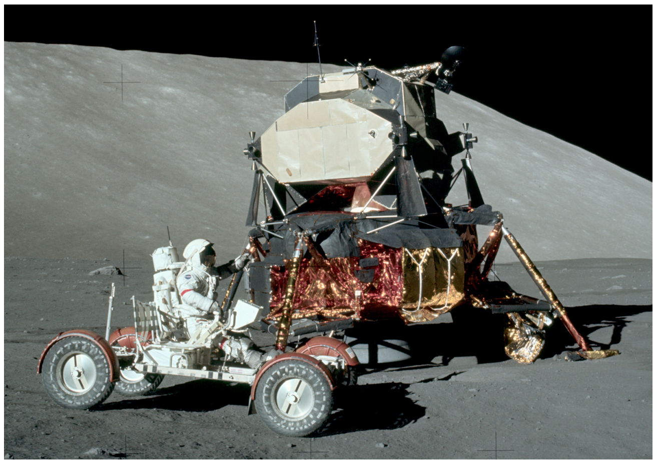


Figure 1. **Astronaut, rover and habitat on Apollo 17 [10].**

a formalisation of the semantics of the Brahms modelling language [16], and system requirements are formalised using linear temporal logic [9]. We show that such formal verification can be used to determine that mission- and safety-critical operations are conducted correctly, and therefore increase the reliability of autonomous systems for planetary rovers in ASRO teams. The Brahms control software developed in this way, and assessed at this higher level of integrity, can then be translated to resilient, effective, and correct software agents that can directly control the rover. These high-level agent control systems can additionally be formally verified in the practical robotic context, providing further assurance and confidence.

This paper contributes the following:

- A methodology to assure the reliability and correctness of autonomous rover behaviour within astronaut–rover teams.
- A high-level Brahms model of an astronaut–rover team scenario in which the rover autonomously assists the astronaut in a range of tasks.
- Formal verification of key requirements of the autonomous rover within the scenario, encoded as logical properties that are used during model checking.
- The potential to extend this model to control simulated or real-life rovers using a Brahms–Java interface.

The structure of the paper is as follows. In Section 2 we describe the astronaut–rover scenario and show how it was modelled using Brahms. Section 3 describes the formal verification technique used to assess the reliability and correctness of the rover's behaviour within the ASRO team. Finally, in Section 4 we provide conclusions and discuss directions for future work. We end this section with an overview of related work.

Related Work

Astronaut–rover teams were first used in practice on the Apollo 15, 16 and 17 missions (see Figure 1). The lunar roving vehicle used minimal autonomy: an autonomous guidance system would continually monitor and report the direction and distance to the lunar module "habitat". More recently, ASRO teams have been simulated and analysed for use in future space exploration missions. Ransan & Atkins [11] describe a multiagent planning system for distributing tasks to astronaut-rover teams. Medina et al. [12] describe the development of a prototype autonomous rover for planetary outpost assembly. Heiskanen et al. [13] developed a dynamic mobile robot simulator for astronaut–rover teams using a centaur-like robot. Fong & Nourbakhsh [14] describe an approach to developing human–robot teams for space exploration. However, none of these papers use formal verification to determine the reliability of ASRO teams, as we do here.

This paper is based on our previous work on Brahms for astronaut–rover teams and multiagent modelling and verification. Sierhuis et al. [7] used Brahms as the basis of the Mobile Agents Architecture at NASA Ames Research Center, which enabled the autonomous operation of rovers, spacesuits and habitats in a simulated Mars environment. Stocker et al. [15] used Brahms to model and formally verify human–robot teamwork for domestic robotic assistants based on a formal semantics of the Brahms language [16]. Webster et al. [17] used Brahms to model and formally verify the behaviour of a domestic robotic assistant deployed at the University of Hertfordshire. Bordini et al. [18] described an approach to formal verification of Brahms models of human–robot teams through translation into the Jason agent programming language. However, this paper is the first use of formal verification to analyse the behaviour of Brahms multiagent systems for ASRO teamwork.

2. MODELLING ASTRONAUT–ROVER TEAMS

Brahms is a general-purpose multiagent workflow language. Multiagent systems allow autonomous systems to be specified in terms of multiple interacting autonomous *agents*. Brahms models generally consist of a set of agents. Each agent consists primarily of workframes and thoughtframes, in which the agent's behaviours are defined. Agents also possess *beliefs* which represent their current view of the world. The actual state of the world may also be encoded using *facts*. Facts and beliefs in Brahms resemble typical programming language variables such as Booleans and integers. Workframes are triggered by a Boolean guard that, when true, will cause a sequence of deeds to be performed. These deeds can include belief modifications, communication, movements and primitive activities. Communication allows beliefs to be sent between agents. Agents can also possess beliefs about other agents' beliefs. Movements take place with reference to an optional geography defined in the Brahms model. Both movements and primitive activities can take a variable amount of time set within minimum and maximum parameters. Thoughtframes are special cases of workframes in which there are no activities or movement, but during which deductions can take place based on information received by an agent. Brahms is supported by the Brahms Composer integrated development environment (IDE) which allows agents to be compiled, simulated, debugged and analysed (see Figure 2).

A Brahms model of an astronaut–rover team was developed based on a scenario in which an astronaut and a rover work together to conduct science and construction activities at a newly-established outpost on a planetary surface such as the moon or Mars. The outpost consists of a habitat which allows the astronaut to rest and work without the use of a spacesuit, and an EVA area in which the astronaut and rover work together. Astronaut–Rover (ASRO) team operations within the scenario are based on the first field experiments of simulated ASRO teams conducted by the NASA Ames and Johnson Research Centers in the Mojave Desert in 1999 [2]. Specifically, the rover is used in the following roles, adapted

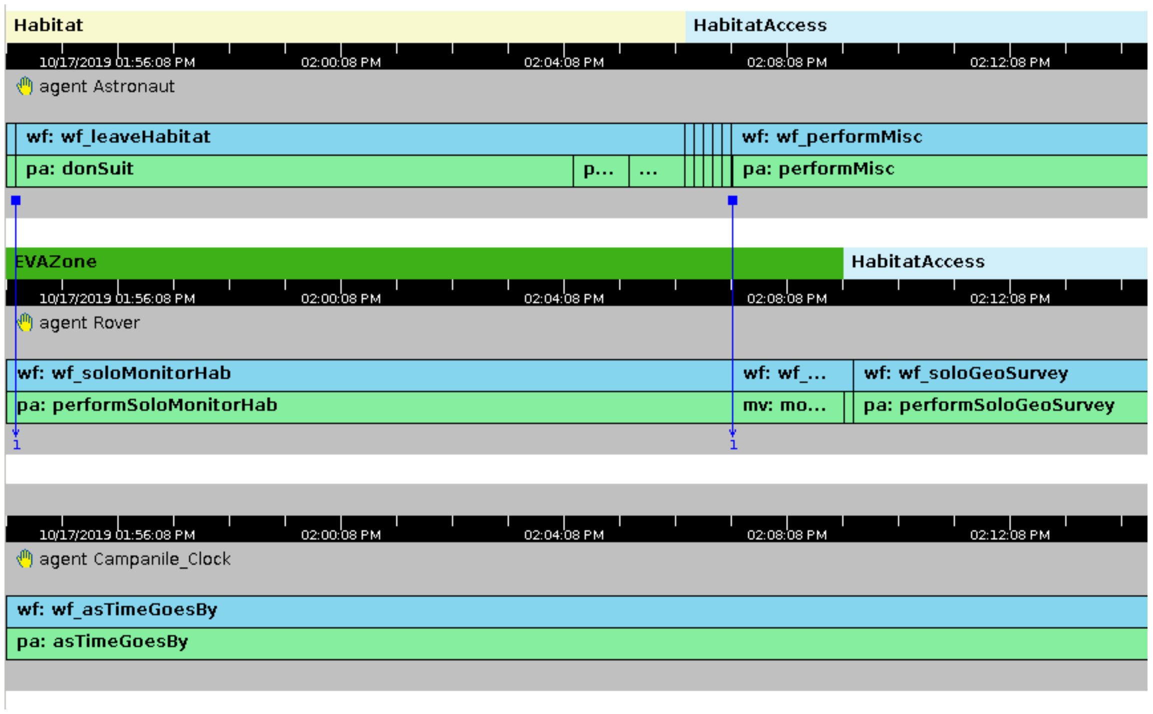


Figure 2. Simulation of the Brahms astronaut–rover model using the Brahms Composer IDE.

for use in our scenario:

1. *Rover as a scout*: the rover examines the traverse area to find potential favourable sites for geological surveying.
2. *Rover as a video coverage assistant*: the rover is used to document the astronaut's activity during extra-vehicular activity (EVA), i.e., activity outside the habitat.
3. *Rover as a field science assistant*: the rover assists the astronaut during a geological survey, by collecting and documenting samples identified by the astronaut.
4. *Rover as a field technical assistant*: the rover is used to carry tools and materials for construction activity undertaken by the astronaut.

In addition, our scenario includes the following roles:

5. *Rover as a monitor*: the rover monitors the astronaut during potentially hazardous or interesting activity. In our scenario, this corresponds to the times within the simulation when the astronaut leaves (egresses) or enters (ingresses) the habitat from the external environment, which involves pressurising or depressurising the habitat airlock.
6. *Rover as a guard*: the rover performs structural integrity checks on the habitat during the periods of time in which the astronaut is inside the habitat.

The astronaut moves between the habitat and the EVA area on a typical work day. The rover detects the astronaut's activities and responds autonomously to assist the astronaut. During times that the astronaut does not need or want to be directly assisted, the rover can conduct solo operations for geological surveying or habitat monitoring. At the start of the work day the astronaut leaves the habitat and conducts a number of surface operations, before retiring to the habitat at the end of the work day. The astronaut may also return to the habitat during the work day, if needed. The astronaut may conduct any of the following behaviours:

1. Maintenance and construction of structures, e.g., for extending the habitat or expanded science experiments.
2. Video EVA, in which the astronaut requests the rover to video them while they perform an activity, e.g., to document the completion of a key mission objective.
3. Geological surveying, in which the astronaut collects rock samples and is assisted by the rover.
4. Returning to the habitat from the work or survey areas.
5. Leaving and entering the habitat.
6. Miscellaneous behaviour. The astronaut may wish to conduct activities that do not directly involve the rover. In this case the astronaut can "dismiss" the rover to perform solo activities for a period of time.

The Brahms model consists of three agents: the Astronaut, the Rover, and the Campanile Clock. The latter agent is used to help maintain a consistent definition and flow of time within the model by sending regular updates to agent regarding the current time. The astronaut agent consists of workframes corresponding to the behaviours described above. For example, the following workframe, `wf_considerPerformConstruction` specifies how the astronaut agent chooses to perform a construction activity:

```
workframe wf_considerPerformConstruction {
 repeat: true;
 priority: 99;

 when(knownval(Campanile_Clock.time < 9)
       and
      knownval(current.location != Habitat)
       and
     knownval(current.
         consideredPerformConstruction = false)
)
 do {
  conclude((current.consideredPerformConstruction
   = true));
  conclude((current.goalPerformConstruction = true
      ),bc:25,fc:0);
 }
}
```

The first line declares the name of the workframe. The workframe is set to repeat (`repeat: true;`), which means that the workframe can be selected by the agent more than once. The next line sets the priority. Workframes can be prioritised, with higher priority workframes being selected first. The `when (G) do Ds` construct states that when the guard `G` is true the deeds `Ds` should be executed. In this case, the workframe will execute at any point in the first 8 hours of the work day, when the astronaut is outside the habitat, and the astronaut has not previously considered doing construction. When the workframe executes, the agent updates its beliefs to represent that it has now considered performing construction, and that it has a goal to perform construction. The latter belief is annotated with belief-certainty (`bc`) and fact-certainty (`fc`), which will be described below.

After the astronaut agent has considered doing construction, and has chosen to do it, the next step is to actually perform the construction activity. This is done in the following workframe:

```
workframe wf_performConstruction {
 repeat: true;
 priority: 99;

 when(knownval(current.goalPerformConstruction =
      true))
 do {
  announcePerformConstruction();
  moveToWorkSite();
  performConstruction();
  conclude((current.goalPerformConstruction =
       false));
 }
}
```

This workframe states that when the agent believes that it has a goal to perform construction, it will communicate with the rover agent to announce that it has decided to perform construction. Next, a movement is executed to update the location of the astronaut to the work site. The following deed is the primitive activity for "performing construction," which is modelled as taking 60 minutes. After construction is complete, the agent concludes that it no longer has a goal to perform construction, as construction is now complete. When the campanile clock agent updates the time for the astronaut agent, the astronaut agent resets its beliefs above having considered performing construction, which allows the `wf_considerPerformConstruction` workframe to be selected again, potentially allowing the astronaut agent to continue performing construction.

Earlier we described how the belief to have a goal to perform construction is updated with a belief-certainty. In this case, the belief-certainty is set to 25%, which means that there is a 25% probability that the belief is set to true, and a 75% probability that the belief will remain false. This means that there is a 25% probability that the astronaut agent will actually perform construction. This is to prevent the astronaut agent from doing construction constantly. Each of the astronaut agent's behaviours can be selected with a 25% probability, allowing behaviours to be chosen non-deterministically. Without the use of belief-certainties, the astronaut agent's behaviours would be completely deterministic and every run of the simulation would result in an identical sequence of behaviours being selected. In order to model our scenario more accurately it is important that the astronaut agent behaves non-deterministically so we can analyse a range of possibilities during verification. Note that the probability of 25% was chosen to allow an interesting range of behaviours to be viewed during simulation. From a formal verification perspective, the exact value of this probability is not relevant. As long as it is not set to 0% or 100%, and is set to some other value, the astronaut agent will choose its behaviours non-deterministically.

As described above, the astronaut agent communicates with the rover agent by sending it a message saying that it intends to begin construction. In reality, the astronaut would not necessarily communicate this directly to the rover. For example, a monitoring system could determine from the astronaut's location and behaviour that they are engaged in construction, and notify the rover on behalf of the astronaut. Either way, this can be modelled as a message from the astronaut agent to the rover agent in our model.

When the rover agent receives this message, a workframe, `wf_considerAssistConstruction`, is triggered. This is similar in content to the `wf_considerPerformConstruction` workframe, and states that once the rover agent believes that the astronaut agent believes that it has a goal to perform construction, that the rover agent will itself form a goal about assisting construction. This is then handled by another workframe, `wf_assistConstruction`, which is similar to `wf_performConstruction` in the astronaut agent, with one significant difference. The rover's workframes contain *detectables* which allow the rover to interrupt its behaviour if the astronaut changes their mind and chooses to do something else. These detectables have the following form:

```
detectable detectGeoSurvey {
 when(whenever)
 detect((Astronaut.goalPerformGeoSurvey = true),
      dc:100)
 then abort;
}
```

This detectable states that whenever the rover agent detects that the astronaut agent has decided to perform a geological survey, then the rover agent will abort its current activity, allowing it to respond to the astronaut's change in behaviour. The rover agent's workframes contain up to five detectables that allow the rover agent to detect various astronaut behaviours and adapt to them.

3. Formal Verification

Formal verification describes an approach to verification of computer systems, processes, and protocols using formal methods. Formal methods involve mathematical models of systems which can be analysed using proof-based methods. The use of proofs gives a higher level of confidence in the results of the verification than running tests or simulations. However, there is a trade-off in that formal models can be time-consuming to construct and automatic proofs often require large amounts of time and memory to execute.

The approach to formal verification used in this paper is called model checking. Model checking involves checking the full state space of a non-deterministic model, and is therefore described as exhaustive. Model checkers have a long history of successful use in verification of high-integrity systems dating back to the early 1980s [19]. We used the SPIN model checker to verify the astronaut–rover team described in the previous section. SPIN verifies models constructed using the process meta-language PROMELA. PROMELA lets us model systems of concurrently-executing and communicating automata. It would be possible to construct a PROMELA model of the Brahms model from the previous section by hand. However this would be laborious and error-prone. Instead we use a software translator called *BrahmsToPromela* [15], which lets us automatically translate Brahms models into PROMELA models. The PROMELA models can then be formally verified using the SPIN model checker. Of course, there is a trade-off in using automatic translation. Automatically-generated models are often larger and less efficient than manually-generated models, and require more time and memory to verify.

Model checking typically takes place with respect to properties, which are formulae encoded in a logical language. These properties are based on system requirements. If the properties can be shown to hold for a model, then we take this as evidence that the system satisfies its requirements. In our case, we use linear temporal logic (LTL), which allows the formalisation of concepts relating to time, e.g., "now and at all points in the future" (via the $\Box$ operator), "now or at some point in the future" ($\Diamond$) and "in the next state" ($\bigcirc$) [9]. This enables formalisation of safety requirements (something bad never happens, $\Box\neg$bad), liveness properties (e.g, something good eventually happens, $\Diamond$good) and fairness properties (e.g., if one thing occurs infinitely often so does another, e.g., $\Box\Diamond$send $\Longrightarrow$ $\Box\Diamond$receive). Using *BrahmsToPromela* extends SPIN's property specification language with a *belief* operator, "B". This allows us to specify that an agent has a belief, e.g., $\mathsf{B}_{\text{Rover}}x$ means that the Rover agent believes x is true.

After using BrahmsToPromela to translate the Brahms model to PROMELA, the property in question is appended to the file containing the PROMELA model. Next, SPIN is used to generate 'C' code representing a finite state automaton representing the ASRO teamwork model. This automaton is then compiled and executed. The results of the execution reveal whether there are any errors in the model, i.e., cases where the property is found to not be true. If this is the case, a counter-example highlighting the error is generated and exported to a file for later examination. If errors are found they can usually be traced back to the point in the Brahms model that caused the problem. This allows bugs to be fixed before starting the process again. Eventually we aim to obtain an error-free model that satisfies all of the properties representing the requirements of the system being modelled.

The first version of the model was based on a 12 hour work day in which the astronaut spent eight hours conducting activities inside and outside the habitat, before retiring to the habitat at the end of the day. However this model was found to be too complex to model-check in a reasonable amount of time. In order to reduce the complexity of the model the work day was shortened to five hours. This resulted in a more manageable model that could be model-checked using less time and memory. This modification of the model was essentially an abstraction, and reduced the size of the model by discarding unnecessary details. In this case we discarded seven hours of the work day in which the astronaut and rover were simply repeating their non-deterministic choice a number of times. The primary aim of the formal verification was to determine the correctness and reliability of the rover's responses to the astronaut's non-deterministic behaviours. By removing repetitious behaviour in the model we were able to make the formal verification tractable and achieve the key aim of verifying the astronaut–rover team and the rover's autonomous behaviour. Results of the formal verification of the Brahms astronaut–rover model are as follows.

Model Validation: Astronaut Behaviour

Before we begin to formally verify the autonomous behaviours of the rover, it is important to validate that the model accurately depicts the behaviour of the astronaut in our scenario. As mentioned in Section 2, the astronaut can choose non-deterministically from a range of possible behaviours. This non-deterministic choice can be validated using the following properties:

$$\Diamond\mathsf{B}_{\text{Astronaut}}(\text{goalPerformLeaveHabitat}) \tag{1}$$

$$\Diamond\mathsf{B}_{\text{Astronaut}}(\text{goalPerformGeoSurvey}) \tag{2}$$

$$\Diamond\mathsf{B}_{\text{Astronaut}}(\text{goalPerformMisc}) \tag{3}$$

$$\Diamond\mathsf{B}_{\text{Astronaut}}(\text{goalPerformVideoEVA}) \tag{4}$$

$$\Diamond\mathsf{B}_{\text{Astronaut}}(\text{goalPerformReturnToHabitat}) \tag{5}$$

$$\Diamond\mathsf{B}_{\text{Astronaut}}(\text{goalPerformConstruction}) \tag{6}$$

These six properties were formally verified using *BrahmsToPromela* and SPIN, and were found to hold for the model.

Verification of Mission-Critical Rover Behaviour

Performing solo geological surveys—This requirement concerns the ability of the robot to perform a solo geological survey when the astronaut does not need the direct assistance of the rover. This is encoded as follows:

$$\Box\left[\begin{array}{c}\mathsf{B}_{\text{Astronaut}}(\text{goalPerformMisc}) \Longrightarrow\\ \Diamond\mathsf{B}_{\text{Rover}}(\text{goalSoloGeoSurvey})\end{array}\right] \tag{7}$$

This property states that it is always the case that if the astronaut agent believes that it is performing a miscellaneous activity (i.e., an activity that doesn't require the assistance of the rover), then the rover will perform a solo geological survey after a period of time. This property was found to hold for the model, and therefore this requirement has been verified.

Assisting astronaut during construction— The rover is required to assist the astronaut during construction tasks. This is formalised as follows:

$$\Box\left[\begin{array}{c}\mathsf{B}_{\text{Astronaut}}(\text{goalPerformConstruction}) \Longrightarrow\\ \Diamond\mathsf{B}_{\text{Rover}}(\text{goalAssistConstruction})\end{array}\right] \tag{8}$$

This property states that it is always the case that if the astronaut agent believes that it has a goal to start construction, then the rover agent will form a corresponding goal to assist in the construction task after a period of time. This property was found to hold for the model, and therefore this requirement has been verified.

Verification of Safety-Critical Rover Behaviour

Astronaut monitoring during habitat egress— One of the rover's safety-critical requirements is to monitor the astronaut during high-risk activities, such as habitat egress, which involves airlock depressurisation. We form the following property corresponding to this requirement:

$$\Box\left[\begin{array}{c}\mathsf{B}_{\text{Astronaut}}(\text{goalLeaveHabitat}) \Longrightarrow\\ \Diamond\mathsf{B}_{\text{Rover}}(\text{cameraStream})\end{array}\right] \tag{9}$$

This property states that it is always the case that if the astronaut decides to leave the habitat, that the rover will start monitoring by setting the camera stream variable to true, indicating that a video stream is being sent back to the habitat (and then back to the ground station for monitoring, if needed). This property was found to hold for the model, and therefore this requirement has been verified.

Habitat monitoring during astronaut rest periods— In our scenario the rover is required to monitor the integrity of the habitat while the astronaut is inside. This is the basis for the following property:

$$\Box \left[\begin{array}{l} \text{Astronaut.location} = \text{Habitat} \Longrightarrow \\ \Diamond \mathsf{B}_{\text{Rover}}(\text{goalSoloMonitorHab}) \end{array} \right] \qquad (10)$$

This states that if the astronaut is in the habitat, then the rover will form a goal to autonomously monitor the habitat. This property uses the location feature of the Brahms language that allows each agent to be associated with a particular geographical location identified in the model. This property was found to hold for the model, and therefore this requirement has been verified.

Computational Demands

The computational demands for the properties verified are given in Table 1. The columns refer to the property number, the number of states in the model, the maximum depth reached by the model checker, and the amount of memory and time used. It can be seen that the six "reachability-style" properties (1–6) used for model validation were the least demanding overall. The four other properties for the mission- and safety-critical requirements (7–8 and 9–10 respectively) were more demanding. This is likely to be a result of the more complex formulae used in these properties, combined with a need to examine the entire state space in order to demonstrate that they hold for the model.

Table 1. Computational Demands for Formal Verification of Properties.

Prop.	States	Depth	Mem. (MB)	Time (s)
1	650	1,298	187	0.01
2	1,874	3,746	199	0.05
3	440,960	35,087	3,284	11.7
4	18,475	16,691	331	0.49
5	2,321	4,198	202	0.07
6	4,275	5,710	218	0.12
7	1,348,495	37,569	9,420	37.9
8	1,160,503	37,569	8,140	31.7
9	1,498,157	37,569	10,439	70.3
10	1,105,121	37,569	7,762	30.4

4. Conclusions

In this paper we have shown how a high-level model of astronaut–rover teamwork can be constructed using the Brahms multiagent workflow language, and verified using the *BrahmsToPromela* software tool and the SPIN model checker. The model was based on a scenario in which an astronaut and a rover worked together in a range of activities including construction, geological surveys, video EVAs and miscellaneous activities. The rover responds autonomously to the astronaut's behaviour and will assist wherever needed. When the rover is not needed, for example when the astronaut has specified that this is the case, or when the astronaut is resting in the Habitat, the rover performs autonomous solo functions including habitat integrity modelling and geological surveying. We showed that this model could be translated to PROMELA, the input language for the SPIN model checker, using a software tool called *BrahmsToPromela*. Once the model is translated it is straightforward to use SPIN to model-check the PROMELA code representing the Brahms model of the astronaut–rover teamwork scenario. It was shown that requirements for key mission- and safety-critical behaviours could be encoded using linear temporal logic, as well as requirements that were used for validation of the model. These requirements were found to be satisfied by the PROMELA code, indicating that the Brahms model on which it was based was correct with respect to those requirements. Model checking of the requirements was also shown to be tractable, requiring ~ 70 seconds in the worst case.

Future Work

While this paper establishes an approach to formal verification of high-level descriptions of astronaut–rover teamwork, there is much that remains to be done. An obvious extension of the work would be to increase the accuracy of the model by incorporating even more behaviours for the astronaut and rover that may conflict and overlap, as would presumably be the case in a real-life astronaut–rover scenario, or even to expand the model to include multiple astronauts and rovers. This would potentially increase the complexity of the model, and the time and space required to formally verify it. Therefore it may be useful to investigate the use of abstraction techniques to reduce complexity, e.g., by dividing up the scenario into different parts, thereby effectively partitioning the state space. Another way to improve execution times would be to optimise the *BrahmsToPromela* software tool, which is at a relatively early stage of development. Yet another option would be to use the formal semantics of Brahms and the parser within the *BrahmsToPromela* tool to generate code for a different model checker that has a closer level of abstraction than SPIN. An obvious choice might be the Agent JPF model checker, which we have developed in previous work [20], [21]. Another avenue for future work would be to modify *BrahmsToPromela* to translate Brahms into the input languages of different model checkers that support probabilistic [22] or real-time model checking [23].

This paper uses a high-level model of astronaut–rover teamwork to enable formal verification, but there is much to be gained from combining this kind of approach with other, more complex and non-exhaustive, verification methods. For example, 3D physical simulators like Gazebo[1] can be used to examine the behaviour of robots in much greater detail. However, simulators can rarely explore the system exhaustively as formal verification does. Likewise, physical real-world tests are even more accurate than simulations, but can be very costly. We advocate an approach in which all of these verification methods, as well as hybrid methods like hardware-in-the-loop can be leveraged together and their results checked with respect to each other, in a process known as corroborative verification and validation [24]. In addition, the Brahms model in this paper could be used to direct the autonomous behaviour of a real or simulated astronaut–rover team, e.g., through a Brahms–Java interface using the Robot Operating System[2].

[1]http://gazebosim.org/
[2]https://www.ros.org/

Acknowledgments

This work was supported by (i) the EPSRC under the Future AI and Robotics Hub for Space (EP/R026092/1), and (ii) the Royal Academy of Engineering.

References

[1] J. P. Grotzinger, J. Crisp, A. R. Vasavada, R. C. Anderson, C. J. Baker, R. Barry, D. F. Blake, P. Conrad, K. S. Edgett, B. Ferdowski, R. Gellert, J. B. Gilbert, M. Golombek, J. Gómez-Elvira, D. M. Hassler, L. Jandura, M. Litvak, P. Mahaffy, J. Maki, M. Meyer, M. C. Malin, I. Mitrofanov, J. J. Simmonds, D. Vaniman, R. V. Welch, and R. C. Wiens, "Mars science laboratory mission and science investigation," *Space Science Reviews*, vol. 170, no. 1, pp. 5–56, Sep 2012. [Online]. Available: https://doi.org/10.1007/s11214-012-9892-2

[2] R. C. Trevino, J. J. Kosmo, A. Ross, and N. A. Cabrol, "First Astronaut - Rover Interaction Field Test," in *International Conference On Environmental Systems*, July 2000, pp. 2000–01–2482.

[3] G. A. Landis, "Robots and Humans: Synergy in Planetary Exploration," in *AIP Conference Proceedings*, vol. 654, no. 853, 2003.

[4] L. Pedersen, "Field Demonstration of Surface Human-Robotic Exploration Activity," in *AAAI Spring Symposium*, 2006, p. 9.

[5] M. Sierhuis, "Modeling and Simulating Work Practice. BRAHMS: a multiagent modeling and simulation language for work system analysis and design," Ph.D. dissertation, Social Science and Informatics (SWI), University of Amsterdam, SIKS Dissertation Series No. 2001-10, Amsterdam, The Netherlands, 2001.

[6] M. Sierhuis and W. J. Clancey, "Modeling and simulating work practice: A method for work systems design," *IEEE Intelligent Systems*, vol. 17, no. 5, pp. 32–41, 2002.

[7] M. Sierhuis, W. J. Clancey, R. L. Alena, D. Berrios, S. B. Shum, J. Dowding, J. Graham, R. van Hoof, C. Kaskiris, S. Rupert, and K. S. Tyree, "NASA's Mobile Agents Architecture: A Multi-Agent Workflow and Communication System for Planetary Exploration," in *Proc. of The 8th International Symposium on Artifical Intelligence, Robotics and Automation in Space - iSAIRAS, Munich, Germany.*, 2005.

[8] G. J. Holzmann, *The Spin Model Checker*. Addison-Wesley, November 2003.

[9] M. Fisher, *An Introduction to Practical Formal Methods Using Temporal Logic*. Wiley, 2011.

[10] NASA. (1972) Apollo 17 first test of the lunar rover. Accessed 17 Oct 2019. [Online]. Available: https://commons.wikimedia.org/wiki/File:Apollo_17_-first_test_of_the_lunar_rover_AS17-147-22527HR.jpg

[11] M. Ransan and E. M. Atkins, "A Collaborative Model for Astronaut-Rover Exploration Teams," in *AAAI Spring Symposium*, 2006, pp. 52–58.

[12] A. Medina, C. Pradalier, G. Paar, A. Merlo, S. Ferraris, L. Mollinedo, P. Colmenarejo, and F. Didot, "A servicing rover for planetary outpost assembly," in *Proceedings of Advanced Space Technologies for Robotics and Automation (ASTRA) 2011*, 2011, https://www.esa.int/Enabling_Support/Space_Engineering_Technology/Automation_and_Robotics/Proceedings_of_ASTRA Accessed 15 Oct 2019.

[13] P. Heiskanen, S. Heikkilä, and A. Halme, "Development of a Dynamic Mobile Robot Simulator for Astronaut Assistance," *Proc. 10th ESA Workshop on Advanced Space Technologies for Robotics and Automation (ASTRA)*, p. 9.

[14] T. Fong and I. Nourbakhsh, "Peer-to-Peer Human-Robot Interaction for Space Exploration," in *AAAI Fall Symposium: The Intersection of Cognitive Science and Robotics: From interfaces to Intelligence*, 2004, p. 4.

[15] R. Stocker, L. A. Dennis, C. Dixon, and M. Fisher, "Verification of Brahms Human–Robot Teamwork Models," in *Proc. 13th European Conf. on Logics in Artificial Intelligence*, ser. LNCS, vol. 7519. Springer, 2012, pp. 385–397.

[16] R. Stocker, M. Sierhuis, L. A. Dennis, C. Dixon, and M. Fisher, "A Formal Semantics for Brahms," in *Proc. 12th International Workshop on Computational Logic in Multi-Agent Systems (CLIMA)*, ser. LNCS, vol. 6814. Springer, 2011, pp. 259–274.

[17] M. Webster, C. Dixon, M. Fisher, M. Salem, J. Saunders, K. L. Koay, K. Dautenhahn, and J. Saez-Pons, "Toward reliable autonomous robotic assistants through formal verification: A case study," *IEEE Transactions on Human-Machine Systems*, no. 99, pp. 1–11, 2015.

[18] R. H. Bordini, M. Fisher, and M. Sierhuis, "Formal verification of human-robot teamwork," in *2009 4th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, Mar. 2009, pp. 267–268.

[19] G. Holzmann, "Inspiring applications of Spin," http://spinroot.com/spin/success.html, 2019, accessed 11 Oct 2019.

[20] L. A. Dennis, M. Fisher, M. Webster, and R. H. Bordini, "Model Checking Agent Programming Languages," *Automated Software Engineering*, vol. 19, no. 1, pp. 5–63, 2012.

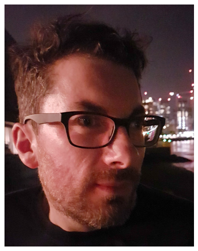
[21] L. A. Dennis, M. Fisher, N. Lincoln, A. Lisitsa, and S. M. Veres, "Practical verification of decision-making in agent-based autonomous systems," *Journal of Automated Software Engineering*, vol. 23, no. 3, pp. 305–359, 2016.

[22] M. Kwiatkowska, G. Norman, and D. Parker, "Prism 4.0: Verification of probabilistic real-time systems," in *Proc. CAV 2011*, ser. LNCS, vol. 6806. Springer, 2011, pp. 585–591.

[23] G. Behrmann, A. David, K. G. Larsen, J. Hakansson, P. Petterson, W. Yi, and M. Hendriks, "Uppaal 4.0," in *Third International Conference on the Quantitative Evaluation of Systems - (QEST'06)*, Sept 2006, pp. 125–126.

[24] M. Webster, D. Western, D. Araiza-Illan, C. Dixon, K. Eder, M. Fisher, and A. G. Pipe, "A Corroborative Approach to Verification and Validation of Human–Robot Teams," *International Journal of Robotics Research*, in press. [Online]. Available: https://doi.org/10.1177/0278364919883338

BIOGRAPHY



Matt Webster *is a senior postdoctoral researcher at the University of Liverpool. With over 15 years of academic and industrial research experience, his research aims to make computer systems safer and more reliable through the development and application of techniques from formal methods. His research interests include verification of AI in space robotics, certification of autonomous unmanned aircraft, the Internet of Things, human-robot interaction, model-checking agent programming languages, computer security and artificial life.*

Louise A. Dennis *is a lecturer in the Autonomy and Verification Laboratory at the University of Liverpool. She is attached to the Centre for Autonomous Systems Technology (CAST) at the University of Liverpool. Her background is in artificial intelligence and more specifically in agent and autonomous systems and automated reasoning. She has worked on the development of several automated reasoning and theorem proving tools, most notably the Agent JPF model checker for BDI agent languages; the lambda-clam proof planning system; and the PROSPER Toolkit for integrating an interactive theorem prover (HOL) with automated reasoning tools (such as SAT solvers) and Case/CAD tools. More recently she has investigated rational agent programming languages and architectures for autonomous systems, with a particular emphasis on verifiable systems and ethical reasoning.*

Clare Dixon *is a Reader in the Department of Computer Science, University of Liverpool, Liverpool, U.K. Her research interests include formal verification and temporal and modal theorem-proving techniques, in particular applied to robotics and autonomous systems. She is a member of the Autonomy and Verification Laboratory and the Centre for Autonomous Systems Technology at the University of Liverpool. She is also a member of the British Standards Institution AMT/10 Committee on Robotics.*

Michael Fisher *holds a Royal Academy of Engineering Chair in Emerging Technologies in the Department of Computer Science of the University of Liverpool. He is the Director of the Universitys Centre for Autonomous Systems Technology and co-chairs the IEEE Technical Committee on the Verification of Autonomous Systems. His a Fellow of both BCS and IET, and is a member of both BSI and IEEE standards committees concerning Robotics and Autonomous Systems.*

Richard Stocker *graduated from the University of Liverpool in 2005, 2009 and 2013 with an Undergraduate degree in Computer Science, a Masters in Advanced Computer Science and a PhD in Formal Verification of Human-Agent-Robot Teamwork. In 2013 he left the UK to work for SGT, Inc as a level 2 Computer Scientist at NASA Ames in California. There he worked on projects for aerospace, particularly simulating workload of single pilot operations and air traffic controllers. Returning to the UK in 2015, Richard started teaching foundation degree students at Middlesex University. Richard then came to the University of Chester to work as a Senior Lecturer in 2017.*

Maarten Sierhuis *is a co-founder and CTO of Ejenta, a San Francisco startup developing an intelligent agent platform with applications in healthcare, insurance and government. Dr. Sierhuis has more than 25 years experience in artificial intelligence and autonomous systems. He is one of the main inventors of the Brahms multi-agent language. As a former Senior Scientist in the Intelligent Systems Division at NASA, he developed autonomous technology for space exploration. He started and headed the Nissan Research Center Silicon Valley to develop autonomous vehicles. Previously, Dr. Sierhuis was director of the Knowledge, Language and Interaction group at the Xerox Palo Alto Research Center (PARC) and a researcher at NYNEX Science & Technology R&D Center. Dr. Sierhuis has a Ph.D. in Artificial Intelligence and Cognitive Science from the University of Amsterdam and a degree in Informatics from The Hague University. Dr. Sierhuis was named #25 of the 100 most creative people in 2015 by Fast Company.*