Safety Protocol Verification of the Canadarm3

Samantha Papais[1]

School of Computing, Queen's University, Kingston, Canada

Abstract

The Canadarm3 will serve as an autonomous robotic system on the Lunar Gateway, performing maintenance, aiding astronauts, and capturing visiting spacecraft. Since the Gateway will not be crewed at all times, verifying the arm's off-nominal behaviours is vital. This model will use hybrid automated planning in PDDL+ to model the arm's functionality, environment, and various protocols it will follow in unlikely scenarios. Previous methods of falsifying such cyber-physical systems used black-box algorithms; however, this model uses a white-box approach in which the functionalities and procedures are known and clearly defined. The falsification of this system takes the form of producing counterexamples to the protocols that have been outlined for the system by intentionally trying to induce failure in the model. [some info here about results when I have them]

1. Introduction

Canada's contribution to Gateway—a lunar-orbiting space station—will be various robotic applications such as the Canadarm3, which will perform actions including maintenance on the craft, aiding astronauts with spacewalks, and catching visiting spacecraft. Since Gateway will be much farther from Earth than the International Space Station and will not be continuously crewed, autonomous function is a necessary and vital component of its design. The use of automated planning in space robotics has many advantages, such as improved system reliability and cost efficiency. However, these benefits are only gained if the agent can function reliably in all situations, including those that are statistically rare. The ways in which a robotic application handles these unlikely scenarios are called its off-nominal behaviours. The goal of this model is to identify gaps in the safety protocols outlined for the Canadarm3; if a fail state can be induced, then the safety protocols may require revision.

This model will use hybrid automated planning to verify the off-nominal behaviours of the Canadarm3 by modelling the functionalities of the arm, the various environmental factors that may affect it, and the protocols it should follow when encountering an unlikely scenario. The first step will be to model the dynamic environment. This step involves modelling the agent's behaviour and the actions it can take when the environment meets specific criteria. It will also describe how these actions affect the environment in turn. The next procedure will be to model the off-nominal behaviours and test the off-nominal behaviours by using the model to find scenarios from which it cannot recover. These failure states can be found by running simulations designed to induce failures. If a plan for a failure state is found, it may indicate that there is an error in the model or that the off-nominal behaviours are incomplete.

2. Background

There are various domains from which the task of verifying the correctness of safety-critical cyber-physical systems (CPS) stems.

Model checking is one such domain. Model checking [4] is an automatic method for verifying a system model against its formal specification with the goal of finding errors during the process of system design and development. Model checking is often used to find counterexamples in a model's design; that is, to find failure states. Current research in using PDDL to verify CPS measures their results against model checkers.

A connection has been established between model checking and automated planning in previous work, such as in Cimatti et al. (1999), who proposed a new approach to planning using methods from software and hardware model checking [8]; however, the process of using PDDL and other planning languages to falsify CPS has been largely unexplored. Edelkamp (2003) analyzed what PDDL might be able to accomplish as a model-checking tool, as well as possible limitations it might have [5]. Through his experiments, he discovered that using planning as a model-checker could introduce a number of benefits, such as higher performance and more concise domain specifications. S. Bogomolov et al. (2014) further connected the use of PDDL and model checking by translating hybrid PDDL+ domains to hybrid automata, allowing them to use hybrid model-checkers to verify their planning problems [6]. Aineto et al. (2022) aimed to explain the behaviour of hybrid systems with PDDL+ since traditional model-checking tools struggle to produce concrete trajectories [7]. They translated their problem into PDDL+ and used planners to find

CISC-813: Automated Planning – Final Project

those trajectories instead. Their results indicated there are advantages, such as greater performance, of using PDDL+ over model-checking tools on complex hybrid systems.

A recent work of Aineto et al. (2023) demonstrated that using PDDL+ to falsify CPS was an effective strategy [3]. One of the biggest limitations of using PDDL as an effective falsification method comes from the fact that falsification algorithms treat CPS as black-box systems due to their complexity. Their method involved using PDDL+ to model a white-box approach. This model most closely builds off of this last approach—it will involve modelling the hybrid domain in PDDL+ and then attempting to create plans that reach a failure state.

3. A Planning Model for Verifying the Safety Protocols of the Canadarm3

- The current model covers the Canadarm3's ability to catch and dock visiting spacecraft and the safety procedures it should follow in case of collisions (will be expanded to other capabilities for final version).
- PDDL+ is being used to model this domain due to the hybrid nature of the problem setting.
- This model treats physical space as if it is 2-dimensional, mostly because ENHSP, the planner being used to find counterexamples, is not equipped to efficiently handle 3-dimensional space.
- There are various discrete actions, continuous processes, and events that work together to simulate the Canadarm3 and its safety protocols. The model, as it currently stands, handles the complex case of what the arm should do if it detects that there may be a collision between itself and another object.
- The functionality currently described surrounds docking spacecraft at the various ports of the Lunar Gateway. Ideally (with no complications), the process of catching and docking a craft should look like this:
    1. detect that there is a craft within sensor range
    2. track the craft
    3. approach the craft until it is close enough that the arm can reach out and grasp it
    4. match velocity with the craft so that reaching towards it is a stable and safe operation
    5. extend the arm out towards the craft
    6. grasp the craft with the arm's "hand"
    7. bring the craft to a free port located on the Gateway
    8. dock the craft to that port
- Consider, though, if the incoming craft is coming too fast and is in danger of colliding with the arm, or if there is space debris floating around that may hit the arm. In these cases, the arm should be able to detect that a collision is imminent and take steps to avoid it. In the safety protocol outlined for this model, the arm's behaviour should be the following:
    1. arm detects that a collision is imminent
    2. arm triggers its safety mode where all normal functionality is disabled, i.e. if it was in the middle of doing something, it should stop and pay attention to the collision
    3. arm should recognize that it is a collision that is imminent, not another hazard
    4. arm should move out of the trajectory of the incoming object until the object has moved a safe distance away
    5. arm should exit its safety mode and continue what it was doing before
- Below is an overview of the various actions, events, and processes that are currently in the model.
- The various actions in the model:
    - `track_craft` will "lock on" to a specific craft that it wants to match velocity with, catch, and dock.
    - `catch_craft` will begin the process of extending the arm to catch the craft.
    - `grasp` will use the "hand" on the end of the arm to grasp the spacecraft.
    - `dock_craft` will begin the process of bringing the craft to one of the ports on the Lunar Gateway and docking it there.
- The various processes in the model:
    - *Note that the `approach_craft` and `match_velocity` processes do not yet make realistic sense. The Gateway (and the Canadarm3 by extension) cannot actually move to be in range of the approaching spacecraft, nor can it be the one to match the velocity of the incoming spacecraft. The Gateway will need to be in perfect orbit around the Moon and thus cannot move or make these adjustments. The visiting craft will have to make these adjustments itself and fall into orbit near the Gateway in order for the catch/dock to be possible.

For this draft, I have left these processes in, though will likely have to revise the way this works for the final deliverable.
  - `approach_craft*` simulates the Gateway moving close enough to the spacecraft to catch it.
  - `match_velocity*` simulates the Gateway matching velocity with the spacecraft that the arm will need to catch.
  - `reach_towards_craft` simulates the continuous action of extending the arm to catch the spacecraft, once the spacecraft has been tracked and has matched its velocity with the Gateway.
  - `collision_avoidance` will move the arm away from the object it is in danger of colliding with.
  - `move_to_dock` simulates moving the craft to a free port once the arm has grasped the craft in its hand.
- The various events in the model:
  - `collision` will trigger when a collision between the arm and another object is imminent. It will trigger the safety mode of the arm, preventing it from completing other actions and processes until the collision is safely avoided.
  - `exit_safety_mode` will exit safety mode only if all collisions have been properly avoided. Indicates that the arm is ready to continue normal execution of actions and processes.
  - `object_detected` will trigger for a given object when the object has entered the range of the arm's sensor.
  - `spacecraft_aligned` will trigger when the arm/Gateway has matched the velocity of the visiting spacecraft. Indicates the end of the `match_velocity` processes and the resulting alignment allows the arm to reach out to the craft.
  - `reached_craft` will trigger when the arm has extended to be close enough to the craft to grasp it. Will end the `reach_towards_craft` process.
  - `arrived_at_port` will trigger when the process of moving towards a port to dock a craft has been complete. Indicates that the craft is ready to be docked.

The above description represents the current state of the model. The domain of this model will be further extended to handle other safety contingencies. For example, the camera system and the sensor system will be further refined within the model so that verification can occur around the protocols if these systems should break.

4. Evaluation

DRAFT VER:

Currently, there are no results to analyze and evaluate; however, there are several predictable outcomes that may result from the use/implementation of this model.

The problem files created for this model will be designed to induce a failure state in the functionality of the Canadarm3. However, the planner may never be able to find a plan—this would indicate that the safety protocols implemented are appropriate and sufficient for the functioning of the arm. Though, it is worth noting that this result does not necessarily guarantee the robustness of the Canadarm3's off-nominal behaviours—the planner just may not be able to finish running because the problem space is too large. In general, it is very difficult to demonstrate complete safety with large domains.

Another case would be if the planner is able to find a plan—that is, the planner is able to induce failure in the system. If the planner is able to find a plan, then that means that the safety protocols implemented in the system are not sufficient, and will have to be revised. This would also indicate that the model is doing what it is intended to do—find gaps in the off-nominal behaviours of the Canadarm3. Note that this relies on the fact that the model is implemented otherwise correctly and with safety protocols that are somewhat reasonable for the system.

This section of the paper will be used to analyze the various problem files given to the planner and evaluate what the results mean/what the cause for the result was.

5. Related Work

There is little recent research specifically exploring the use of PDDL+ for falsifying CPS. The most relevant contribution in this area is by Aineto et al. (2023), who presented one of the first attempts to apply automated planning with PDDL+ for system falsification [3]. As discussed previously, they used PDDL+ to model hybrid systems and find counterexamples that violate the systems' safety requirements. They were able to demonstrate that PDDL+-based falsification performs competitively against Falstar, a state-of-the-art falsification tool.

Beyond using PDDL+, other recent research has employed various related methods to verify CPS. For example, Yamagata et al. (2018) proposed a two-layered falsification framework guided by Monte Carlo Tree Search (MCTS) [1]. This method efficiently explores continuous input spaces to identify behaviors that lead to property violations; however, it still considers the hybrid system to be a black-box, similar to previous methods of model checking. MCTS balances the exploration and exploitation of the input space effectively, demonstrated in their results by the out-performance of MCTS over other search strategies.

Akazaki et al. (2018) introduced a deep reinforcement learning approach to falsification, turning the problem into a learning task in which an agent discovers adversarial inputs that trigger unsafe system states [2]. Like the approach taken by this model, they used their agent to find counterexamples to the intended functionality of the CPS. Due to the nature of reinforcement learning, this approach also treats the CPS like a black-box, where the agent learning about specific inputs and actions would lead to a violation. One of their main goals was to reduce the number of simulations while also deeply exploring the problem space. Like most other black-box methods, this approach is limited in its ability to be easily interpreted and understood.

6. Summary

Brief summary so far:

- Motivation for the model: ensure the safe and reliable function of the Canadarm3's autonomous operations through
- Previous work, briefly: prior work showed that using PDDL+ to falsify CPS is a viable and efficient strategy
- Goal: to falsify CPS by modelling in PDDL+ and then trying to find a plan that would lead to failure
- Method in more detail: modelled the Canadarm3's functionality and safety protocols in PDDL+. Used ENHSP to find failure states.
- Results: pending

6.1. Future Work

This model does not encompass the full capability Canadarm3, nor all of the safety protocols it would need to be considered safe. The following are limitations of this model that could be used as motivation for future work:

- This model does not take into account the second, smaller arm that the Canadarm3 will have attached to it, meant for repairing the larger arm to reduce the need for spacewalks by astronauts.
- This model only considers a 2-dimensional space.
- This model treats the arm as if it does not have joints (aside from the one swivel point at the "hand" where it is attached to the Gateway). In reality, the Canadarm3 will have seven degrees of freedom.

7. References

NOTE: these aren't organized or complete yet.

[1] Two-Layered Falsification of Hybrid Systems Guided by Monte Carlo Tree Search https://ieeexplore.ieee.org/abstract/document/8418450).

[2] Akazaki, T.; Liu, S.; Yamagata, Y.; Duan, Y.; and Hao, J.2018. Falsification of cyber-physical systems using deep reinforcement learning. In International Symposium on Formal Methods, 456–465. Springer.

[3] D. Aineto, E. Scala, E. Onaindia, and I. Serina, "Falsification of cyber-physical systems using pddl+ planning," Proceedings of the International Conference on Automated Planning and Scheduling, vol. 33, no. 1, pp. 2–6, Jul. 2023. [Online]. Available: https://ojs.aaai.org/index.php/ICAPS/article/view/27172

[4] E. M. Clarke, "Model checking," in International conference on foundations of software technology and theoretical computer science. Springer, 1997, pp. 54–56.

[5] S. Edelkamp, "Limits and possibilities of pddl for model checking software," in International Conference on Automated Planning and Scheduling, 2003.

[6] S. Bogomolov, D. Magazzeni, A. Podelski, and M. Wehrle, "Planning as model checking in hybrid domains," Proceedings of the AAAI Conference on Artificial Intelligence, vol. 28, no. 1, Jun. 2014. [Online]. Available: https://ojs.aaai.org/index.php/AAAI/article/view/9037

[7] D. Aineto, E. Onaindia, M. Ramirez, E. Scala, and I. Serina, "Explaining the behaviour of hybrid systems with pddl+ planning," in Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence (IJCAI-22), 07 2022, pp. 4542–4548.

[8] Cimatti https://www.researchgate.net/publication/2800895_Planning_via_Model_Checking_A_Decision_Procedure_for_AR