Commentary on the paper *"Experiment on Verification of a Planetary Rover Controller"* by A. Akhavan, S. Bensalem, M. Bozga, E. Orfanidou

Commentator: Marco Pistore
DIT - University of Trento - Italy
pistore@dit.unitn.it

This paper addresses the problem of checking the correctness of the K9 Rover Executive, an experimental platform for autonomous vehicles developed at NASA Ames and targeted for the exploration of the Martian surface. The Executive provides a means to command the vehicle through the execution of plans that control the movements, the experimental apparatus, and the other resources of the rover. A plan defines a hierarchical description of vehicle's actions. Actions and tasks are annotated with time constraints on their starting and completion time.

The verification of the Executive has to face two sources of unpredictability.

- First, different plans can be defined to command the rover. Indeed, the main aim of the Executive is to allow for the definition and upload of the plans during the mission of the rover.
- Second, each plan can generate different non-deterministic execution paths, depending on the outcomes of action execution and on the events that happen in the rover's environment. The plan language allows for branching depending on conditions that need to be checked at execution time; moreover, the plans allow for flexibility with respect to the duration of action execution; finally, the plans allow for a basic mechanism for managing the possibility of failure of command actions (a plan may define whether the failure of an action has to be ignored or whether it leads to the failure of the task in the hierarchy which has triggered the action).

The Executive is correct if, for every possible plan, all the executions of the plan allowed by the Executive are compatible with the plan semantics. This means, for instance, that an execution of the plan can trigger an action only if the corresponding start condition is satisfied. And that the execution of a given sequence of actions has to be stopped if one of the actions has failed and the recovery mechanism requires to break the whole task.

Different approaches are possible for checking the correctness of the Executive. One approach is testing: given a certain plan, and an execution of this plan in a given configuration of the environment, one can check whether the execution is correct. By repeating this check on different configurations and on different plans, one can increase the level of confidence in the correctness on the Executive. Another approach consists in formally verifying the correctness, using e.g., model checking or theorem proving techniques. This may be hard or impossible due to the complexity of the Executive. Moreover, in most cases formal verification is partial, since it is based on an abstract model of the Executive.

A third approach, followed in the paper, is to check, given a plan, that all the executions of that plan are correct. This is a compromise between the former two approaches: similarly to testing, it only checks correctness for specific plans; however, the correctness of the Executive with respect to the given plans is formally verified by considering all their possible executions.

This paper focuses on a specific form of correctness, namely correctness with respect to the time constraints. However, the proposed approach, and the following comments, are general and apply also to other forms of correctness.

The paper describes the approach in a clear and precise way, but does not provide enough information to permit an evaluation of the proposed technique and of its practical applicability. In the following, we propose some criteria that can be adopted for evaluating the approach and for comparing it with other verification approaches.

Complexity of the verification task

From a theoretical point of view, the verification that the Executive is correct for all plans is more complex than the verification for a given plan. Indeed the latter kind of correctness is a corollary of the former. However, one could claim that, if we consider plans of high complexity (e.g., with a big number of constraints, a complex pattern of non-deterministic behaviors, complicated time constrains...), then the verification of the correctness for a plan may be as difficult as the verification in the general case. If this claim is wrong in the specific domain, it would be important to understand what makes the verification of a specific plan intrinsically simpler.

A similar comment also applies from a practical perspective. If plans become too complex, then their verification becomes unfeasible. What is the complexity of plans for which the verification is still feasible in practice? How does this complexity compare with the plans that one expects to deal with in practice in the application domain?

An approach for managing more complex plans is to adopt stronger simplifying assumptions and abstractions when modeling the Executive and the plans. If the model of the Executive is very abstract, however, the verification of its correctness becomes possible also in the general case. One has to identify the right level of abstraction, which makes the general verification case still unfeasible, but which allows for a practical verification of specific plans. Does the level of abstraction adopted in the paper satisfy this condition?

Effectiveness in discovering bugs

The main criterion for evaluating a verification technique is its effectiveness in detecting bugs. A question not answered in the paper is if the proposed technique has been able to detect bugs, either genuine or artificially injected in the Executive. Or, more in general, what are the kinds of bugs for which the authors expect their technique to be more efficient than testing and general verification.

Looking to this problem from another point of view, a possible way for comparing the effectiveness of the proposed approach is to identify different classes of bugs which are relevant for the application domain and to see which technique is more efficient in detecting these bugs in a given amount of time. We remark that also formal verification techniques have to be considered in this "bug hunting" competition. Indeed, some of these techniques (e.g., different kinds of model checking techniques) do not offer only "all-or-nothing" verification but have been proved to be able to detect bugs also in models that are too big to be proved correct.

Practical applicability of the approach

One of the strongest aspects of the proposed approach is that it fits the verification needs of the specific application domains. Indeed, the goal is to guarantee that, once a plan is uploaded to the rover, then all its possible executions are correct. Testing is insufficient to this purpose, since it may skip executions that can occur in practice. A complete verification of the executor is overkill, since it verifies the correctness of the Executive also for plans that it will never need to execute. (In certain cases, it might be preferable to implement a simplified but efficient Executive that is able to run correctly only a subset of the possible plans, if these plans are sufficient for all practical purposes, and if it is possible to detect and block invalid plans before execution.)

However, the time constraints on the verification are radically different in the cases one wants to verify the correctness of the Executive or of the specific plans. Indeed, in the former case the verification can be done off-line, before the mission of the rover begins, and can take months to complete. In the latter case, the verification has to be done between the generation of a plan and its upload to the rover, that is, in a couple of hours, according to the current practice. An important question to be answered is whether the proposed approach will be able to fit these strong time constraints for the verification.

To conclude, a complete answer to the questions above is clearly outside the scope of the paper. However, taking into account these questions — and practical applicability in particular — is of uttermost importance for proposing viable verification approaches for autonomous space applications.