Model Checking Autonomy Models for a Martian Propellant Production Plant

Peter Engrand
NASA, Kennedy Space Center FL
Peter.engrand-1@ksc.nasa.gov

Abstract - Expanded exploration of our Solar System will require more sophisticated autonomous assets to be developed and deployed. Model based autonomous control systems are a primary technology solution to this problem. A critical factor in the successful operations of these systems is to ensure that the models behave correctly. The Kennedy Space Center (KSC) has been pursing in conjunction with Ames Research Center the application of model-checking techniques for an Intelligent Systems Software for an In-Situ Resource Utilization (ISRU) plant for future manned Mars missions. Model checking is a formal technique which can exhaustively evaluate a finite state model for satisfiability of a logical property. The main goal of our current model checking effort is to develop tools and methodologies for efficient evaluation and certification of future Livingstone modeling applications which are declarative in form. As a result of this investigation a potential new re-usable specification pattern was derived which allows one to check a model for the existence of correct variable dependencies within the model.

Introduction – The human exploration of Mars has the potential to re-ignite public interest in the space program. As NASA moves towards the completion of the International Space Station, work is underway to prepare mankind for the next step in the exploration and colonization of our solar system. The Kennedy Space Center (KSC) has been pursing in conjunction with Ames Research Center application of Intelligent Systems Software to an In-Situ Resource Utilization (ISRU) plant based on the Reverse Water Gas Shift reaction (RWGS). This plant, built in KSC's Applied Chemistry Laboratory, is capable of producing a large amount of Oxygen with a small quantity of seed Hydrogen. In a human Mars mission, this plant would be required to operate for 500 or more days without human intervention. KSC has considerable experience applying intelligent systems to launch processing operations. This experience is being used to apply Model-Based Reasoning technologies to the control of the ISRU plant. The heart of the RWGS intelligent system is a high-level system model of the test bed written in the Livingstone modeling language. These models are used by intelligent control agents for test-bed state identification and mode recovery operations.

The decision to use model-based technology as part of the overall control software approach adds challenges to the conventional software engineering process. These challenges arise from the relative novelty of applying this approach into an operational environment (i.e. it is non-conventional, instead of employing a more explicit, quantitative, procedural approach to modeling we are now using a qualitative, logically constrained approach). With this in mind, the focus of this investigation was to explore optimal methodologies for applying model checking methods for the verification of Livingstone models for ISRU control.

Part of the RWGS model verification process being employed on these models is to use analytical methods and tools, in particular the use of temporal logic and model checking. The models are encoded as finite-state machines which makes them amenable to verification and validation using model-checking tools. Desired properties which the software models should exhibit are represented by temporal formulas. The task of the model-checker is to see if the model satisfies the temporal formula. Currently, analytical verification and debugging of RWGS Livingstone models employs a model checking tool called SMV as the preferred tool of choice due to its ability to explore entire state spaces efficiently (which is one of the raison-d'être of the model checking approach in addition to its relative amount of automation). This paper reviews the results of this analytical effort so far at KSC.

This paper will approach the model checking task in a layered approach beginning first with a discussion of the Kennedy Space centers Spaceport Engineering concept. The next section will begin the describing the application by describing a manned Mars Mission model which includes an in-situ autonomous propellant production plant. This will be followed by a description of KSC's Reverse water Gas Shift prototype which serves as a technology test-bed for the application of In-Situ resource utilization and autonomous control. The next section will discuss the method of autonomous control using the Ames' Model-Based Autonomous control System called Livingstone (a fuller treatment of this topic can be read in Larson & Goodrich [3]). The rest of the paper deals with the practical issues encountered using model checking techniques to evaluate the Livingstone RWGS models.

KSC Spaceport Engineering Concept

Historically the Kennedy Space Center has been one of the planet's pre-eminent launch sites with over fifty years of experience launching both manned and un-manned space vehicles. Over this past half century it has been found that the majority of life cycle costs at a launch site are attributable to operations and support activities (e.g. launch vehicle and payload assembly,

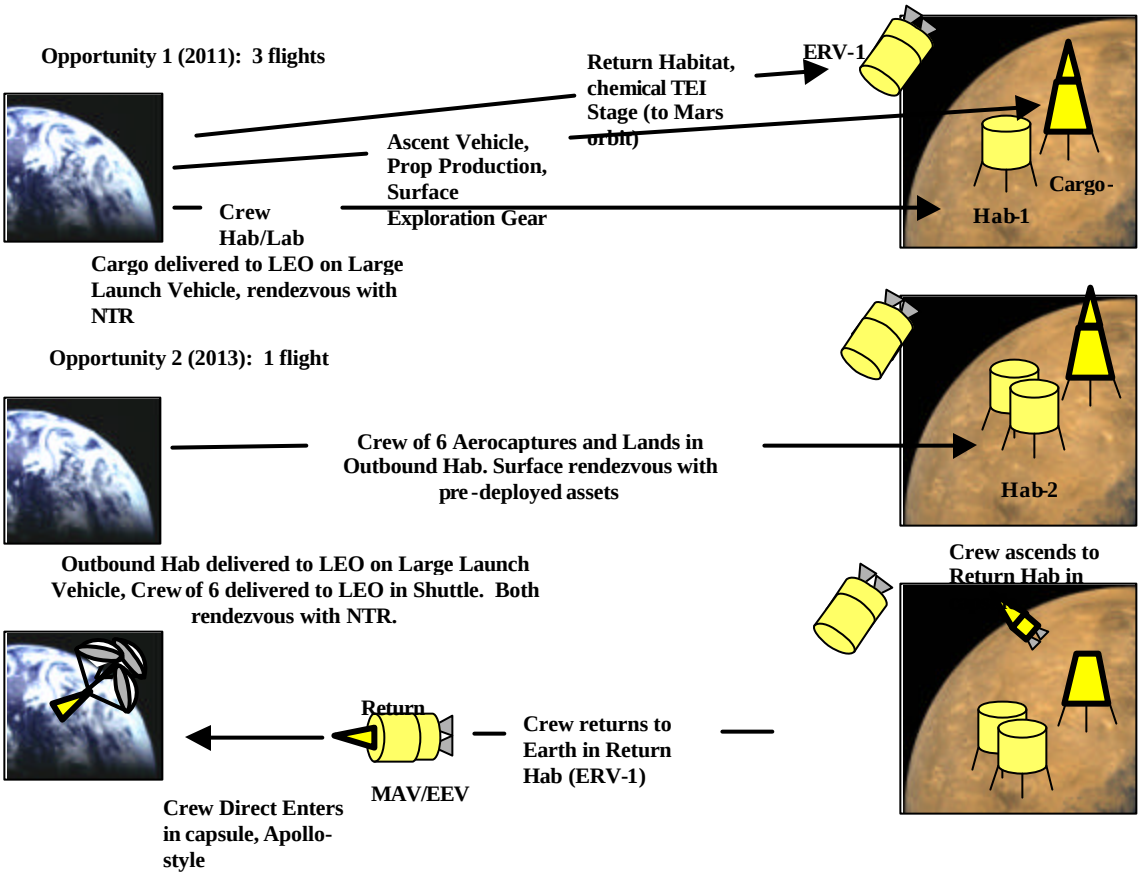

Figure 1. Manned Mars Mission Model

integration, test &Checkout, fueling etc.). Due to the complex and sometimes hazardous nature of launching a mass into space, a large complement of skilled technical personnel are required for a safe and successful launch. A key goal of the agency is to provide more efficient exploitation of space by reducing the costs of accessing space. Given the current labor intensive approach to launch site operations, greater degrees of automation must be employed to increase operational efficiencies. To that end KSC is focusing on developing automated launch site technologies which not only help to reduce cost and enhance safety but also to develop technology that will be applicable across programs and environments (including extra-terrestrial launch sites). Dr. Zubrin, in his book *The Case for Mars* [1], calls for the use of indigenous resources to lower the mass that must be carried to Low Earth Orbit. This concept, called In-Situ Resource Utilization (ISRU), has been captured in NASA's new design reference mission which envisions an initial deployment of a robotic fuel production facility and depot two years prior to a manned landing. Figure 1 shows a graphic summary of this mission concept. There are considerable advantages to using indigenous resources. One of the most significant drivers for the size of a Mars exploration launch vehicle is the amount of mass you need to carry to Mars and back. Calculations show that for every pound carried to Mars and back, forty (40) pounds must initially be carried to low earth orbit. The most significant mass fraction of any launch vehicle is the fuel it carries. Major savings can be achieved by making the fuel for the ascent from the Martian surface while on Mars. This reduces the size of the Mars Lander's engines and the fuel mass carried. This savings ripples back through the entire mission architecture, reducing the size of the launch vehicle, transfer stage, etc..

The Reverse Water Gas Shift (RWGS) is one potential solution for the production of propellants on Mars. The reaction works as follows: Martian atmospheric carbon dioxide is combined with hydrogen (brought from earth) in the following reaction.

$$CO_2 + H_2 = CO + H_2O$$
$$DH = +9 \text{ kcal/mole}$$

The water vapor produced, is condensed and collected in various water trap tanks and is electrolyzed, the oxygen is stored and the hydrogen is recovered and re-circulated into the input stream. Since all the hydrogen is reused,

Autonomous Control of a RWGS System

The RWGS test bed is designed for unattended ISRU, and its control system illustrates many features and benefits of intelligent software.

RWGS control strategy To achieve maximum oxygen production, the RWGS system must operate at its full design capacity as determined by such factors as the reactor size and membrane surface area. The system must control the feed flow relationship between hydrogen and carbon dioxide carefully so as not to waste reactant or slow production. If component degradation occurs, the autonomous system must redirect flows to adapt to changing circumstances. Understanding of this strategy is of fundamental importance in deriving specifications for the model checking task

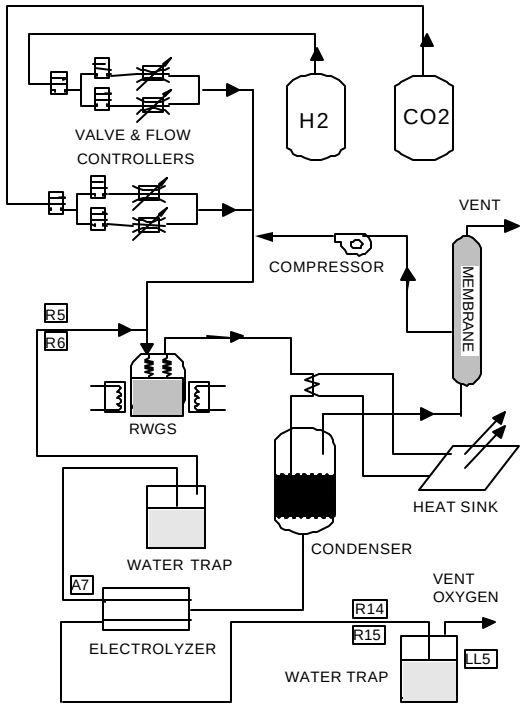

Figure 2. RWGS Schematic

RWGS and Livingstone

The RWGS test bed uses Livingstone monitoring and diagnosis software developed at Ames Research Center. Ames has been working with KSC to apply Livingstone to ISRU since 1998. The software provides built-in autonomy capabilities for RWGS. The heart of the RWGS intelligent system is a high-level system model of the test bed written in the Livingstone modeling language. The model is a simple, declarative statement of the behavior of RWGS components and the connections between them. Information from the design of the test bed is simply translated, part-by-part and concept-by-concept into Livingstone statements. As an example of a Livingstone model a valve module containing mass flow controllers (FC1 and FC2) and solenoid valves (SV1, SV2 and SV3) regulates the gas flow as shown in figure 3. The Livingstone model is hierarchical. A flow branch consists of a two-way solenoid valve and a mass flow controller. Two flow branches are connected to a three-way solenoid to make a flow module. The behavior of the solenoid valves, and flow controllers are modeled as components. The engineer defines states for the valve components corresponding to various normal and abnormal operating modes (as in shown in figure 4). The labels on the links correspond to device commands. Logical propositions define the behavior of the valve while it is in the associated mode. One of the key benefits of this modeling paradigm is that the engineer is only responsible for describing the *local* behavior of each component, the relationships that exist between components and any "system concepts" such as mass and/or energy balances that affect operation. Livingstone then uses this specification to compose a larger, system model that can be used to reason about the *global* behavior of the entire system given the mode of each component. Once the model is complete and connected to test bed instrumentation, the advisory and autonomy features of the Livingstone engine are available for use. These uses include system health monitoring, diagnosis of component failures, flexible reconfiguration, redundancy management, adaptability to degraded environments, and tolerance for component faults and incomplete sensor information.

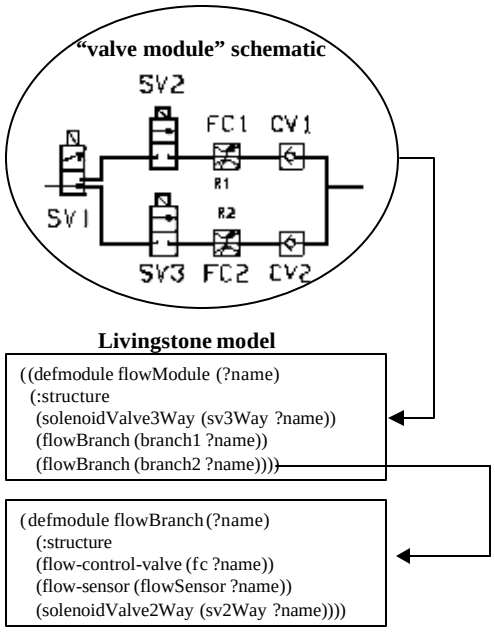

Figure 3. Translating a schematic into Livingstone

Model-Checking the Prototype RWGS Models

Due to the extended duration of autonomous operations correct system control depends critically on the correctness of the Livingstone models. Conventional approaches to verification of models which rely on informal testing techniques ,though straight forward to implement, suffer from a lack of completeness when attempting to answer questions dealing with global model behaviors. Livingstone models are declarative in nature meaning that instead of enumerating all behaviors the developers simply state constraints on system behavior leaving specific behaviors implicit.

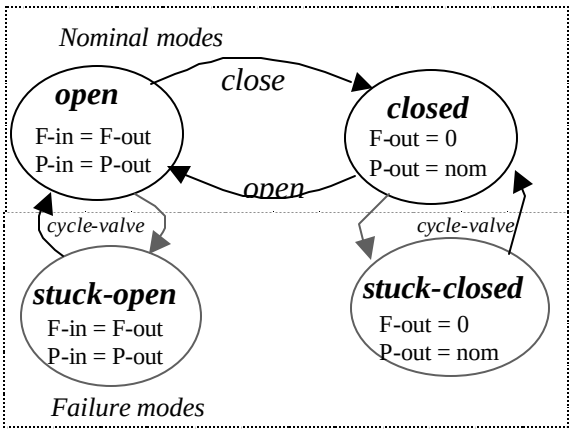

Figure 4. Valve Model

Although a Livingstone controller is inherently more complex than most conventional control software applications, the fact that it's models are represented as a finite state machines make it conducive for model-checking techniques. A caveat must be stated, though, in that although model-checking does allows for an exhaustive search of all states in a model, the obvious drawback is the size of the state-space being checked (otherwise known as the "State Explosion Problem").

One of the fundamental issues for effective verification of models deals with the construction of meaningful specifications that verify correct behavior of the model (where a rough and ready definition of "meaningful" is a temporal form which captures some global system behavior, for example high level design specification or requirements). The specifications should be structured in order to detect three types of modeling errors. These errors include, 1) errors in which results are rejected although in fact they are sufficiently credible (this is referred to as "Model Builder's Risk"), 2) errors in which invalid results are accepted, even though they are not sufficiently credible (Model Users Risk), and 3) Errors in which an invalid problem is incorrectly formulated and the wrong problem is solved generating an irrelevant modeling result. In order to detect these faults, the verifier must have an adequate understanding of RWGS behavior in order to generate effective verification specifications.

As stated previously, the focus of this investigation is to explore optimal methodologies for applying model checking methods for verifying Livingstone models . For our purposes verification can be defined as: The process of determining that the Livingstone RWGS model implementation accurately represents the developer's conceptual description and specification. Before verification is performed the set of verification specifications need to be stated for which the Livingstone RWGS model must satisfy. These specifications are given a logical formalism which assert how the system evolves over time. Temporal logic is commonly used for this purpose where the meaning of a formula is determined with respect to a transition graph. Computational Tree Logic (CTL) is the particular type of temporal logic used for specification representation which models time as having a branching structure. The specific model checking tool used is SMV developed at Carnegie Mellon University (CMU). SMV is based on a language for describing hierarchical finite–state concurrent systems. In order to perform verification on the Livingstone RWGS models the specification as well as the Livingstone model must first be translated into an SMV program. This task is performed automatically by a translator developed by both CMU and the Ames Software Engineering Group (see Pecheur & Simmons [5]). Once the translation of both model and specification have occurred SMV then proceeds to check that the model satisfies the specification. If the result is negative SMV will provide the user with a trace which can be used as a counter-example for the checked property to aid the developer in tracking down the source of the error.

Statement of Formal Verification Problem :

The verification problem for Livingstone models can be schematized as shown in fig. 5 . In words what the schematic states is the following :
Given :

1. A Livingstone model which is an abstraction of some physical system.
2. A mathematical model of that physical system ;

Find a corresponding abstraction that maps from the physical property to the CTL property such that, the Livingstone model satisfies the CTL property iff the physical property satisfies the physical system.

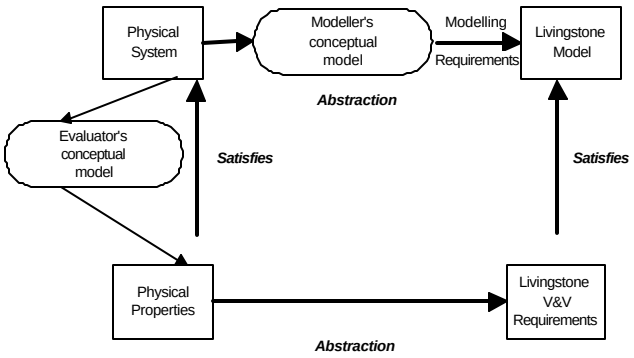

Figure 5.

Another way of stating this is the following :

1. Given a physical system governed by physical laws,
2. A Livingstone model is abstracted from physical system
3. The person performing the verification expresses properties of the physical system (e.g. ODEs, PDEs, Discrete transitions etc.)
4. Abstract CTL properties from the physical representations (properties) such that …
5. the model satisfies the CTL properties (i.e. Livingstone verification requirements) …..
6. If and only if, the Physical system satisfies the stated Physical properties.

This formal evaluation effort focused on observing and studying various modeling faults. These faults were then studied to understand the underlying causes of the faults phenomena so that specification properties could be inferred that could lead to future re-usable specification patterns.

These observations consisted of two modeling faults (which were instances of the previously explained type 2 fault). These two faults, which had been previously encountered by the modelers, had their causes already known prior to the beginning of the formal investigation. Working with known faults and their causes, allowed us insight into understanding and development of appropriate model-checking techniques applicable for these kinds of faults.

The first fault to be investigated dealt with a mis-modeled fluid flow behavior which existed in an earlier version of an ISRU system. This system called the Sabatier-Electrolysis works as follows: carbon dioxide from the Martian atmosphere is reacted with hydrogen (brought from earth). Carbon dioxide is obtained by flowing Martian atmosphere through a component called a sorption pump which contained a catalytic bed that absorbs the CO2 content of the atmosphere. Once the bed is saturated atmospheric in-flow is stopped and the absorbed CO2 is extracted from the catalytic bed through a heating process and pumped through a set of flow branches to the rest of the system where Liquid Methane (fuel) and oxygen (oxidizer) are produced and stored.

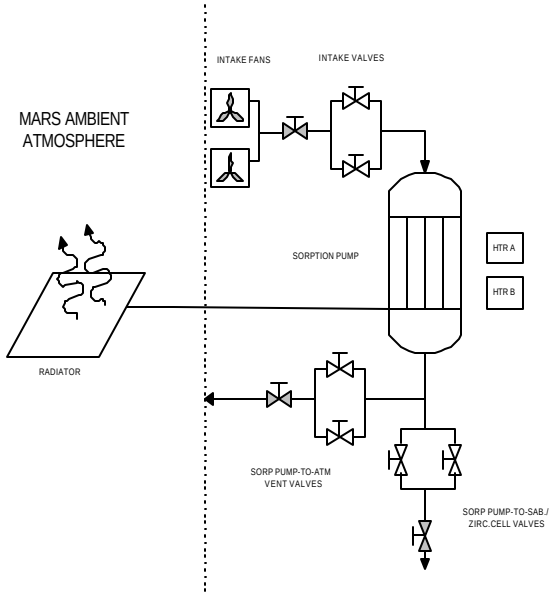

Figure 6. Schematic showing Sorption pump and neighboring components for a Sabatier-Electrolysis ISRU

The modeling fault was manifested in the way the flow branch outlet of the Sorption pump was modeled. In this case, the developers failed to take into account a fluid flow property called admittance of this flow-branch into the overall computation of admittance in the model (admittance is a measure of fluid impedance of pipes and orifices). This omission led to the model manifesting flow through a flow-branch when there should have been none (i.e. in the situation where a valve upstream of a pipe, which controlled flow through the flow branch, was in a closed state the model showed flow being transmitted through the branch). In order to detect this fault using CTL, a simple property was specified which stated the correct behavior of flow through the flow branch when the control valve was closed . I.e.

It should always be the case that if the admittance outlet of the pump is off (i.e. no flow is exiting from the Sorption Pump) then there will be no flow through the component "z-flow-module".

Where "z-flow-module" is the component name of the specific flow branch (Sorp pump-to-Sab/zirc Cell Valves in fig. 6). Or in CTL form as,

(AG (admittance outlet = off $\Rightarrow$ flow z-flow-module = off))

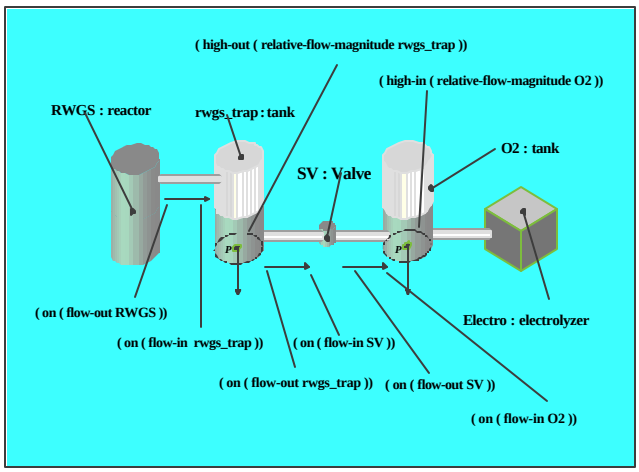

Figure 7. RWGS Schematic for 2nd fault

When the Livingstone model and specification were translated and processed by SMV, a counter-example is returned. The counter example demonstrated the sought for admittance problem by showing a violation of the specification (i.e. outlet admittance = off and z_flow-module = high.

The second, and more interesting, fault is described as follows. The model in which this fault occurred is schematized in figure 7. Figure 7 shows a schematic of a preliminary RWGS Livingstone model, showing the RWGS reactor connected to a condenser tank (RWGS trap) , control valve (SV), another condenser tank (O2) and into the electrolyzer. In this situation the developers wished to model flow between the condenser tanks via a simple set of constraints. A constraint was needed that described the condition in which the contents of the RWGS trap emptied into the O2 trap based only on the magnitude of their respective relative flow magnitudes. In Livingstone, one way of representing this behavior was as an invariant (called a fact in Livingstone) such as,

```
;Fact
    (And
        (When (on (flow-out rwgs_trap))
              (high-out (relative flow-magnitude
              rwgs_trap)))

        (When (on (flow-in O2))
(high-in (relative-flow-magnitude O2)))
```

Where this fact states that if there is an out-flow of fluid from the RWGS trap then the RWGS trap is in an emptying state (high-out) and if their should be flow into the O2 tank then it should be in filling state (high-in). In fact what was found by the developers during their normal testing was that when the O2_trap was filling the RWGS trap was also (incorrectly) filling. In reviewing the models the developers discovered a discrepancy in the way this flow behavior was constrained. In practice what was modeled was the following,

```
;Facts
   (And
       (When (on (flow-out rwgs_trap))
             (high-out (relative flow-magnitude rwgs_trap)))

       (When (on (flow-in O2))
          (high-in (relative-flow-magnitude rwgs_trap)))
```

As can be seen by comparing with the first fact the error lies in the proposition sub-expression," *(high-in (relative-flow-magnitude rwgs_trap))* . Here the argument rwgs_trap is mistakenly in the place of where the O2 trap argument name should be.

Though an obvious error that one would expect to detect easily, this "mis-naming" fault proved to be difficult to detect both using testing by the developer and model checking with SMV. This was due to the fact that by this single object name change a new (and undesired) constraint was added, i.e.

```
(When (on (flow-in O2))
        (high-in (relative-flow-
            magnitude rwgs_trap)))
```

and the desired constraint was non-existent, i.e.

```
(When (on (flow-in O2))
        (high-in (relative-flow-
            magnitude O2)).
```

In order to detect this error in CTL the concept of *variable dependency* was used. Variable dependency is a way of checking relations introduced in order to approximate the continuous physical constraints that are manifest in the RWGS. When we use the term variable dependency what we aim in defining is a functional dependency between Livingstone model variables (i.e. a variable Y functionally depends on variable X if for a given X there can be only one Y).

This dependency can only exist if it has first been established that Y causally depends on X in the physical domain. A Livingstone model does not causal dependencies explicitly; everything is expressed as logical constraints, and one can only look for functional dependency where causal dependency is already known to exist. Therefore if one wants to check that a value of some attribute y is indeed a function of attribute x then one can state this in CTL as,

$$(EF (\text{some } x \,\&\, \text{some } y)) \Rightarrow (AG (\text{some } x \Rightarrow \text{some } y))$$

In other words, if for some value of x you get some value of y, then for the same x you will always get the same value.

The specific dependency we wish to check for is the nature of the relative flow of the RWGS trap. I.e.

Specification :

" some-flow-in, some-flow-out, some-relative-flow, relative-flow

EF (flow-in rwgs-trap = some-flow-in & flow-out rwgs-trap = some-flow-out & relative-flow rwgs-trap = some-relative-flow rwgs-trap) $\Rightarrow$

AG (flow-in rwgs-trap = some-flow-in & flow-out rwgs-trap = some-flow-out $\Rightarrow$ *relative-flow rwgs-trap= some-relative-flow rwgs-trap)*

As can be seen this specification is a disjunction of two separate propositions which would require SMV to produce two separate traces, one for each proposition in order to provide a complete counter-example for this property, i.e.

$$EF\ p \Rightarrow AG\ q = AG \sim p \cup AG\ q.$$

By design SMV only will give a single counter-example trace, that for the EF p case. In practice using SMV, applying this specification needs to proceed in a two step manner. First, the entire specification, as stated above, was applied, returning a one-state trace showing an example of EF p.

Flow-out rwgs-trap = off
Flow-in rwgs-trap = off
Relative-flow-magnitude = equal

Figure 8. First Counter –Example
EF (some x & some y) is true

The second step involved getting a counter-example of AG ~p and for the property as a whole. In order to do this the second half of the specification was re-instantiated as a separate specification using the flow and relative-flow values from the first counter-example. Application of this new specification generated the second counter-example trace showing a two state trace beginning with the RWGS trap in a quiescent state and then transitioning to a filling state even though there was no flow into or out of the trap

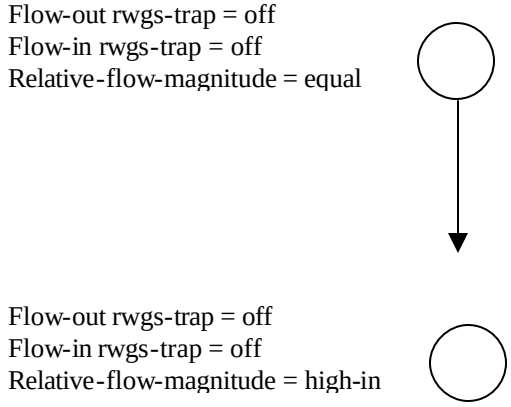

Figure 9. 2^{nd} counter –example

Inspection of the error trace then demonstrated a problem with the relative-flow magnitude of the RWGS trap showing a violation of the specification, hence flagging a problem with the RWGS trap's relative flow magnitude constraint.

Results & Observations

The first three observations deal with the computational efficiency of the model checker.

- The size of the RWGS Model used which contained these two faults scenarios is @ 10^{17} States. Although this is a relatively large state space SMV needed less than a minute to return a result. This was due to the structure of the Livingstone model itself which was relatively "flat" (i.e. the maximum depth of transitions were no more than three deep).

- The size of the Latest RWGS Model is estimated @ 10^{55} states

- An enhancement to SMV by Bwolen Yang [4], greatly enhances the memory performance of SMV for the larger version of the model.

- Writing a correct temporal specifications can be a subtle an error prone task. Part of the task of writing temporal specifications requires a need to "V&V" the specification itself in order to check that the syntax and form of the formula truly conveys the correct semantics of the property one wants to check for.

- As a result of this investigation of variable dependency of the RWGS trap a new specification pattern was derived which allows one to check in a

model that the correct variable dependencies exist (and that valid causality constraints exist among system components and modules) within the model. A property specification pattern is a generalized description of a commonly occurring requirement on the permissible state/event sequences in a finite-state model of a system. A property specification pattern describes the essential structure of some aspect of a system's behavior and provides expressions of this behavior in a range of common formalisms (see Dwyer, Avrunim, Corbett [2]). One can think of a specification pattern as a generic correctness property for a class of objects, such as the property of avoidance of dead-locks in concurrent/parallel systems. In terms of model checking and SMV these two type 2 errors characterized properties of the models which were static in nature and did not involve reasoning about the state transition properties of the model which is the forte of SMV. The schematic in figure 10 summarizes the methodology for implementing this pattern within SMV.

- As an added benefit during the investigation using variable dependency, two additional modeling faults were detected which went undetected during the developers normal testing phase of the model.

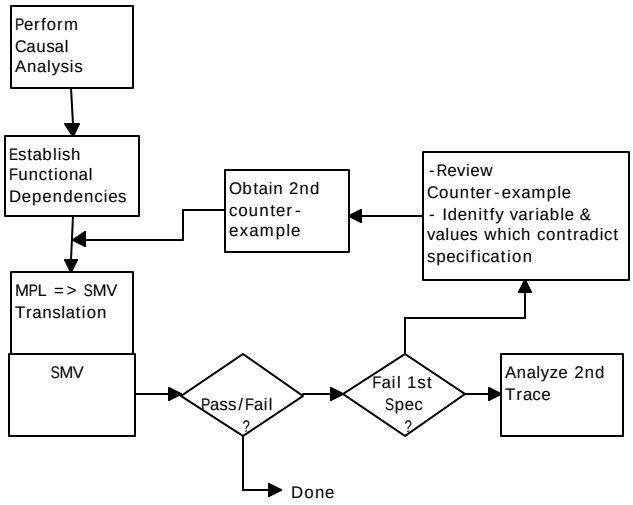

Figure 10. Variable Dependency Methodology

Conclusion

Expanded exploration of our Solar System will require more sophisticated autonomous assets to be developed and deployed. The Model based Autonomous system is a primary technology solution to this problem. A critical factor in the successful operations of these systems is to ensure that the models behave correctly, but due to their complexity conventional informal testing techniques will need to be augmented by more formal approaches. Toward this goal this paper presented some preliminary observations and results in the use of model-checking techniques in the evaluation of a certain kind of model (i.e. declarative Livingstone models). As with any kind of evaluation, the evaluation is only as good as it's measure, therefore the generation of formal specifications is of fundamental importance. The formal methods practitioner is Challenged to craft a set of correct and concise temporal specifications which capture behaviors that can query the model for safety and liveness properties This task is made easier if many of these specifications can be captured as specifications patterns which can be re-used for in evaluation of different modeling applications.

References

[1] R. Zubrin and R. Wagner. The Case for Mars: The plan to settle the Red Planet and why we must. The Free Press, 1996.

[2] Property Specification for Finite-State Verification, Dwyer M.B., Avrunin G.S., Corbett J.C., http://www.cis.ksu.edu/santos/spec-patterns, 1999.

[3] Intelligent Systems Software for Human Mars Missions, Larson W.E., Goodrich C.H., 51st International Astronautical Congress, Rio de Janeiro, Brazil, 2000.

[4] Optimizing Symbolic Model Checking for Invariant-Rich Models (abstract). B.Yang, R. Simmons, R. Bryant, and D. O'Hallaron. In Proc. of International Conference on Computer-Aided Verification (CAV'99).

[5] Charles Pecheur, Reid Simmons. From Livingstone to SMV: Formal Verification for Autonomous Spacecrafts. Proceedings of First Goddard Workshop on Formal Approaches to Agent-Based Systems, NASA Goddard, April 5-7, 2000. To appear in Lecture Notes in Computer Science, Springer Verlag.