Formal Verification of a Descent Guidance Control Program of a Lunar Lander*

Hengjun Zhao[1,4], Mengfei Yang[2], Naijun Zhan[1,**], Bin Gu[3], Liang Zou[1,4], and Yao Chen[3]

[1] State Key Lab. of Computer Science, Institute of Software, CAS, Beijing, China
{zhaohj,znj,zoul}@ios.ac.cn
[2] Chinese Academy of Space Technology, Beijing, China
[3] Beijing Institute of Control Engineering, Beijing, China
[4] University of Chinese Academy of Sciences, Beijing, China

Abstract. We report on our recent experience in applying formal methods to the verification of a descent guidance control program of a lunar lander. The powered descent process of the lander gives a specific hybrid system (HS), i.e. a sampled-data control system composed of the physical plant and the embedded control program. Due to its high complexity, verification of such a system is very hard. In the paper, we show how this problem can be solved by several different techniques including simulation, bounded model checking (BMC) and theorem proving, using the tools Simulink/Stateflow, iSAT-ODE and Flow*, and HHL Prover, respectively. In particular, for the theorem-proving approach to work, we study the invariant generation problem for HSs with general elementary functions. As a preliminary attempt, we perform verification by focusing on one of the 6 phases, i.e. the slow descent phase, of the powered descent process. Through such verification, trustworthiness of the lunar lander's control program is enhanced.

Keywords: Lunar lander, formal verification, hybrid systems, reachable set, invariant.

1 Introduction

Recently, China just launched a lunar lander to achieve its first soft-landing and roving exploration on the moon. After launching, the lander first entered an Earth-Moon transfer orbit, then a 100 kilometers (km)-high circular lunar orbit, and then a 15km $\times$ 100km elliptic lunar orbit. At perilune of the elliptic orbit, the lander's variable thruster was fired to begin the powered descent process, which can be divided into 6 phases. As shown in Figure 1, the terminal phase of powered descent is the slow descent phase, which should normally end several meters above the landing site, followed by a free fall to the lunar surface. One of the reasons to shut down the thruster before touchdown is to reduce the amount of stirred up dust that can damage onboard instruments.

* This work has been partly supported by National Basic Research Program and "863 Plan" of China (2014CB340700 and 2011AA010105) and NSFC 91118007.
** Corresponding author.

C. Jones, P. Pihlajasaari, and J. Sun (Eds.): FM 2014, LNCS 8442, pp. 733–748, 2014.

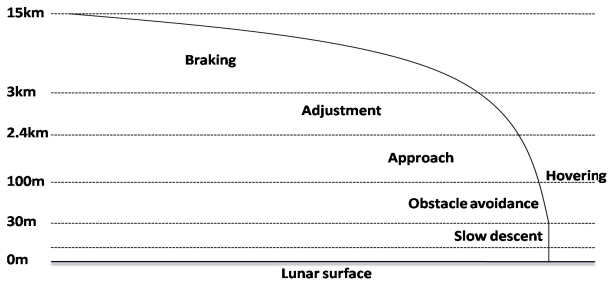

Fig. 1. The powered descent process of the lunar lander

Powered descent is the most challenging task of the lunar lander mission because it is fully autonomous. Due to communication delay, it is impossible for stations on earth to track the rapidly moving lander, and remote control commands from earth cannot take effect immediately. The lander must rely on its own guidance, navigation and control (GNC) system to, in real time, acquire its current state, calculate control commands, and use the commands to adjust its attitude and engine thrust. Therefore the reliable functionality of the GNC system is the key to the success of soft-landing. The motivation of this work is to enhance the trustworthiness of the guidance control program used in the powered descent process by application of formal methods.

The general framework of our approach is as follows. Firstly, with the help of engineers participating in the lunar lander project, we build a closed loop model consisting of the lander's physical dynamics and the program controlling the lander; the program model is excerpted from the real C-code program by keeping the critical control flows and numerical computations while abstracting away non-essential information. In addition, properties about the closed-loop system that are the engineers' main concerns are proposed. Then these properties are analyzed or formally verified.

The closed-loop model has the following prominent features: 1) the physical dynamics is modelled by ordinary differential equations (ODEs) with general elementary functions (rational, trigonometric, exponential functions etc.); 2) the program has complex branching conditions and numerical computations; 3) the physical process is frequently interrupted by control inputs from the program; 4) the system suffers from various uncertainties. Verification of such a system is beyond the capacity of many existing verification tools. Our solutions are as follows: 1) we first build a Simulink/Stateflow model of the closed-loop system and analyze its behaviour by simulation; 2) we then perform bounded model checking (BMC) of the system w.r.t. proposed properties using the tools iSAT-ODE [5] and Flow* [3]; 3) thirdly, with the tool Sim2HCSP [16,15] we automatically translate the Simulink/Stateflow graphical model to a formal model given by HCSP [8,13], a formal modelling language of HSs, and then perform unbounded safety verification of the system using HHL Prover [14], a theorem prover for HSs, by extending our previous work on invariant generation for polynomial HSs [11] to HSs with general elementary functions.

We have tried to show the effectiveness of three different formal verification tools because each of them has its own strength (and weakness) and cannot be replaced by

the others: iSAT-ODE can deal with very complex logical structures and data types (real, integer, Boolean) of programs, and provides flexible control of unwinding depth for BMC; Flow* can cope with large initial sets and perturbation of system models; HHL Prover can save lots of efforts in safety verification, especially when considering safety in a large or even unbounded time interval.

In this paper, we mainly focus on applying the above framework to the slow descent phase. Verification is performed before the real program is deployed on the lunar lander's computer, and thus strengthens our trust in the dependability of the program.

1.1 Related Work

Verification of full feedback system combining the physical plant with the control program has been advocated by Cousot [4] and Goubault et al. [7]. There are some recent work in this trend which resembles our work in this paper. In [2], Bouissou et al. presented a static analyzer named HybridFluctuat to analyze hybrid systems encompassing embedded software and continuous environment; subdivision is needed for HybridFluctuat to deal with large initial sets. In [12], Majumdar et al. also presented a static analyzer CLSE for closed-loop control systems, using concolic execution and SMT solving techniques; CLSE only handles linear continuous dynamics. In [1], Saha et al. verified stability of control software implementations; their approach requires expertise on analysis of mathematical models in control theory using such tools as Lyapunov functions.

There are some recent work on application of formal methods in the aerospace industry. For example, in [9] Johnson et al. proved satellite rendezvous and conjunction avoidance by computing the reachable sets of nonlinear hybrid systems; in [6] Katoen et al. reported on their usage of formal modelling and analysis techniques in the software development for a European satellite.

Paper Organization. The rest of this paper is organized as follows. In Section 2, we give a detailed description of the slow descent phase and the related verification problems. In Section 3, we build the Simulink/Stateflow model and then analyze the system's behaviour by simulation. In Section 4 we formally verify the proposed properties by BMC and theorem proving. The paper is concluded by Section 5.

2 Description of the Verification Problem

Overview of the Slow Descent Phase. The slow descent phase begins at an altitude (relative to lunar surface) of approximately 30m and terminates when the engine shutdown signal is received. The task of this phase is to ensure that the lander descends slowly and smoothly to the lunar surface, by nulling the horizontal velocity, maintaining a prescribed uniform vertical velocity, and keeping the lander at an upright position. The descent trajectory is nearly vertical w.r.t. the lunar surface (see Figure 2).

The operational principle of the GNC system for the slow descent phase (and any other phases) can be illustrated by Figure 3. The closed loop system is composed of the lander's dynamics and the guidance program for the present phase. The guidance program is executed periodically with a fixed sampling period. At each sampling point,

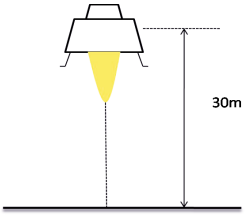

Fig. 2. The slow descent phase

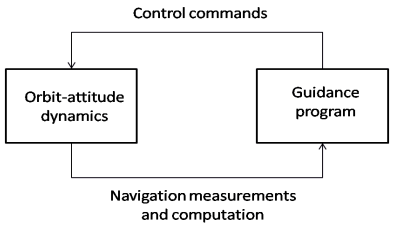

Fig. 3. A simplified configuration of GNC

the current state of the lander is measured by IMU (inertial measurement unit) or various sensors. Processed measurements are then input into the guidance program, which outputs control commands, e.g. the magnitude and direction of thrust, to be imposed on the lander's dynamics in the following sampling cycle.

We next give a mathematical description of the lander's dynamics as well as the guidance program of the slow descent phase. For the purpose of showing the technical feasibility and effectiveness of formal methods in the verification of aerospace guidance programs, we neglect the attitude control as well as the orbit control in the horizontal plane, resulting in a one-dimensional (the vertical direction) orbit dynamics.

Dynamics. Let the upward direction be the positive direction of the one-dimensional axis. Then the lander's dynamics is given by

$$\begin{cases} \dot{r} &= v \\ \dot{v} &= \frac{F_c}{m} - gM \\ \dot{m} &= -\frac{F_c}{Isp_1} \\ \dot{F}_c &= 0 \\ F_c &\in [1500, 3000] \end{cases} \quad \text{and} \quad \begin{cases} \dot{r} &= v \\ \dot{v} &= \frac{F_c}{m} - gM \\ \dot{m} &= -\frac{F_c}{Isp_2} \\ \dot{F}_c &= 0 \\ F_c &\in (3000, 5000] \end{cases}, \text{ where} \tag{1}$$

- r, v and m denote the altitude (relative to lunar surface), vertical velocity and mass of the lunar lander, respectively;
- F_c is the thrust imposed on the lander, which is a constant in each sampling period;
- gM is the magnitude of the gravitational acceleration on the moon, which varies with height r but is taken to be the constant 1.622m/s^2 in this paper, since the change of height ($0 \leq r \leq 30\text{m}$) can be neglected compared to the radius of the moon;
- $Isp_1 = 2500\text{N·s/kg}$ and $Isp_2 = 2800\text{N·s/kg}$ are the two possible values that the *specific impulse*[1] of the lander's thrust engine can take, depending on whether the current F_c lies in $[1500, 3000]$ or $(3000, 5000]$, and thus the lander's dynamics comprises two different forms as shown in (1);
- note that the terms $\frac{F_c}{m}$ in (1) make the dynamics non-polynomial.

[1] Specific impulse is a physical quantity describing the efficiency of rocket engines. It equals the thrust produced per unit mass of propellant burned per second.

Guidance Program. The guidance program for the slow descent phase is executed once for every 0.128s. The control flow of the program, containing 4 main blocks, is demonstrated by the left part of Figure 4.

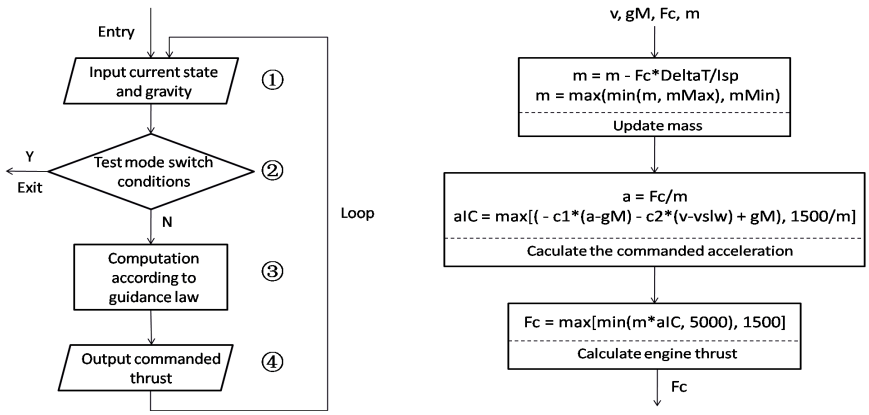

Fig. 4. The guidance program for the slow descent phase

The program first reads data given by navigation computation (block 1), and then decides whether to stay in the slow descent phase or switch to other phases by testing the following conditions (block 2):

(SW1) shutdown signal 1, which should normally be sent out by sensors at the height of 6m, is received, and the lander has stayed in slow descent phase for more than 10s;
(SW2) shutdown signal 2, which should normally be sent out by sensors at the height of 3m, is received, and the lander has stayed in slow descent phase for more than 10s;
(SW3) no shutdown signal is received and the lander has stayed in the slow descent phase for more than 20s.

If any of the above conditions is satisfied, then the GNC system switches from slow descent phase to no-control phase and a shutdown command is sent out to the thrust engine; otherwise the program will stay in the slow descent phase and do the guidance computation (block 3) as shown in the right part of Figure 4, where

- v and gM are the vertical velocity and gravitational acceleration from navigation measurements or computation; note that we have assumed gM to be a constant;
- F_c and m are the computed thrust and mass estimation at last sampling point; they can be read from memory;
- $DeltaT = 0.128$s is the sampling period;
- Isp is the specific impulse which can take two different values, i.e. 2500 or 2800, depending on the current value of F_c;

- $mMin = 1100$kg and $mMax =$3000kg are two constants used as the lower and upper bounds of mass estimation;
- $c_1 = 0.01$ and $c_2 = 0.6$ are two control coefficients in the guidance law;
- $vslw = -2$m/s is the target descent velocity of the slow descent phase;
- the output F_c (block 4) will be used to adjust engine thrust for the following sampling cycle; it can be deduced from the program that the commanded thrust F_c always lies in the range $[1500, 5000]$.

Verification Objectives. Together with the engineers participating in the lunar lander project, we propose the following properties to be verified regarding the closed-loop system of the slow descent phase and the subsequent free fall phase.

Firstly, suppose the lunar lander enters the slow descent phase at $r = 30$m with $v = -2$m/s, $m = 1250$kg and $F_c = 2027.5$N. Then

(P1) **Safety 1:** $|v - vslw| \leq \varepsilon$ during the slow descent phase and before touchdown[2], where $\varepsilon = 0.05$m/s is the tolerance of fluctuation of v around the target $vslw = -2$m/s;

(P2) **Safety 2:** $|v| < vMax$ at the time of touchdown, where $vMax = 5$m/s is the upper bound of $|v|$ to avoid the lander's crash when contacting the lunar surface;

(P3) **Reachability:** one of the switching conditions (SW1)-(SW3) will finally be satisfied so that the system will exit the slow descent phase.

Furthermore, by taking into account such factors as uncertainty of initial state, disturbance of dynamics, sensor errors, floating-point calculation errors etc., we give

(P4) **Stability and Robustness:** (P2) and (P3) still holds, and an analogous of (P1) is that v will be steered towards $vslw = -2$m/s after some time.

3 Simulation

We first build a Simulink/Stateflow model[3] of the closed-loop system for the slow descent phase. Then based on the model we analyze the system's behaviour by simulation.

According to Section 2, the physical dynamics is specified by (1), which is modelled by the Simulink diagram shown in Figure 5.

In Figure 5, several blocks contain parameters that are not displayed:

- the threshold of lsp is 3000, which means lsp outputs 2800 when F_c is greater than 3000, and 2500 otherwise;
- the initial values of m, v and r ($m = 1250$kg, $r = 30$m, $v = -2$m/s) are specified as initial values of blocks m1, v1 and r respectively.

[2] Note that if no shutdown signal is received, there exists possibility that the lander stays in the slow descent phase after landing.

[3] All the details of simulation and verification in this paper can be found at `http://lcs.ios.ac.cn/~zoul/casestudies/hcs.rar`

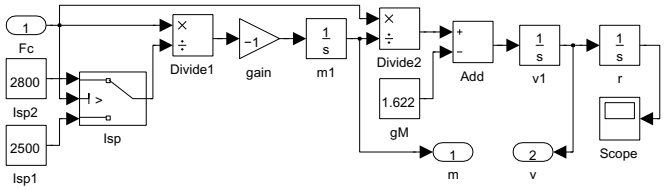

Fig. 5. The Simulink diagram of the dynamics for the slow descent phase

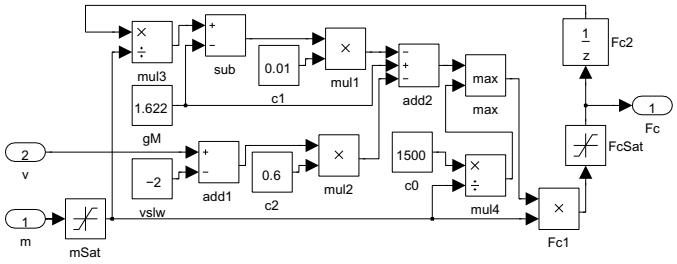

Fig. 6. The Simulink diagram of the guidance program for the slow descent phase

As specified in Figure 4, The guidance program includes three parts: updating mass m, calculating acceleration *aIC*, and calculating thrust F_c. The Simulink diagram for the guidance program is shown in Figure 6, in which the sample time of all blocks are fixed as 0.128s, i.e. the period of the guidance program. In Figure 6, blocks m and mSat are used to update mass m, blocks Fc1 and FcSat are used to calculate thrust F_c, and the rest are used to calculate acceleration *aIC*. Blocks mSat and FcSat are saturation blocks from Simulink library which limit input signals to the upper and lower bounds of m and F_c respectively.

The simulation result is shown in Figure 7. The left part shows that the velocity of the lander is between -2 and -1.9999, which corresponds to (P1); the right part shows that if shutdown signal 1 is sent out at 6m and is successfully received by the lander, then (SW1) will be satisfied at time 12.032s, which corresponds to (P3).

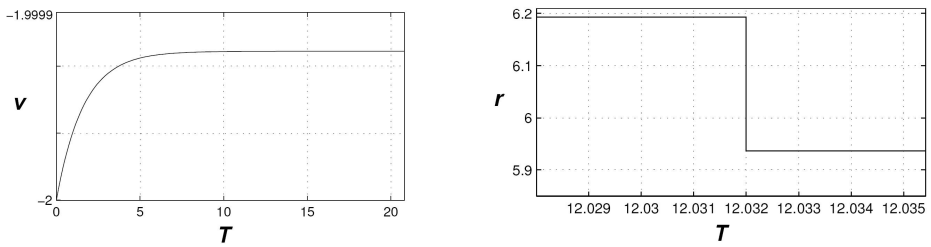

Fig. 7. The simulation result

4 Verification

In this section, we formally verify the properties (P1)-(P4) proposed in Section 2. Firstly, using iSAT-ODE [5] we build an exact model of the dynamics (1) and the guidance program shown in Figure 4, and then verify (P1)-(P3) by BMC with an initial set of a single point as specified in Section 2. Secondly, we take various uncertainties into account and verify (P4) by computing the system's (bounded) reachable set using Flow* [3]; to simplify the modelling in Flow*, we have made reasonable simplifications of the guidance control program. Finally, we show how theorem proving can be an alternative to BMC by performing unbounded verification of (P1) using HHL Prover [14].

4.1 Verification by Bounded Model Checking

Bounded model checking (BMC) is a verification technique that, for a given state transition system and an initial set of states, answers whether an unsafe state can be reached by unwinding the transition system to a depth of k. BMC is suitable for the verification of sampled-data systems because the periodic sampling and control naturally induce state transitions with a fixed time step. The tool iSAT-ODE [5] is a bounded model checker that handles nonlinear arithmetic and nonlinear differential equations.

Modelling in iSAT-ODE. Thanks to the support of Boolean, integer and real data types, as well as such functions as max, min, abs etc. in iSAT-ODE, modelling of the closed loop system for the slow descent phase is straightforward. We first define two Boolean variables mode_slow and mode_free to represent the slow descent phase and the free fall phase respectively. We further require that at any time one and only one of mode_slow and mode_free is TRUE. Each sampling cycle induces two kinds of state transitions, i.e. the continuous and discrete transition, which are distinguished by a Boolean variable jump. For example, the following texts:

```
mode_slow and !jump -> (d.r / d.time = v);
mode_slow and !jump -> (d.v / d.time = Fc/m - gM);
mode_slow and !jump -> (d.m / d.time = -Fc/Isp1);
mode_slow and !jump -> (d.Fc / d.time = 0);
```

can be used to define a continuous transition under dynamics (1) with specific impulse Isp_1, where !jump denotes the negation of jump and Isp1 is the constant 2500. Similarly, an update of Fc by a discrete computation has the following form:

```
mode_slow and jump -> Fc' = (***);
```

where Fc′ denotes the value of Fc after transition, and (***) is not the language of iSAT-ODE but the abbreviation of the omitted updating assignments of Fc. The duration of each transition is represented by a real variable delta_time, which equals the sampling period in the continuous case and 0 in the discrete case.

The critical part is to model the conditions of switching from the slow descent phase to the free fall phase, i.e. (SW1)-(SW3). Based on whether the shutdown signal 1 or 2 is received, we build three different models with the conditions (SW1), (SW2) and (SW3)

respectively. For example, if shutdown signal 1 is sent out exactly at a height of 6m and is successfully received, then (SW1) will be used in the model and it is encoded as:

```
mode_slow and jump -> (mode_free' <-> r <= 6 and time > 10);
```

where time denotes the total time elapsed in the slow descent phase. The properties (P1)-(P3) will be verified on all three models.

In our model we assume the lander's velocity becomes 0 immediately upon touchdown and stays at 0 afterwards.

Verification in iSAT-ODE. Bounded model checking in iSAT-ODE can be done by specifying a target set (formula) whose reachability is to be checked, as well as the minimal and maximal unwinding depth of the state transition system for constructing BMC formulas. For the model of the slow descent phase, since each sampling cycle corresponds to two transition steps[4], if the system's reachable set in n sampling cycles is going to be checked against the target formula, then the minimal and maximal depth should be specified as 0 and $2n$ (or $2n-1$) respectively.

We first try to verify (P3) by setting the target formula to !mode_slow. If the current phase is the slow descent phase, then the result unsatisfiable will be returned; otherwise satisfiable will be expected. In our model mode_slow is initially set to TRUE. We check for each $k \geq 0$ to find the first k that gives the satisfiable answer, which means phase switching happens at k. However, according to our experience, at the unwinding depth where the target formula becomes satisfiable, iSAT-ODE will run for a long time until memory is exhausted without giving an answer. In practice, when this phenomenon is observed, it is very likely that the target formula is satisfiable at the current depth, so if we check against the negation of the target formula, then an unsatisfiable answer will be expected. A good *rule of thumb* is that with iSAT-ODE, it's better to check against a target that is indeed *unsatisfiable*. In this way, we have shown that:

- if shutdown signal 1 is received, phase switching happens at $k=188$, i.e. the end of the 94^{th} sampling cycle, or equivalently the time 12.032s (consistent with Figure 7);
- if shutdown signal 1 is not received and shutdown signal 2 is received, phase switching happens at $k = 212$;
- if no shutdown signal is received, phase switching happens at $k = 314$.

We then try to verify (P1) by setting the target formula to the negation of (P1):

$$\mathsf{r} > 0 \text{ and } !(\mathsf{v} >= -2.05 \text{ and } \mathsf{v} <= -1.95) \ .$$

Since we are only considering the lander's velocity in the slow descent phase, this target is checked for depth $0 \leq k \leq 187, 0 \leq k \leq 211, 0 \leq k \leq 313$ respectively for three different models. In this way we have successfully verified (P1).

We finally try to verify (P2) by first getting an estimation of the ranges of v and r at the time phase switching happens, i.e. $k = 188, 212, 314$ for the three different models respectively. To this end, we have to guess a possible range of v or r and then check

[4] In our model, the k-th transition is a continuous transition if k is an odd number, and a discrete transition if k is an even number.

against the negation of the estimated range in iSAT-ODE. It's a process of trial and error. Bipartition of intervals can be applied. Eventually, we get

- if shutdown signal 1 is received, then $r \in [5.9, 6.0]$ (consistent with Figure 7) and $v \in [-2.05, -1.95]$ when phase switching happens;
- if shutdown signal 1 is not received and shutdown signal 2 is received, then $r \in [2.8, 2.9]$ and $v \in [-2.05, -1.95]$ when phase switching happens;
- if no shutdown signal is received, then $r = 0, v = 0$ when phase switching happens, and by the verified (P1) we have $v \in [-2.05, -1.95]$ whenever $r > 0$.

Since slow descent phase is followed by free fall, using the range estimations of v and r and the dynamics of free fall, we show that in all three cases $|v| < 5$m/s upon touchdown.

The cost of the above verification, on the platform with Intel Q9400 2.66GHz CPU running a Debian virtual machine with 3GB memory allocated, is shown in Table 1.

Table 1. Time and memory cost of the verification in iSAT-ODE

	Model with (SW1)	Model with (SW2)	Model with (SW3)
(P1)	2min46sec, 477MB	3min46sec, 594MB	14min3sec, 1.8GB
(P2)	24sec, 304MB	31sec, 378MB	50sec, 602MB
(P3)	1min22sec, 290MB	2min1sec, 350MB	2min7sec, 62MB

4.2 Verification with Uncertainties

We have shown how properties (P1)-(P3) can be verified using iSAT-ODE, by assuming the initial state to be a single point, and the continuous dynamics, sampling time points, navigation and guidance computations etc. are all exact. However, in practice such ideal models do not exist because disturbances and noises are unavoidable in the physical world. Therefore it is meaningful to analyze the performance of the lander's GNC system by taking into account various uncertainties. To this end, we next verify (P4) proposed in Section 2 using Flow* [3], a tool for computing over-approximations of the reachable sets of continuous dynamical and hybrid systems. The prominent features of Flow* include the handling of non-polynomial ODEs, ODEs with uncertainties, reset functions with uncertainties, and so on, which all facilitate our modelling here.

Modelling as Hybrid Automata. Basically, in Flow* a hybrid system is modelled as a hybrid automaton (HA). If we build a complete model in Flow* for the slow descent phase using the program in Figure 4, then the max and min functions would make the transition relation in the resulting HA very complex. To simplify the modelling and verification in Flow*, we make the following assumption which will be justified later:

(A1) throughout the execution of the guidance program, the value of m lies in the range $[mMin, mMax]$, and the value of

$$F_c' \mathrel{\widehat{=}} -c_1 \cdot (F_c - m \cdot gM) - c_2 \cdot (v - vslw) \cdot m + m \cdot gM \tag{2}$$

lies in the range $[1500, 5000]$.

Under assumption (A1) all the max and min functions can be simplified and it is easy to check that the computation of thrust in the guidance program is equivalent to $F_c := F'_c$.

As in iSAT-ODE, we can also build three different models in Flow* with the switching conditions (SW1), (SW2) and (SW3) respectively. In the following, we only discuss the model with (SW1) as illustrated by Figure 8, and the verification work done with it.

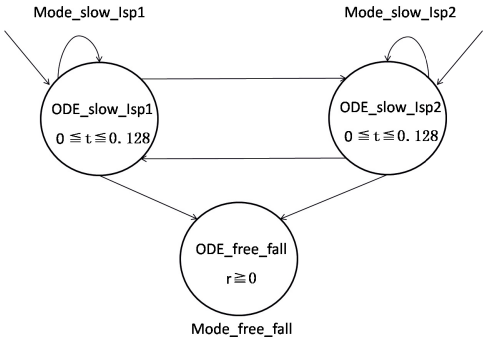

Fig. 8. The HA model of the slow descent phase

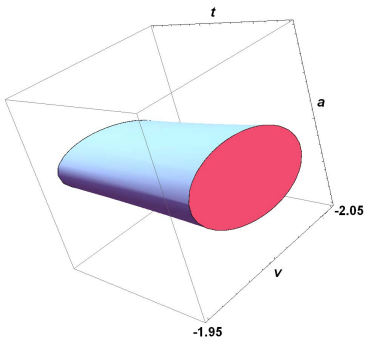

Fig. 9. The invariant for HHL Prover

For Figure 8, we give the following explanations:

- the three modes represent the slow descent phase with specific impulse 2500, 2800, and the free fall phase, respectively; the mode domains are shown in the picture; the continuous dynamics are the two in (1) and the standard dynamics of free fall on the lunar surface; all dynamics are augmented with the flow rate of time $\dot{t} = 1$ and $\dot{T} = 1$, where t represents the local elapsed time in the current sampling cycle and T denotes the total elapsed time since the beginning;
- all the discrete jumps take place at $t = 0.128$ and t is reset to 0 for every jump;
- the jumps from Mode_slow_lsp1 or Mode_slow_lsp2 to Mode_free_fall depend on the truth value of (SW1), i.e. $r \leq 6 \wedge T > 10$;
- the jumps from Mode_slow_lsp1 and Mode_slow_lsp2 to themselves, or the jumps between them, depend on (SW1) and the comparison of F'_c (defined in (2)) to 3000; the value of F_c is updated to F'_c for every such jump.

Introducing Uncertainties. We next modify the model in Figure 8 by introducing into it various kinds of uncertainties according to different origins:

- The initial states are chosen to be intervals, e.g. $v \in [-2.5, -1.5]$, $r \in [29.5, 30.5]$, $m \in [1245, 1255]$ $F_c \in [2020, 2035]$,[5] and so on.
- Add interval disturbances to dynamics (1) and the dynamics of free fall. The causes of such uncertainties could be: the direction of F_c may deviate from the vertical

[5] Thus the initial mode should be the slow descent phase with specific impulse 2500.

direction; the specific impulse may not be exactly 2500 or 2800; the engine may not be able to keep a constant thrust in one cycle; the acceleration of gravity is not the constant 1.622 but changes with height; and so on. For example, we have $\dot{m} = -\frac{F_c}{2500} + [-0.1, 0.1]$ if the specific impulse has a ± 300 perturbation around 2500.

- In the guidance program, the value of F_c is stored in memory so it is not changed between two sampling points, while the actual thrust imposed on the lander may not be constant in one cycle; besides, due to uncertainty of specific impulse, the estimated value of m by the program using fixed specific impulse values 2500 or 2800 may deviate from the real mass value. Therefore we introduce two new variables m_p and F_p, whose time derivatives are zero, to distinguish between program variables and continuous state variables.
- The measurement of time in the computer system may not be precise, and thus the length of one sampling cycle may vary in the range, say $[0.127, 0.129]$. This should be reflected in the domains and transition guards of the hybrid automaton.
- The measured height may suffer from sensor errors, say ± 0.1m, and thus the shutdown signal may be sent out at a height of 6 ± 0.1m. Therefore we revise the phase switching condition by taking into consideration such imprecision.
- The measured velocity may also suffer from sensor errors, say ± 0.1m/s. Since the value of m (or m_p) is greater than 1000kg, by (2), this may cause a fluctuation of nearly 100N of the commanded thrust. Therefore we revise (2) by

$$F_p' \widehat{=} -c_1 \cdot (F_p - m_p \cdot gM) - c_2 \cdot (v - vslw) \cdot m_p + m_p \cdot gM + [-100, 100] \ . \quad (3)$$

  In the computation of F_p, there may also exist floating point errors, which we claim can be absorbed by the large interval $[-100, 100]$.

Computation Results. We compute the reachable set of the above described model with a time bound of 25s and an unwinding depth (the maximal number of allowed jumps) of 200. The computation costs 19 minutes and 769MB memory on the platform with Intel Q9400 2.66GHz CPU and 4GB RAM running Ubuntu Linux. The relations between v, r, F_p, m_p and T and shown in Figure 10 which can be explained as follows:

- The ranges of T in all pictures are within $[0, 18]$. Neither the time bound 25 or the unwinding depth 200 is reached during the flowpipe computation, which implies that the result covers all the reachable states of the hybrid automaton in Figure 8.
- The top left picture shows the relation between v and T. Since the initial range of v is $[-2.5, -1.5]$, property (P1) does not make sense. However, we can still conclude that the system has a good asymptotic property, that is, the value of v converges to a stable interval, approximately $[-2.25, -1.75]$ after some time. Besides, it can be seen from the picture that v is always above the level -5m/s; actually property (P2) can be formally verified with the support of safety checking in Flow*. Furthermore, from the sharp decrease in v we can infer[6] property (P3), that is, starting from any initial state the system will finally switches to the free fall phase.
- The top right picture shows the relation between r and T.

[6] A formal proof can be obtained by looking into the mode information of computed flowpipes.

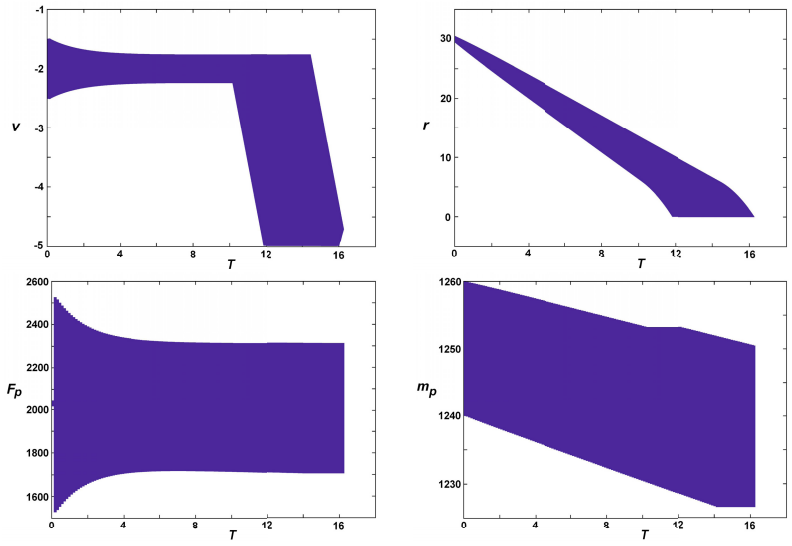

Fig. 10. Reachable sets given by Flow*

- The bottom left picture shows the relation between F_p and T. Although the initial range of F_p is narrow (defined to be $[2015, 2040]$), by (3), the variation of v in $[-2.5, -1.5]$ would cause fluctuation of F_p by several hundreds. Nevertheless, the picture shows that the range of F_p also stabilizes after some time. Besides, F_p always lies in $[1500, 2600]$, which justifies our assumption (A1).
- The bottom right picture shows the relation between m_p and T. The initial value of m_p is defined to be $[1240, 1260]$. It can be seen that m_p always lies in $[1225, 1260]$, which also justifies the assumption (A1).

4.3 Verification by Theorem Proving

One disadvantage of verification by BMC is that it cannot verify a safety property at all time. Even if we only care about properties within a bounded time interval, BMC may not work with very large intervals, since more resources are required for larger unwinding depths, a fact confirmed by Table 1. We show that theorem proving can be a good alternative to BMC for safety verification, by verifying (P1) using HHL Prover [14].

Transformation to HCSP. We first build a Simulink/Stateflow model similar to the one in Section 3 with simplified thrust computations according to assumption (A1), which has been justified by the verification results of Flow*. We then automatically translate the model into a formal model given by HCSP using the tool Sim2HCSP [16]. Basically the transformed HCSP process is as follows:

```
definition P :: proc where
"P == PC_Init;PD_Init;t:=0;(PC_Diff;t:=0;PD_Rep)*"
```

In process P, PC_Init and PD_Init are initialization procedures for the continuous dynamics and the guidance program respectively; PC_Diff models the continuous dynamics given by (1) within a period of 0.128s; PD_Rep calculates thrust F_c according to (2) for the next sampling cycle; variable t denotes the elapsed time in each sampling cycle.

Verification in HHL Prover. In order to verify property (P1), we give the following proof goal in HHL Prover:

```
lemma goal : "{True} P {safeProp; (l=0 | (high safeProp))}"
```

where safeProp stands for $|v - vslw| \leq \varepsilon$. The parts True and safeProp; specify the pre- and post-conditions of P respectively. The part (l=0 | (high safeProp)) specifies a duration property, where l=0 means the duration is 0, and high is just a syntax construct.

After applying proof rules in HHL Prover with the above proof goal, the following three lemmas remain unresolved:

```
lemma constraint1: "(t<=0.128) & Inv |- safeProp"
lemma constraint2: "(v=-2) & (m=1250) & (Fc=2027.5)
  & (t=0) |- Inv"
lemma constraint3: "(t= 0.128) & Inv
  |- substF([(t,0)], substF([(Fc,
     -0.01*(Fc-1.622*m) - 0.6*(v+2)*m + 1.622*m)],Inv))"
```

In a more readable way, the three lemmas impose the following constraints:

(C1) $0 \leq t \leq 0.218 \wedge Inv \longrightarrow |v - vslw| \leq \varepsilon$;
(C2) $v = -2 \wedge m = 1250 \wedge F_c = 2027.5 \wedge t = 0 \longrightarrow Inv$;
(C3) $t = 0.128 \wedge Inv \longrightarrow Inv(0 \leftarrow t; F_c' \leftarrow F_c)$, with F_c' defined in (2);
(C4) Inv is the invariant of both constrained dynamical systems

$$\langle ODE_1; 0 \leq t \leq 0.128 \wedge F_c \leq 3000 \rangle \text{ and } \langle ODE_2; 0 \leq t \leq 0.128 \wedge F_c > 3000 \rangle \ ,$$

where ODE_1 and ODE_2 are the two dynamics defined in (1).

We will address the problem of invariant generation in the subsequent subsection.

Invariant Generation. Invariant generation for polynomial continuous/hybrid systems has been studied a lot [11]. To deal with systems with non-polynomial dynamics, we propose a method based on variable transformation. For this case study, we replace the non-polynomial terms $\frac{F_c}{m}$ in ODE_1 and ODE_2 by a new variable a. Then by simple computation of derivatives we get two transformed polynomial dynamics:

$$ODE_1' \mathrel{\widehat{=}} \begin{cases} \dot{r} = v \\ \dot{v} = a - 1.622 \\ \dot{a} = \frac{a^2}{2500} \end{cases} \text{ and } ODE_2' \mathrel{\widehat{=}} \begin{cases} \dot{r} = v \\ \dot{v} = a - 1.622 \\ \dot{a} = \frac{a^2}{2800} \end{cases} . \tag{4}$$

Furthermore, it is not difficult to see that the update of F_c as in (2) can be accordingly transformed to the update of a given by

$$a' \mathrel{\widehat{=}} -c_1 \cdot (a - gM) - c_2 \cdot (v - vslw) + gM \ . \tag{5}$$

As a result, if we assume Inv to be a formula over variables v, a, t, then (C2)-(C4) can be transformed to:

(C2') $v = -2 \wedge a = 1.622 \wedge t = 0 \longrightarrow Inv$;
(C3') $t = 0.128 \wedge Inv \longrightarrow Inv(0 \leftarrow t; a' \leftarrow a)$, with a' defined in (5);
(C4') Inv is the invariant of both constrained dynamical systems $\langle ODE'_1; 0 \leq t \leq 0.128\rangle$ and $\langle ODE'_2; 0 \leq t \leq 0.128\rangle$[7] with ODE'_1 and ODE'_2 defined in (4).

Note that the constraints (C1) and (C2')-(C4') are all polynomial. Then the invariant Inv can be synthesized using the SOS (sum-of-squares) relaxation approach in the study of polynomial hybrid systems [10]. With the Matlab-based tool YALMIP and SDPT-3, an invariant $p(v, a, t) \leq 0$ as depicted by Figure 9 is generated. Furthermore, to avoid the errors of numerical computation in Matlab, we perform post-verification using the computer algebra tool RAGlib[8] to show that the synthesized $p(v, a, t) \leq 0$ is indeed an invariant. Thus we have successfully completed the proof of property (P1) by theorem proving. On the platform with Intel Q9400 2.66GHz CPU and 4GB RAM running Windows XP, the synthesis costs 2s and 5MB memory, while post-verification costs 10 minutes and 70MB memory.

5 Conclusions

We studied a short piece of program used for the guidance and control in the terminal slow descent phase of a lunar lander. With the assistance of engineers from the lunar lander project, a closed-loop system linking the program and the lander's dynamics was mathematically described, and safety-critical properties about the system were proposed. These properties were all successfully verified by using or extending several existing formal verification techniques that can handle continuous-discrete interactions, general nonlinear differential equations and uncertainties. The dependability of the lunar lander's guidance control program was enhanced through such verification.

The preliminary results in this paper show good prospect of closed-loop verification of embedded software in sampled-data control systems. For future work, we will first try to perform a thorough verification of the lunar lander system; we also plan to investigate more effective invariant generation or flowpipe computation methods for general nonlinear ODEs. An ambitious goal is to develop a tool that can be used by engineers.

Acknowledgements. We are indebted to Prof. Chaochen Zhou, Dr. Shuling Wang, Dr. Yanxia Qi and Dr. Zheng Wang for the fruitful discussions with them on this work. The availability of iSAT-ODE is by courtesy of Prof. Martin Fränzle and Mr. Andreas Eggers. We thank Dr. Xin Chen for his instructions on the use of Flow*, and thank Prof. Mohab Safey El Din for the use of RAGlib. We also thank the anonymous referees for their valuable comments on the earlier draft.

[7] We have abstracted away the domain constraints on F_c.
[8] `http://www-polsys.lip6.fr/~safey/RAGLib/`

References

1. Anta, A., Majumdar, R., Saha, I., Tabuada, P.: Automatic verification of control system implementations. In: EMSOFT 2010, pp. 9–18. ACM, New York (2010)
2. Bouissou, O., Goubault, E., Putot, S., Tekkal, K., Vedrine, F.: HybridFluctuat: A static analyzer of numerical programs within a continuous environment. In: Bouajjani, A., Maler, O. (eds.) CAV 2009. LNCS, vol. 5643, pp. 620–626. Springer, Heidelberg (2009)
3. Chen, X., Ábrahám, E., Sankaranarayanan, S.: Flow*: An analyzer for non-linear hybrid systems. In: Sharygina, N., Veith, H. (eds.) CAV 2013. LNCS, vol. 8044, pp. 258–263. Springer, Heidelberg (2013)
4. Cousot, P.: Integrating physical systems in the static analysis of embedded control software. In: Yi, K. (ed.) APLAS 2005. LNCS, vol. 3780, pp. 135–138. Springer, Heidelberg (2005)
5. Eggers, A., Ramdani, N., Nedialkov, N., Fränzle, M.: Improving the SAT modulo ODE approach to hybrid systems analysis by combining different enclosure methods. In: Software & Systems Modeling, pp. 1–28 (2012)
6. Esteve, M.A., Katoen, J.P., Nguyen, V.Y., Postma, B., Yushtein, Y.: Formal correctness, safety, dependability, and performance analysis of a satellite. In: ICSE 2012, pp. 1022–1031. IEEE Press (2012)
7. Goubault, E., Martel, M., Putot, S.: Some future challenges in the validation of control systems. In: ERTS 2006 (2006)
8. He, J.: From CSP to hybrid systems. In: A Classical Mind: Essays in Honour of C. A. R. Hoare, pp. 171–189. Prentice Hall International (UK) Ltd, Hertfordshire (1994)
9. Johnson, T.T., Green, J., Mitra, S., Dudley, R., Erwin, R.S.: Satellite rendezvous and conjunction avoidance: Case studies in verification of nonlinear hybrid systems. In: Giannakopoulou, D., Méry, D. (eds.) FM 2012. LNCS, vol. 7436, pp. 252–266. Springer, Heidelberg (2012)
10. Kong, H., He, F., Song, X., Hung, W.N., Gu, M.: Exponential-condition-based barrier certificate generation for safety verification of hybrid systems. In: Sharygina, N., Veith, H. (eds.) CAV 2013. LNCS, vol. 8044, pp. 242–257. Springer, Heidelberg (2013)
11. Liu, J., Zhan, N., Zhao, H.: Computing semi-algebraic invariants for polynomial dynamical systems. In: EMSOFT 2011, pp. 97–106. ACM, New York (2011)
12. Majumdar, R., Saha, I., Shashidhar, K.C., Wang, Z.: CLSE: Closed-loop symbolic execution. In: Goodloe, A.E., Person, S. (eds.) NFM 2012. LNCS, vol. 7226, pp. 356–370. Springer, Heidelberg (2012)
13. Zhou, C., Wang, J., Ravn, A.P.: A formal description of hybrid systems. In: Alur, R., Henzinger, T.A., Sontag, E.D. (eds.) HS 1995. LNCS, vol. 1066, pp. 511–530. Springer, Heidelberg (1996)
14. Zou, L., Lv, J., Wang, S., Zhan, N., Tang, T., Yuan, L., Liu, Y.: Verifying Chinese train control system under a combined scenario by theorem proving. In: Cohen, E., Rybalchenko, A. (eds.) VSTTE 2013. LNCS, vol. 8164, pp. 262–280. Springer, Heidelberg (2014)
15. Zou, L., Zhan, N., Wang, S., Fränzle, M.: Formal verification of Simulink/Stateflow diagrams. Tech. Rep. ISCAS-SKLCS-13-07, State Key Lab. of Comput. Sci., Institute of Software, CAS (2013)
16. Zou, L., Zhan, N., Wang, S., Fränzle, M., Qin, S.: Verifying Simulink diagrams via a Hybrid Hoare Logic prover. In: EMSOFT 2013. IEEE Press (2013)