The Returning Rocket: Friend or Foe?

15–424: Foundations of Cyber-Physical Systems
Final Project
David Franklin, Philip Massey

Tuesday 9th December, 2014

Abstract

Modern space exploration has recently entered a golden age due to the increased accessibility of space travel. This increased accessibility is largely due to the lower costs of launching satellites and other spacecraft into orbit. One of the newest competitors in the spacecraft launching industry, Space Exploration Technologies, plans on developing reusable rockets to lower this cost even further. These reusable rockets rely heavily on automation for their guidance, navigation, and controller systems. Formal verification of these systems would allow for assurance of the safety of these multi-million dollar rockets during their return. We present our modeled systems representative of the physics behind this situation, and our developed controllers that correctly uphold the appropriate safety conditions for them. With our controllers formally validated and verified using differential dynamic logic powered KeYmaera proving tool, the space industry continues its path towards easily accessible spaceflight.

1 Introduction

The field of cyber-physical systems is a recently developed field exploring the collaboration between computational and physical elements. Cyber-physical systems have been increasingly prominent, and can easily be found in a diverse range of industries including aerospace, energy, manufacturing, transportation, and entertainment. Out of the many areas of this new field, the most exciting aspect is automation. Automation involves a computational controller completely deciding the fate of its physical entities, with little to no outside or human involvement. Automation is a particularly

interesting topic because the increased emphasis on safety, as ideally there is little or no manual oversight of these systems.

One of the newest form of automation in this industry is currently being researched and developed by Space Exploration Technologies (SpaceX). SpaceX's current and most pressing goal is to drastically reduce the cost of commercial space travel. Elon Musk, the CEO of SpaceX, believes that space travel can see the same success that air travel has seen over the past few decades due to the lower prices. SpaceX plans on reducing the price of space travel by tens of millions of dollars by manufacturing rockets that are reusable. By reusing rockets, the cost of a launch is dominated by the price of fuel and payload expenses instead of the price of a new rocket.

The Falcon 9 rocket currently being manufactured and tested at SpaceX are able to achieve partial re-usability [4]. After primary stage separation, the first stage rocket returns to the Earth while the second stage rocket (with the payload still attached) performs a second burn to move into the correct orbit. After this separation, the first stage rocket performs a vertical flip to realign the main engines underneath itself. It then reignites these engines, slowing its descent. During the descent, the first stage lowers its deployable landing legs to both help increase the surface area to increase air resistance to help slow it down, and to aid in stabilizing the first stage. Finally, the first stage lands on an autonomous spaceport drone ship in the middle of the ocean, where it is then ferried back to shore [2].

The process of the returning first stage rockets is critical to the advancement of reusability in rockets, and verifying the process would be an incredible step forward. Formal verification has proved its success in many industries, with companies such as Intel and Motorola investing in validation of their hardware [3]. However, cyber-physical systems such as a returning first stage rocket provide challenges outside of the verification of the controlling software. When the system includes controlling physical entities, the physics of the real world provide a challenge to ensuring safety.

Our submission for the Cyber-Physical Systems Verification and Validation Grand Prix is our analysis of the returning of the first stage rocket. This includes our model of the physics behind the descending first stage rocket. We constructed our system using differential dynamic logic, an extension of modal logic used to better reason about computer programs (which are used for control logic) as well as including the continuous motion of the differential equations that govern the system (which are used for movement of the physical entities). In our analysis, we used the tool KeYmaera [5], an implemented differential dynamic logic prover, to validate our models.

In this paper, we will first walk through the physics of this system,

including our analysis of the descent and the inverted pendulum problem that represents balancing the rocket. From this analysis, we will derive the necessarily preconditions of the system as well as the safety conditions which must hold throughout. After said conditions are listed, we will describe the development process for our controller. We took a different approach than the usual for the development of the controller. We recognized that this descent is split between two different physical phenomena, and we developed a separate controller for each one. As will be discussed, we believe that developing a controller for each phenomena, and allowing them to work with any variant of the other controller, is a more general and better solution than developing a controller that accounts for both.

2 Related Work

During liftoff and throughout its flight, a rocket must maintain very strict safety conditions and trajectories. Even the smallest deviation is considered an extreme risk, as a crashing rocket is essentially a missile. Therefore upon the first sign of danger a range safety officer will detonate the rocket through its flight termination system. With multiple millions of dollars at stake [6], much research has been invested into validating the safety of rockets. Companies like Boeing have invested heavily into formal and statistical verification [1].

However, the thought of a rocket being reusable, even partially, is an incredibly recent concept. The recent endeavors of SpaceX to build and test prototypes of their Falcon 9 Reusable rocket are pushing the limits of this technology. Sadly, due to governmental restrictions and the young age of the field of research, very little information on the testing mechanics is publicly available. However, due to the rapid speed of development of these rockets, it is reasonable to infer that the majority of testing is done statistically and not formally.

3 The Physics of a Returning First Stage Rocket

During the development of our model for the returning first stage rocket, we spent significant time researching the physics of the situation, so that we could derive the necessary safety conditions. The portion of the Falcon 9 flight that we chose to analyze is immediately after the deployment of the landing legs, which occurs after the initial vertical flip and primary stage separation. We chose this portion of the flight to analyze because it is the

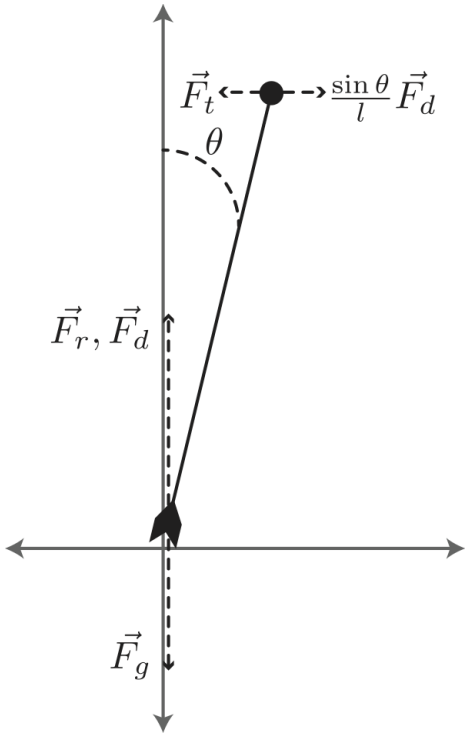

Figure 1: A free body diagram of the entire physical system.

most important section for re-usability, which is where the largest impact of this innovation will be felt. In order to successfully validate the return of the first stage rocket for this section of the flight, we need to properly model a system, and then a controller for this system, that accounts for the two physical phenomena that occur during the section. Our ultimate goal is to develop controllers that successfully account for both phenomena.

Figure 1 is an free body diagram of both physical phenomena in our system. It shows all of the forces acting on the first stage rocket during the descent, and how the tilt angle θ relates to the forces applied on the rocket.

The first physical phenomena that we need to account for in our system is gravity. It is fairly obvious that during the return of the first stage rocket, gravity will continue to accelerate the rocket towards the ground until it eventually crashes. The first stage rocket is able to counteract gravity by using its main engines to create a thrust in the opposite direction, thus decelerating it. The physics behind the descent can be described by the

following equation:

$$\frac{dp}{dt} = F_g + F_d + F_r \tag{1}$$

In the equation above, the left hand side represents the change in momentum, while the right represent the summation of three acting forces, namely of gravity, the rocket's thrust, and any possible retarding forces such as air resistance.

However, we developed our model to drastically simply this equation through reasonable assumptions for the physical system. The first assumption is that there are no retarding forces acting on the rocket, or any force that resists motion (which are typically proportional to the velocity). This includes forces such as air resistance.

We then make the assumption that the first stage rocket travels through a constant gravitational field throughout its descent, simplifying the gravitational force,

$$F_g = G\frac{m_1 m_2}{r^2} \approx mg. \tag{2}$$

Finally, we make the assumption that the mass of the first stage rocket does not change and all motion is non-relativistic. Making this approximation results in a change to the left side of Equation (1),

$$\frac{dp}{dt} = m\frac{dv}{dt} + v\frac{dm}{dt} = m\frac{dv}{dt} = ma. \tag{3}$$

During the descent of the first stage rocket, it burns fuel to provide thrust to decelerate itself. Without any retarding forces (that may have mass dependence), a constant larger mass implies more thrust needs to be applied in order to slow the rocket. This means that by making the assumption of constant mass, the system is forced to decelerate a heavier object, which is harder to prove the safety of. Therefore, proving the system with the assumption is safe also proves safety for the system without the assumption. These assumptions reduces the equation of motion for the descent of the first stage rocket to

$$a = -g + F_d. \tag{4}$$

Note that we assume unit mass for our first stage rocket. This assumption does not change our model, it merely removes a variable from the system to ease calculations.

The second physical phenomena that needs to be accounted for is the possibility of the first stage rocket tipping over during the descent. When looking at just the first stage rocket after stage separation, it can be modeled as a rod with a point mass at either end. The bottom point represents

the weight from the engines, and the top point represents the weight from the connections to the second stage. The intermediate weight can be disregarded, as most of the fuel is gone and the fuselage is incredibly lightweight. When on the ground, and without the landing legs deployed, if the first stage rocket is at any degree of tilt θ, gravity will cause a torque to be exerted on the rocket. This torque will cause the rocket to pivot about its base and cause θ to grow. By setting up and solving the appropriate Lagrangian equations, it can be shown that the equation of motion that define the change in θ due to gravity's torque is

$$\ddot{\theta} = \frac{g}{l} \sin \theta. \tag{5}$$

One interesting fact about this equation is that there is a steady state solution of $\theta = 0$. When there is no angle of tilt, gravity does not cause a torque and therefore θ will never change. However, in reality, this cannot happen. There will always be some slight deviation from the center of mass over the center of the base of the first stage rocket, so $\ddot{\theta} \neq 0$. This nonzero $\ddot{\theta}$ will eventually cause the rocket to tip over. Therefore, our system must be able to provide a force in the opposite direction to balance the first stage rocket. This changes the equation of motion to

$$\ddot{\theta} = \frac{g}{l} \sin \theta + F_t \tag{6}$$

where F_t is the force the first stage rocket can provide using its side thrusters.

However, this equation is still making the assumption that our rocket is grounded. When descending from the peak of the rocket's trajectory, it is hopeful that the rocket is significantly above the ground. A classic result of physics is that in absence of retarding forces (which our simulation is as due to assumptions made previously), a rigid object in free fall will maintain its rotation forever. This can be seen from the previous equation, as in free fall the first stage rocket no longer feels the gravitational field as it would on the ground, and Equation (4) becomes

$$g = 0 \implies \ddot{\theta} = \frac{0}{l} \sin \theta = 0 \tag{7}$$

Another interesting case, which is the case we use in our analysis, is when the rocket provides an acceleration when trying to slow its descent. It is this case that ties together the descent and the balance of the first stage rocket. When the rocket uses its main engines to slow its descent, it leaves free fall, and locally feels a force equivalent to gravity if the gravitational constant

was that of its acceleration. Therefore, if we assume for sake of simplicity that the rocket is of unit length, the equation of motion that defines the angle of tilt for the first stage rocket during the return is

$$\ddot{\theta} = a \sin \theta + F_t \tag{8}$$

where a is the vertical acceleration currently being provided by the main engines. It is this equation that unifies the two phenomena in our model. In the next section, we discuss the initial and safety conditions that any controller must abide by in order to successfully validate this system.

4 Controller Initial and Safety Conditions

In order to successfully verify and validate a controller for the return of a first stage rocket, we first need to define the initial and safety conditions that it will maintain. Using the equations of motion for our system, as defined in Equation (4) and Equation (8), we can condition the variables in our system to assure safety.

Below is the list of initial conditions that our system meets to meet the requirements of our model.

- Gravity has some constant gravitational coefficient $G > 0$. This coefficient is constant due to assumptions previously stated and justified in Equation (2). This coefficient must be positive to correctly model the physical phenomena of gravity.

- The first stage rocket can land with some maximum safe velocity $V_{safe} < 0$. This velocity is negative as the first stage rocket is allowed to have some downwards velocity on impact, but cannot have a larger downwards velocity without crashing.

- The first stage rocket can provide a maximum thrust force $F_d > G$. It is very important that the rocket's main engines can provide enough force to overcome gravity, as without this assumption the first stage rocket could not possibly decelerate to a safe landing speed.

- The first stage rocket can provide a maximum tilt force F_t.

- The first stage rocket will start at some non-negative height $h_0 = H \geq 0$. This is necessary, as it is a physical limitation that the rocket's position is above the ground.

- The first stage rocket initially has some downwards vertical velocity, $\dot{h}_0 < 0$. This is equivalent to starting the simulation immediately after the initial rotation and deployment of the landing legs, because those two events occur after the peak of the rocket's trajectory which occurs with $v = 0$.
- The first stage rocket initially has some angle of tilt $-\theta_{\max} \leq \theta_0 \leq \theta_{\max}$.
- The first stage rocket initially has no rotational velocity $\dot{\theta}_0 = 0$.
- As our controller is a time-based controller, there is some refresh rate of $T > 0$ in which after at most T time it can make another controller decision.

These initially conditions completely describe the state of the system before the beginning of the simulation. We now also define the safety conditions that our system must maintain in order to meet the requirements of our model. We do not consider efficiency conditions here, as they are not necessary to verify the safety of our system. They are, however, a bonus optimization that makes our controller more likely to be used in a practical application. Below are our safety conditions.

- The height of the rocket must always be non-negative as a negative height would correspond to a state that is not physically possible.
- When landing, the first stage rocket must have a vertical velocity of a magnitude of at most V_{safe}, $|v| \leq V_{safe}$. This is to ensure that the rocket does that land so fast that it either breaks its landing legs or the spaceport below it.
- When landing, the first stage rocket must have an angle of tilt between the maximum allowed, or $-\theta_{\max} \leq \theta \leq \theta_{\max}$.

With these safety conditions defined, we can revisit our initial conditions and update them. In order to properly validate our model, our system must initially hold the safety conditions. Furthermore, we must impose additional, stricter conditions in order to ensure that our safety conditions can continue to be satisfied as time passes. The particular restrictions necessary are:

- To address the safety conditions that we will specify below, the first stage rocket needs to either have its initial downwards velocity be greater than the safe velocity, $v \geq V_{safe}$, or it needs to be able to slow

down to a safe velocity. We know that the first stage rocket will be able to slow itself down to a safe velocity if the acceleration needed to change its velocity over its initial height is less than the maximum thrust force it can apply minus gravity, $(v^2 - V_{safe}^2)/2h \leq F_c - G$ (Note: Its almost important that $h \neq 0$).

We now have described the initial conditions and safety requirements for both of the physical phenomena present in our system, meaning that we have defined the total requirements for the complete system. In the next couple sections, we describe how we developed our controllers to correctly accommodate these requirements, starting with the two separate controlled descent controller and balance controller.

5 Controlled Descent Controller

Our first controller is one that is responsible for controlling the first stage rocket's descent. Starting after the initial vertical rotation and deployment of the landing legs, it is responsible for decelerating the rocket until its touchdown on the ground or on the autonomous spaceport drone ship. We developed this controller to be safe, efficient, and to work with any given balance controller.

In order to provide safety, the controller needs to be able to slow the first stage rocket down to a safe velocity V_{safe} by the time it has reached the landing height $h = 0$. From our initial and safety conditions, we know that there must exist and acceleration that the first stage rocket can provide that will meet this conditions, so the first strategy is to have the first stage rocket provide that force until it reaches the ground. As this meets the desired safety conditions, this controller is acceptable.

However, it can be improved. By looking closer into the different situations that the first stage rocket can be in, we expanded upon the base controller by allows for some more efficient solutions. The first situation is when the first stage rocket is close enough to the ground and moving slowly enough that it can simply free fall to the target. When this is the case, we allow the first stage rocket to do so, as it is the most efficient solution available (as it uses no fuel and also arrives in the shortest amount of time). The other situation is when the first stage rocket is already at a safe downwards speed. If this is the case, then the first stage rocket has no need to use it's main engines to completely cancel out the force of gravity, as it is safe to speed up. Our controller sets the engines to allow the rocket to accelerate

up to the safe speed by the time it lands. This allows it to use less fuel and arrive faster.

It's important to mention that this controller can work is any balance controller. Granted, this is trivially true because the balance controller has no effect on the descent, it's still important to mention because if we develop a balance controller that works with any descent controller, then we have shown a completely general solution to safely returning the first stage rocket (with some efficiency included!).

The *.key file for this controller can be found in the Appendix.

6 Balance Controller

The second controller we made, the balance controller, is responsible for assuring that the rocket does not tip over during the descent. Starting at some original θ, the balance controller is developed to ensure that after landing the first stage rocket maintains an angle $0 \leq \theta \leq \theta_{max}$. Note that this is different than the bounds described in the safety conditions. However, by the design of our controller, basic symmetry arguments can be applied to account for the other half of the initial and safety conditions.

To provide this safety, our controller uses the assumption that initially, $\dot{\theta} = 0$. Our controller then assigns an acceleration to cancel out the acceleration caused by gravity, the exact amount needed to set $\ddot{\theta}$ in Equation (8) to 0. This means that the rotational velocity never changes, and because it was initially 0 the rotation never changes. Therefore if the first stage rocket was initially safe (which is must be by the initial conditions), then it will remain safe. However, one variable that can change is the acceleration force felt by the rocket to due an increase or decrease in vertical deceleration. Our controller allows for this to change at every time step, and then adjusts for the change in force. This means that our balance controller is general enough to handle any possible descent controller!

One difficulty that arrived when designing this controller is that differential dynamic logic cannot directly represent sine and cosine, as they do not always map to rational numbers. However, we were able to account for this by using the approximation that $\sin x = x$, turning Equation (8) into

$$\ddot{\theta} = a\theta + F_t. \tag{9}$$

. We used this approximation in both our controller as well as in the differential equations that control the physical evolution of our system. While the use of approximations in differential equations can result in models which

do not properly represent the physics of the system, we believe their use is justifiable here. Because our tilt thrust is always chosen such that the net rotational acceleration is zero, we simply needed to be consistent with our calculations between the one used in the controller and the one used by the differential equations. The actual value we calculate is essentially irrelevant because it gets cancelled out in the summation.

The *.key file for this controller can be found in the Appendix.

(Aside) Advanced Balance Controller

After completing the first revision of our balance controller, we explored expanding it to remove one of the initial conditions of our system. One of the major assumptions our previous controller relied on was that $\dot{\theta} = 0$. This condition allowed us to maintain a constant angle of tilt throughout the descent, as we were able to have the tilt thrusters cancel out the rotational acceleration caused due to the vertical deceleration. However, if we remove this assumption, then simply canceling out the rotational acceleration would not maintain our invariant, because some initial rotation velocity would cause the rocket to tip over at a constant speed. The goal of this controller would instead to be align the first stage rocket, or to end with $\theta = 0$. Another benefit of being able to create a controller that would aggressively straighten the first stage rocket is that it would be prepared to handle future additions to the system, such as noise and retarding forces.

However, when working to develop this controller, we ran into significant difficulties. The largest difficulty comes from the equations of motion that describe the balance controller. As seen in Equation (5), the equation of motion that governs the physics has the rotational acceleration proportional to the sine of its rotation. Despite its simple appearance, the solution to this second-order nonlinear ordinary differential equation is incredibly complicated. Solving this equation results in

$$\theta = \pm \mathrm{am}\left(\frac{1}{2}\sqrt{(c_1 - 2)(t + c_2)^2} \mid -\frac{4}{c_1 - 2}\right). \tag{10}$$

where c_1 and c_2 are constants of integration and $\mathrm{am}(x \mid m)$ is the Jacobi Amplitude function. Even if this equation was solvable analytically, we have no hope of being able to use it in our verification. KeYmaera, and differential dynamic logic as a whole, is severely limited in its operation over irrational numbers such as e and π as it cannot reason about them directly. The solution to this differential equation is founded in exponentials, so we cannot represent it or reason about it in our directly logic language of choice.

We also note that even if we make the common approximation that $\sin x \approx x$ in the physics (which is usually improper and may even invalidate the proof), the equation is still a second-order nonlinear ordinary differential equation with solution:

$$\theta = c_1 e^t + c_2 e^{-t} \tag{11}$$

which is relatively simple but still has the problem that we cannot represent it directly due to the exponentials.

However, despite being unable to reason about the solution directly, doesn't mean we cannot reason about the solution at all. For example, our a previously developed circular motion controller was able to reason about these sine and cosine indirectly through the use of differential equations. A method we explored was to bound the possible acceleration and distance traversed in one time period of our controller. As done in the previous controllers developed, we can use an upper bound on the possible displacement each iteration by placing an upper bound on the rotational acceleration the first stage rocket could experience. This leads to

$$\theta_f = \theta_0 + v_0 t + \frac{1}{2} F_d t^2 \tag{12}$$

where $\sin x$ is upper bounded by 1 and the descent acceleration is maximized to F_d. Sadly, there is a serious flaw with bounding the change in rotation by with equation. If we use said equation and bound displacement this way, we assume that there is a constant acceleration. The largest challenge from designing a controller about the equation of motion is that when $\theta \neq 0$, the force applied on the first stage rocket changes with its rotation. Because a controller must assign some tilt thrust and then wait some set period of time (as it is a time based controller) before it can adjust to a new tilt thrust, we were unable to develop a controller that could successfully cope with the changing force from the $\sin x$ factor. A controller that uses a constant acceleration for its bounds will not take the current rotation into account, and can easily under or overestimate the force being applied on the first stage rocket, which can lead to unsafe conditions.

7 Conclusions and Future Work

Reusable rockets would be a revolutionary step towards humanity's destiny with spaceflight, and SpaceX has made major steps towards this goal. A formal verification of these systems furthers this progress, by showing that a reduced form of the system can be logically proven. We developed a

model and accompanying controller in differential dynamic logic and successfully used KeYmaera to verify and validate them. Our results, created by sequentially developing controllers to account for different physical phenomena, formally guarantee that our system of a returning first stage rocket will successfully land, meeting the necessary safety conditions.

We developed two individual controllers, each responsible for an individual physical phenomena that occurs during the return of a first stage rocket. The first is the controlled descent controller, which provides safety and efficiency for the descent and is also general enough to account for any balance controller. The second is the balance controller, which provides safety and prevents the rocket from ever tipping over. This controller is also general enough to account for any controlled descent controller. These two controllers combined cover a completely general solution to the return of the first stage rocket, and also allow for expansion due to their generality.

When constructing our controllers and system, we made sure to pay careful attention to the surrounding physics to assert that we were creating a valid model of the reality of the returning first stage rocket. However, in this analysis, we made many assumptions that were necessary to reduce the complexity of our model to be solvable with our knowledge and tools for cyber-physical systems in the time given. This leaves open the possibility for further research to extend our analysis. Such areas for future work could be exploring fuel considerations and how limited resources could favor differ controllers, such as those that do not always have their engines burning. Another would be the inclusion of retarding forces and noise. Finally, a major contribution to this project would be the analysis of a larger portion of first stage rocket's trajectory, or even more components of the launch of a Falcon 9 rocket.

Throughout the research, design, development, composition and eventual completion of this project, equal work was done by both team members.

References

[1] Boeing. Cyber-physical systems: An aerospace industry perspective, 2008 (access 2014-12-03. `http://www.ee.washington.edu/research/nsl/aar-cps/winterrev4.pdf`.

[2] Miriam Kramer. Spacex to try rocket landing on floating ocean platform, elon musk says, 2014 (accessed 2014-12-04. `http://www.space.com/27538-spacex-reusable-rocket-test.html`.

[3] Nicolas Mokhoff. Intel, motorola report formal verification gains, 2014 (accessed 2014-12-01. http://www.eetimes.com/document.asp?doc_id=1215957.

[4] Jason Paur. Spacex tests precision landing rocket, 2014 (accessed 2014-12-04). http://www.wired.com/2012/11/spacex-nasa-milestone-ccicap/.

[5] Andre Platzer. Keymaera: A hybrid theorem prover for hybrid systems, 2014 (accessed 2014-12-04). http://symbolaris.com/info/KeYmaera.html.

[6] SpaceX. Falcon 9 capabilities & services, 2013 (access 2014-12-10. http://www.spacex.com/about/capabilities.

Appendix

Attached to this paper is the source *.key files for our two controllers. Descent:

```
/* REQUIRED INFORMATION:

Statistics can be obtained from Proof -> Show Proof
    Statistics.

Name(s): David Franklin, Phil Massey
AndrewID(s): dfrankli, pmassey
KeYmaera version: 3.6.15
Backends used (Mathematica, Z3, ...): Mathematica
    10.0.1.0
Nodes: 622
Branches: 57
Interactive steps: 33
Arithmetic Solver: 0.693
Time: 0.023
Proof completes (Y/N): Y

*/

\functions {
    R AV;       /* Spacecraft's vertical acceleration */
```

```
    R G;       /* Acceleration due to gravity */
    R T;       /* Time-trigger limit on evolution */
    R V_safe; /* Upper bound on safe landing
       velocities */
}

\programVariables {
    R h;     /* Spacecraft's current height */
    R v_h;   /* Spacecraft's current velocity */
    R a_h;   /* Spacecraft's current acceleration */
    R t;     /* Time */
}

\problem{
(
    /* Initial Conditions */
    G > 0 &
    T > 0 &

    /* Safe velocity is some downwards speed */
    V_safe < 0 &

    /* The initial velocity is some downwards speed */
    v_h < 0 &

    /* The rocket must be able to apply some thrust to
        overcome gravity in
     * order to decelerate at all */
    AV > G &

    /* The initial hieght is some positive height */
    h > 0 &

    /* The rocket must initially be safe. This means
       that it either is
     * already at a safe speed, or if it's off the
        ground it is able to
     * slow down to a safe speed in the given distance
         with the known
     * maximum thrust */
```

```
    ((v_h >= V_safe) | (!(h=0) & (V_safe^2 - v_h^2) /
       (-2*h) <= (AV - G)))
)
->
\[
    (
        /* Control acceleration */

        /* If the rocket is not at a safe speed,
           choose an acceleration
         * such that it lands at a safe speed */
        if (v_h < V_safe) then
            a_h := (V_safe^2 - v_h^2) / (-2*h)
        else

            /* If the rocket can free fall and still
               have a safe velocity,
             * do so as it's efficient */
            if (-1*(v_h - V_safe) / T < -G) then
                a_h:= -G

            /* If the rocket cannot free fall but is
               at a safe velocity,
             * choose an acceleration such that it's
                still safe. */
            else
                a_h:= -1*(v_h - V_safe)/T
            fi
        fi;

        /* Evolve */
        t := 0;
        {
            v_h' = a_h &
            h' = v_h &
            t' = 1 &
            t <= T &
            h >= 0
        }@invariant (
```

```
            /* The vertical velocity can never be
               positive -- inefficient */
            v_h <= 0 &

            /* The vertical acceleration can never be
               more than when the
             * engines are at their maximum thrust nor
                 can be less than when
             * they are off */
            a_h <= AV - G &
            a_h >= -G &

            /* At every given timestep, the rocket
               needs to be able to have
             * a reachable acceleration such that it'
                ll land with a safe
             * speed */
            ((v_h >= V_safe) | (!(h=0) & (V_safe^2 -
               v_h^2) / (-2*h) <= (AV - G)))
        )
    )*
    @invariant(
        /* Loop invariants */
        /* The vertical velocity can never be positive
            -- inefficient */
        v_h <= 0 &

        /* The rocket can never go below the ground */
        h >= 0 &

        /* Again, the rocket must either be safe or
           able to become safe */
        ((v_h >= V_safe) | (!(h=0) & (V_safe^2 - v_h
           ^2) / (-2*h) <= (AV - G)))
    )
\]
(
    /* Safety condition */
    /* The rocket cannot be below the ground, and if
       it's not currently at
```

```
     * a safe speed then it must be above the ground
        */
    h >= 0 & (v_h < V_safe -> h > 0)
)}
```

Balance:

```
/* REQUIRED INFORMATION:

Statistics can be obtained from Proof -> Show Proof
   Statistics.

Name(s): David Franklin, Phil Massey
AndrewID(s): dfrankli, pmassey
KeYmaera version: 3.6.15
Backends used (Mathematica, Z3, ...): Mathematica
   10.0.1.0
Nodes: 149
Branches: 10
Interactive steps: 0 <- This isn't true though. Used
   re-use from a prev. vers.
Arithmetic Solver: 0.771
Time: 0.043s
Proof completes (Y/N): Y

*/

\functions {
    R AV;       /* Spacecraft's max vertical
       acceleration */
    R T;        /* Time-trigger limit on evolution */
    R MaxARot; /* Spacecraft's max rotational
       acceleration */
    R MaxRot;  /* Maximum rotation we're allowing the
       spacecraft to be in */
}

\programVariables {
    R a_h;     /* Spacecraft's current vertical
       acceleration */
```

```
    R a_rot;  /* Spacecraft's current rotational
       acceleration */
    R v_rot;  /* Spacecraft's current rotational
       velocity */
    R t;      /* Time */
    R rot;    /* Angle between the spacecraft and the
       y axis; */
}
\problem{
(
    /* Initial conditions */
    AV > 0 &
    T > 0 &
    MaxRot > 0 &

    /* The maximum rotational acceleration must be
       enough to overcome the
     * force that would be applied in the most extreme
       situation, which
     * occurs when the rocket is at it's maximum tilt
       and is decelerating
     * with the maximum thrust */
    MaxARot >=  AV * MaxRot &

    /* No initial rotation velocity or acceleration --
       assumption */
    v_rot = 0 & a_rot = 0 &

    /* Initial rotation must be safe, cannot be
       outside the bounds */
    0 <= rot & rot <= MaxRot
)
->
\[
    (
        /* Non deterministically assign a value to the
           acceleration so we
         * can later combine this with a strategy for
           descent */
        a_h := *; ?(0 <= a_h & a_h <= AV);
```

```
        /* Choose appropriate rotational acceleration
           */
        /* The acceleration chosen is one to stop any
           rotational velocity */
        a_rot := -a_h * rot;

        /* Evolve the system */
        t :=0;
        {
            v_rot' = a_rot + a_h * rot &
            rot' = v_rot &
            t' = 1 &
            t <= T
        }
        @invariant(
           /* Differential invariant */
           t >=0
        )
    )*
    @invariant(
        /* Loop invariant */
        /* The acceleration our controller applies
           cannot be greater than
         * the maximum amount it is allowed */
        -MaxARot <= a_rot & a_rot <= MaxARot &

        /* Our controller is designed to not allow any
            rotationl velocity */
        v_rot = 0 &

        /* The rotation cannot leave either bound */
        0 <= rot & rot<= MaxRot
    )
\]
(
    /* Valid controls -- cannot be accelerating the
       rocket more than the
     * maximum amount allowed */
    -MaxARot <= a_rot & a_rot <= MaxARot &
```

```
    /* Safety condition */
    0 <= rot & rot<= MaxRot
)}
```