R2D2 Goes to Buggy

Anastassia Kornilova and Emily Yeh

12/9/14

1 Abstract

In this project, we set out to model a buggy car that is automatically controlled by R2D2 on a track of curves and hills. We analyze the kinematics of motion in these environments to create constraints which ensure safe control decisions. Guaranteeing the safety of this roads is important for a number of applications ranging from controlling self-driving cars to constructing safe race tracks. Formal verification techniques applied in this paper would ensure safety in all experimental situations, as long as initial constraints are met. Finally, we prove that the proposed models verify the safety and efficiency of vehicles on wide variety of roads.

2 Introduction

Following one of Carnegie Mellon University's oldest traditions, we sets out to model the safety and efficiency of the buggy race. Traditionally, buggy is a race in which drivers in made-made bullet-shaped cars are pushed and free-rolled around a track. The track consists of uphill sections in which the buggy is pushed from behind by a human pusher/runner and a downhill free-roll section where the driver has control of braking and steering. Buggy racing has always presented an interesting challenge of combining safety (to ensure the buggy doesn't crash) and efficiency (to win the race).

The models presented in this paper approach the complexity of buggy racing by simulating a vehicle's movement around varying continuous turns in the track. Instead of a pusher and driver, though, the buggy is controlled by R2-D2, who has control over acceleration and braking of the buggy car. We show that the buggy can stay within the track while maintaining a safe velocity, even as the track evolves. At the same time, we also show that the buggy can maintain at least a minimum velocity throughout the race which leads to improved efficiency. We explored tracks modeled by helixes with either varying slopes of radiuses; this allowed us to model a wide variety of possible curves and hills found on real roads. To simulate the numerous conditions that may occur on real roads, our models randomly generate track segments, and then the controller (R2D2) makes a decision about acceleration that will keep it in the track. To ensure that R2D2 can always make a safe decision, we derive constraints of the track and acceleration based on properties of kinematics. These properties can be used in live systems to guarantee that racing tracks are safe and that cars have sufficient braking power. By adding these properties, we were able to formally verify that the buggy can safely travel through a variety of dangerous twists and turns.

While the systems verified in this project will help R2-D2 win the buggy races, they can be applied to any vehicles which encounter hills and turns. Flagship companies in the automobile and technology industries that are developing self-driving cars desire to develop safe controls for a variety of roads. Experimental methods have helped develop pseudo-safe models and simulations, but cyber-physical systems models provide symbolically proven controls that guarantee safety regardless of experimental conditions. These models could also be used in driving assistant systems by providing insight to help drivers be both safe and efficient. The models discussed in this paper provide an important step forward towards safe controls of turns and hills.

In Section 4, we discuss the details of the system and the physics used to model this system. In Section 5, we go over the constraints derived to ensure the safety and efficiency of the system.

3 Background

Our project builds on methods used for collision-avoidance in robotics and similar cyber-physical systems.

In robotics, motion planning algorithms are used automatize control decisions for robots in a variety of environments. These algorithms helps robots avoid collisions through prior knowledge of the workspace or sensory data. Basic path planning algorithms look for a ”geometric collision-free path for a single robot in a known static workspace” [5]. Similarly to this technique, we use known properties of the track to predict if a collision is possible given a control decision.

While we know the general properties of our track, we do not know the global layout of the track because it is randomly generated. this is an accurate reflection of live environments, but several techniques for local collision avoidance have been developed in robotics. The Dynamic Time Window Approach is a local control planning technique that splits up motion into time steps. At every time step, the algorithm evaluates the current state and derives an approximate model of the admissible trajectories of the robot [2]. Nearness Diagram Navigation was developed for collision avoidance in densely cluttered areas using divide and conquer approach based on different possible scenarios [6]. We used a simplified divide and conquer approach to create safe controls in all possible tracks that our model generates. While our models deal with environments simpler than the ones that these collision-avoidance algorithms were developed for, those algorithms were only tested experimentally, while our model is formally verified. Thus, our models will work with any live conditions as long as they meet the initial conditions and constraints associated with each model.

Several cyber-physical systems have incorporated properties similar to the ones in our paper. Circular motion for robots has been considered in the CMU’s 15-424 class; however, this models do not account for effects of centripetal acceleration and gravity. 3-D motion effects are also seen in roller coaster simulations. Prior research modeled these 3-D systems by modeling x, y, and z components separately as directional forces. Virtual roller coaster models incorporate track friction in x-direction forces, wheel friction in y-direction forces, and effects of gravity and normal force are included in the z-direction [7].

One self-driving car system models future paths in order to determine the best course of motion. The car controls its curvature and velocity as it navigated, using an unstable or uncomfortable curvature as the bounds of the safety condition [3]. This is consistent with the safety conditions used for the systems presented in this paper, in which the curvature is dependent on a velocity that is bound by a minimum and maximum constant. The self-driving car system generates potential paths and chooses the best path by choosing the one with the lowest expected cost. One issue presented by this paper that we had to account for was that there was a cost in generating these paths. With different road shapes, obstacles, and car velocities, there was greater risk associated with taking more time to choose a best path, the trade-off being that finding more paths increased the confidence in the chosen best path. Our model avoids this cost because we constrain the possible tracks and control decisions to several variables; however, because these variables are symbolic, they can represent a number of real values. Furthermore, while the cyber-physical paper described is used in simulations, it is not formally verified. Thus we improve on existing systems by providing a model that can be proven to be safe regardless of specific experimental conditions.

4 The Model

4.1 Overview

We built two models. Both represent the buggy moving around a helical track; however, in the Helix model the radius of the track changes and in the Helix_Hills model the slope of the track changes.
Our model contains an automatically controlled car (buggy) moving on a randomly generated track. We use a Hybrid System that pairs discrete decisions with the continuous evolutions of physics. This model is time-triggered; the discrete decisions relating to the track and the buggy are separated by some number of time-steps (where the maximum time between decisions is bounded by an arbitrary variable $T > 0$). In-between the decisions, the model evolves continuously according to properties described in sections 4.3 and 4.4. At each control decision, the next section of the track is generated, and the buggy makes a decision based on the new track segment. The process for generating the track is described in section 4.2; the process for making control decisions is described in the section 4.4.

This model simplifies live systems in several ways: first, the controller could make decisions more often than the track changes, second the track changes may not always be synchronized with the control decisions, and finally, the controller may not always know how the track is going to change. The first caveat is handled by the model, as making multiple decisions in one track segment is equivalent to the track making the same decision multiple times in a row, but by choosing different controls. The second simplification helps create less conservative controls, since if the track could change unpredictably in-between the robot's decisions, only very conservative decisions may be possible. If the controller had some knowledge of the upcoming decisions, then the control decisions could be extended from the ones described in this paper. Furthermore, in live systems, major changes to a track are likely to happen less often than one has access to controls (i.e the lag between control decisions may be within a couple of seconds, but changes in the track may happen closer to a minute apart). Finally, the controller may not immediately know how the track changes, but with good sensors it could find out soon after the track changes.

4.2 The Track

We used a track shaped like a helix with a fixed width. In our models we varied two parameters: the slope and the radius of the helix. Sections of this helix will represent a wide variety of turns found in real-life tracks. In the Helix model, the slope is kept constant, but the track radius is varied: it can expand or shrink linearly, or remain the same. A track that is expanding is demonstrated in the following figure. In the Helix_Hills model we vary the slope by increasing or decreasing it, but

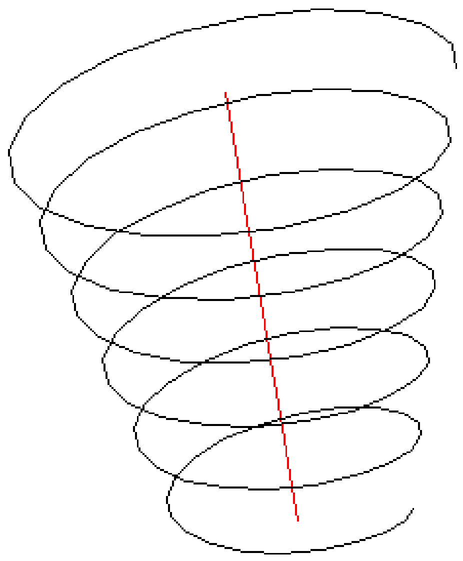
Figure 1: Expanding Helix

keep the radius fixed. Varying these two parameters allows us to model a variety of turns present on real roads. In the model of the actual buggy course below, we can see a number of turns that can be modeled with short sections of the helix.

4.3 Physics of Motion

Because the model uses an helical track, effects of both motion on an incline and circular motion are considered. Since a helix is a circular track with a slope, we can consider the physics for circular motion in a flat plane if we adjust the forces to reflect the incline.

Physics of Incline On a given segment, the slope has a constant angle with respect to the ground. If this slope has angle θ, then by definition, the normal force is: $F_N = m * g * \cos(\theta)$. The normal force is used to determine the friction in the plane of motion. If the coefficient of friction is μ, then the force of friction is $F_F = \mu * F_N$ [4].

Physics of Circular Motion In the plane of the track, the buggy is moving around a level curve.

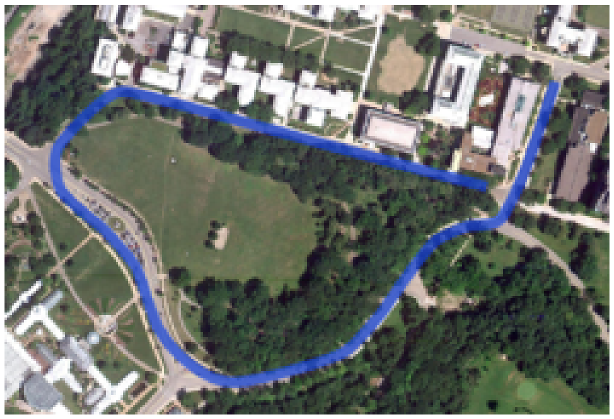

Figure 2: CMU Buggy Course

In this situation, the centripetal force is supplied by the force of friction [1]:

$$F_F = F_C \tag{1}$$

$$\mu * m * g * \cos(\theta) = m * v^2/r \tag{2}$$

The r denotes the radius of travel in the plane of the track relative to the center. If we let $fr = \mu * g * \cos(\theta)$, cancel out mass and rearrange the terms, we get that:

$$r = v^2/fr \tag{3}$$

In the model, we assume that fr is constant because we assumed that μ and θ stays the same. While on a live track, μ may vary, it would not vary significantly, unless there were unexpected hazards like oil or ice. While we vary the slope in the Helix_Hills model, we do not consider the effect of changing slope on the fr parameter; this is a simplifying assumption that allows us to reduce the complexity of the model. Furthermore, on real hills, the $\cos(\theta)$ may not change very significantly as θ varies.

Physics of Acceleration When the buggy wants to move up or down a slope, it applies a force F_B in the direction of motion. Two additional forces are applied in the downhill direction: the force of gravity ($m * g * \sin(\theta)$) and the force of friction (F_F). We've assume that F_F stays the same throughout all the models. In the Helix model, the force of gravity also stays the same, so when we refer to an acceleration a, we assume that the effect of these forces has been applied. In the Helix_Hills model, we assume the force of friction has been applied, but the effect of gravity is factored in separately.

Given this model, if the buggy is accelerating in the direction of motion, then the radius (r) will change with respect to v.
In the Helix model: If the car is accelerating with some acceleration a from initial radius r_1 and velocity v_1, the new velocity of the car after time T will be:

$$v_2 = v_1 + a * T \tag{4}$$

The new radius will be:

$$r_2 = \frac{v_2^2}{fr} = \frac{(v_1 + a * T)^2}{fr} \tag{5}$$

In the Helix_Hills model: We consider the force of gravity to be g and assume it changes with rate s, then the new velocity will be:

$$v_2 = v_1 + (a + g) * T + s^2 * T/2 \tag{6}$$

The new radius will be:

$$r_2 = \frac{(v_1 + (a + g) * T + s^2 * T/2)^2}{fr} \tag{7}$$

This model is used to describe how the buggy moves and make its control decisions.

4.4 Control Decisions

The buggy only has control over its acceleration. We removed control of steering because we assume that the buggy follows the properties of circular motion. Furthermore, the buggy only has three choices for acceleration: A for accelerating, $-B$ for braking and 0 for coasting. Since the model is symbolic, A and $-B$ can represent many possible real accelerations. The buggy has access to its own velocity, the current radius of the track, and how the track will evolve. It can use these properties to ensure the safety and efficiency properties until the next control decision. The specifics of these control decisions depend on specific model properties and are described in Section 5.

4.5 Safety and Efficiency Properties

Safety is defined in terms of two conditions. The first is avoiding collision with the edges of the track. Specifically, the radius that the buggy is traveling at should be between the radius of the inside and outside edges of the track. Second, the velocity should be below some maximum velocity. For efficiency, we require that the velocity remains above some minimum velocity.

5 Proof Techniques

In this section, we describe the techniques we used to prove the safety and efficiency properties. We use the following variables to describe the properties described in Section 4.

- v - velocity of the buggy
- $vMin$ - minimum velocity
- $vMax$ - maximum velocity
- $buggyR$ - the radius the buggy is traveling at
- $trackR$ - the radius of the inner edge of the track
- $tRate$ - the rate at which the track is expanding
- $width$ - the width of the track

- T - the maximum time between control decisions
- t - some time between control decisions ($0 <= t <= T$)
- g - force of gravity ($Gcos(\theta) for gravity coefficient G$)
- $sign$ - the change in the track's slope

5.1 The Helix Model

In this model, we use the structure outlined in section 4.1. The hybrid system is represented with a loop, where on each iteration a new track segment is generated, then the buggy picks an acceleration, and finally the physical system evolves. There are three ways the track could evolve: expanding with *outRate*, shrinking with $-inRate$ or remaining the same. For acceleration, the buggy can pick between A, $-B$ or 0. The final safety and efficiency conditions are the same as the ones described in section 4.5. Using the variables, there will be:

$$trackR <= buggyR <= trackR + width,\ vMin <= v <= vMax$$

. In the proof, we use a loop invariant to show which properties will be true after each iteration of the loop. The properties in the loop invariant are:

1. $trackR <= buggyR$
2. $buggyR <= trackR + width$
3. $v >= vMin$
4. $v <= vMax$
5. $buggyR = v^2/fr$
6. $trackR + width <= vMax^2/fr$
7. $vMin^2/fr <= trackR$

Because the final safety and efficiency conditions are in the loop invariant (1-4), proving the loop invariant will directly imply the final conditions. The loop invariant conditions follow from the tests associated with the control decisions and the properties of the continuous evolutions.

To ensure that a safe control decision can always be made, we introduce the following constraints on the variables. These constraints also help us test if a control decision can be safely made. The constraints are associated with the different ways that the track can evolve.

Expanding with outRate: In this case, we chose to brake up safe decisions as follows: If the buggy is in the inner half of the track (trackR ¡= buggyR ¡= trackR+width/2), then it will be safe for the buggy to accelerate with A. If the buggy is in the outer half of the track, then it will be safe for the buggy to coast. We add the following constraints to the model, to guarantee that the decisions are safe:

Coasting Safety: When the buggy is coasting in the outer half, it can not collide with the outer edge, because that edge is expanding away from it. However, we need a constraint to make sure the buggy does not collide with the inside edge which is expanding. The property that must be true is:

$$v^2/fr \geq trackR + outRate * t \tag{8}$$

The largest possible value for t is T and the smallest possible value for v^2/fr is $trackR + width/2$, so we plug these values into the constraint:

$$trackR + width/2 \geq trackR + outRate * t \tag{9}$$

$$width/2 \geq outRate * T \tag{10}$$

This inequality implies that the inequality in (8) will be true for all relevant values. Thus this property allows the buggy to coast safely in the outer half of the expanding track.

Acceleration Safety: When the buggy is accelerating from the inner half, it has the risk of colliding with both the inner and the outer edge. In order to avoid collisions with the outer edge, we test for safety at time T because the rate with which the buggyR is changing is accelerating and the rate with which the outer edge is expanding is constant, so once the buggyR gets bigger than $trackR + width$, it can not get smaller than it again. Thus, as a constraint we ensure that accelerating from the middle of the track will be safe:

$$(v + A * T)^2/fr <= trackR + width + outRate * T \tag{11}$$

We substitute $trackR + width/2$ for v^2/fr and $vMax$ for v because these values represents the worst case, and thus imply that all other cases will be safe:

$$(2 * vMax * A * T + A^2 * T^2)/fr <= width/2 + outRate * T \tag{12}$$

In order to ensure that we don't collide with the inner edge, we can not just check the inner edge. The quadratic nature of the buggyR means we can be safe at time T, but not be safe earlier. Consider the following diagram:

Here, the blue line represents the buggyR, the pink line represents trackR, where the x-axis is time and the y-axis is the radius. In this graphic, we see that the buggyR could be safe at beginning and the end, but unsafe in the middle, because it will dip below the inner radius of the track. To resolve this issue, we consider the following inequality.

$$buggyR = (v + A * t)^2/fr >= v^2/fr + 2 * v * A * t/fr \tag{13}$$

We use the right side of the inequality to test for safety, because linear models avoid the situation demonstrated in the figure above. Although this model is more conservative, it is easier to model and let's us avoid dealing with complicated solutions to the quadratics. We now want to satisfy the following equation:

$$v^2/fr + 2 * v * A * T/fr >= trackR + outRate * T \tag{14}$$

For the constraint we substitute $trackR$ for v^2/fr and $vMin$ for v because these variables represent the worst case. This leads to the constraint:

$$2 * vMin * A * T/fr >= outRate * T \tag{15}$$

These constraints ensure that accelerating will be safe from the inner half of an expanding track.

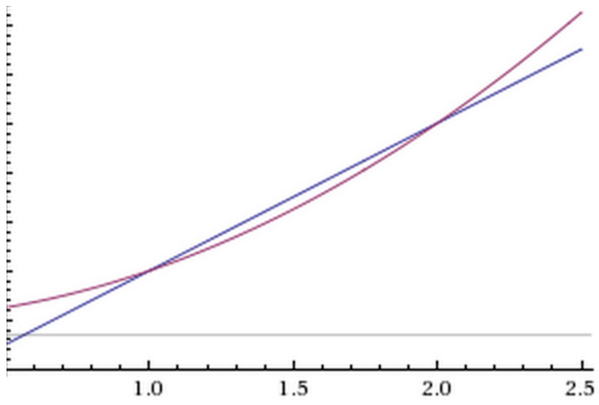

Figure 3: A dangerous case

Shrinking with -inRate: In this case, we add constraints to ensure that it is safe to coast in the inner half and brake in the outer half.

Coasting Safety: When coasting, we need a constraint to ensure that the outer edge does not collide with the track. Using logic as in the expanding track coasting safety, we show that:

$$v^2/fr <= trackR + width - inRate * T \tag{16}$$

by ensuring that we can coast safely in the middle ($v^2/fr = trackR + width/2$):

$$-width/2 \leq -inRate * T \tag{17}$$

Braking Safety: Similarly to the acceleration model, we need to avoid collision with both the inner and the outer edge. The reasoning is derived similarly to the expanding case and leads to the following constraints:
Outer edge Safety:

$$-2 * vMin * B/fr + B^2 * T/fr <= -inRate \tag{18}$$

Inner edge safety:

$$T * (2 * vMax * B/fr - inRate) <= (width/2) \tag{19}$$

In the Inner edge safety, we approximate buggyR with the linear equation:

$$v^2/fr - 2 * v * B * T/fr \tag{20}$$

Remaining the same: In this case, it will be safe for the buggy to coast, because then none of the radii will change and safety will be preserved.

These constraints only refer to constants, so they do not change as the model evolves. These constraints guarantee that a safe decision can always be made, and they ensure that the tests for safety in the control decision lead to the loop invariant conditions. We briefly explain the tests associated with the buggy control:

- A: $(tRate! = outRate | buggyR <= trackR + width/2) \& (v + a * T)^2/fr <= tRate * T + trackR + width \& v + a * T <= vMax)$. If the track is expanding, the constraints ensure that it will be safe to do so in the inner half. Otherwise, we check for collisions with the outer border at the max time because the buggy will be the only one with a growing radius (the track radius will be either constant or shrinking).

- -B: $(tRate! = -inRate | buggyR >= trackR + width/2) \& (v + a * T)^2/fr >= tRate * T + trackR \& v + a * T >= vMin)$. If the track is shrinking, we've ensured that it will be safe in the outer half. For other tracks, it is sufficient to check the endpoint.

- 0: $buggyR >= tRate * T + trackR \& buggyR <= tRate * T + trackR + width$, since buggy is not changing its radius, we make sure that neither side of the track runs into it.

Thus, the acceleration control decisions imply that loop invariants 1-4 will hold after the system evolves. Loop invariants 6 and 7 are guaranteed by the tests associated with the track generation control, which goes as follows:

- $tRate = -inRate$ if $trackR - inRate * T >= vMin^2/fr$.

- $tRate = outRate$ if $trackR + outRate * T <= vMax^2/fr$

- $tRate = 0$ with no additional conditions.

Finally, loop invariant 5 is guaranteed by the ODE for the buggy: $buggyR' = 2 * v * a/fr$. Thus, we have shown that all the loop invariants can be met, which implies that the final conditions can be met as well.
(Note: The described hybrid system and proof can be found in files Helix.key and Helix.key.proof)

5.2 The Helix-Hills Model

The helix-hills model follows the general model structure outlined in Section 4.1. The track of this model is represented by a circle (of constant radius), where the track's slope changes on each iteration. After this new hill slope is determined, the buggy chooses an acceleration and evolves accordingly. The system non-deterministically chooses for the slope of the hill to increase (*sign*=1), decrease (*sign*=-1), or stay the same (*sign*=0). The buggy can control its acceleration by choosing to accelerate by A, -B, or 0. The final safety and efficiency conditions are the same as the ones described in Section 4.5. Initially, the system is defined as satisfying the following variable properties:

$$trackR = vMin^2/fr$$

$$trackR + width = vMax^2/fr$$

$$trackR <= buggyR <= trackR + width$$

$$vMin <= v <= vMax$$

$$-G <= g <= G$$

In the proof, a loop invariant ensures the system's soundness and safety after each iteration of the loop. The loop invariant includes the following properties:

1. $trackR <= buggyR$

2. $buggyR <= trackR + width$
3. $v >= vMin$
4. $v <= vMax$
5. $buggyR = v^2/fr$

The final safety and efficiency conditions are included within the loop invariant, so the final conditions directly follow if the loop invariants are ensured. The initial conditions, variable constraints, and tests of the system ensure that the loop invariants are satisfied as the system evolves through the ODEs. The constraints (similar to the previous model) are associated with the different characteristics of the current track.

Downhill Track: When the track has $g > 0$, the buggy is moving on a downhill slope with a positive force of gravity affecting $buggyR$ and a. The buggy follows the following constraints in order to guarantee its safety.

- Deceleration Safety: When the buggy is at the outer edge of the track and decelerating, it has to avoid reaching $vMax$ (the outer edge) as a result of a changing slope. Therefore, we check to see if $a = A$ would cause our buggy to crash into the outer edge in the worst case: when $v_0 = vMax$.

$$vMax >= vMax + (-B + g) * t + sign * (t^2)/2 \tag{21}$$

In the worst case, t=T and g=G since a large acceleration is caused by the slope if the buggy is moving at maximum g and when sign is also the highest it can be (sign=1).

$$vMax >= vMax + (-B + G) * T + (T^2)/2 \tag{22}$$

This ensures that a downhill slope will not cause us to hit the outer edge when we are braking.

- Acceleration Safety: When the buggy is near the middle (inner half) of the track and accelerating, we want to ensure that it does not crash into the outer edge of the track. The worst case for this situation is when the buggy is right in the middle of the track ($v = (vMax + vMin)/2$) and when the force of gravity is G ($g = G$).

$$vMax >= (vMax + vMin)/2 + (A + G) * T + (T^2)/2 \tag{23}$$

This constraint guarantees that acceleration in the inner half of the track will not cause the buggy to crash into the outer wall.

Uphill Track: When the track has $g < 0$, the buggy is moving on an uphill slope with negative force of gravity. The negative g causes the buggy to risk crashing into the inner edge of the track. The buggy follows the following constraints in order to guarantee its safety.

- Deceleration Safety: When the buggy is on the outer half of the track and braking, an uphill slope introduces a negative component of acceleration that could cause the buggy to crash into the inner edge of the track. Therefore, we want to ensure that braking at the worst case - right in the middle of the track - will not cause us to crash into the inner edge. Similar to the previous constraints, the worst case situation is when $t = T$ and $g = -G$ (also, sign=-1).

$$vMin <= (vMin + vMax)/2 + (-B - G) * T - (T^2)/2 \tag{24}$$

- Acceleration Safety: When the buggy is on the inner half of the track and accelerating, it must be able to accelerate enough so that the uphill slope won't cause it to crash with the inner track. In the worst case, we are at the inner track ($v = vMin$) and trying to accelerate ($a = A$) with a negative acceleration slowing us down ($g = -G$) and negative sign (sign=-1).

$$vMin <= vMin + (A - G) * T - (T^2)/2 \tag{25}$$

Similar to Model 5.1, these constraints do not change throughout the ODEs, so their safety is preserved. The tests ensure that the buggy safely chooses an acceleration given these constraints, allowing the buggy to safely evolve. Prior to the buggy's controlled decision of its acceleration, the system first chooses a new hill slope change.

- *sign*=1 if $g + T <= G$
- *sign*=-1 if $g - T >= -G$
- *sign*=0 with no additional conditions.

Once the system has chosen a new *sign*, the buggy chooses an acceleration. The control tests fall under the following potential decisions:

- A: $((v+(a+g)*T+sign*T^2/2)^2/fr <= trackR+width\&(v+(a+g)*T+sign*T^2/2) <= vMax))$. If the buggy is on the inner half of the track, the constraints ensure that it will be safe to accelerate. However, if the buggy isn't, then it must be true that it will not hit the outside of the track and also not reach $vMax$.
- -B: $((v+(a+g)*T+sign*T^2/2)^2/fr >= trackR\&(v+(a+g)*T+sign*T^2/2) >= vMin)$. If the buggy is on the outer half of the track, the constraints ensure that it will be safe to decelerate. Otherwise, we check that it won't reach $vMin$ or collide with the inner track edge.
- 0:$(((v + (a + g) * T + sign * T^2/2)^2/fr >= trackR\&(v + (a + g) * T + sign * T^2/2)^2/fr <= trackR + width))$. If the buggy is able to coast without reaching the outer or inner edge of the track, then it is safe to choose coasting as a control decision.

The final invariant (5) for this model follows from the ODE $buggyR' = 2 * v * (a + g)/fr$ since $v' = a + g$. Therefore, all of the loop invariants are preserved.

For this model, it's important to note that the friction variable fr doesn't change as the slope of the hill changes. As explained in Section 4.3, the force of friction F_F has an F_N component. Since $F_N = m*g*cos(\theta)$, F_F will also change as the slope changes. However, this system simplifies this effect by keeping fr constant, thereby simplifying the model of v and the buggy's path *buggyR*.

(Note: The described hybrid system and proof can be found in files Helix_Hills.key and Helix_Hills.key.proof)

6 Discussion

In this project, we set out to model how a vehicle moves around a race track in an attempt to model the cyber-physical system of CMU's traditional buggy races. Our models simulate a vehicle's movement around turns with expanding and shrinking radii, as well as a vehicle's movement up and down hills. This successfully incorporated many aspects of our original proposed goal, including moving around turns and on hills. The Helix model completely models a helix with a constant slope and shrinking and expanding radii. The Helix_Hills model successfully consider the effects of a changing slope on the acceleration of the buggy; however, this model does not consider the effects of the changing slope on the change in friction. Furthermore, we were not able to create a model with both a changing slope and changing radius. We were not able to model the later two situations, because they required more complicated constraints that we could not derive. In order to account for effects of a moving border and changing friction the model would need to be very conservative or rely on methods we have not explored.

One issue we encountered was about how to model the track. We initially set out to have a model that count model arbitrary turns and straight sections; however, because we only consider circular motion, our track needed to evolve around a set center. We changed our model to be a helix with varying parameters; while this allowed us to represent a number of turns, this simplification does not allow us to accurately represent all possible tracks.

Overall, the models presented in this paper addressed maintaining safety when moving around a changing track. There are three main ways this project could be further developed: first a model for a changing slope could be complete and incorporated with the Helix model. This would allow to represent an even greater variety of tracks. Second, better efficiency conditions could be constructed that help the robot not only maintain a minimum velocity, but actually pick the fastest path around a turn. Finally, the track generation could be de-coupled from the control decision to represent more realistic systems. Once these steps have been modeled, R2-D2 will be able to move around the racetrack more quickly and fulfill his dreams by winning the race at last.

7 Conclusion

This project presented two final deliverable models: a helix model (Helix) with dynamic track radii and a helix model with variable hill slopes (Helix_Hills). These systems (.key files) were proven (.key.proof files) and successfully simulate how a vehicle could autonomously travel on a helix. Ours models show that with a few constraints on the track and controls, it is possible to have a safe model of a variety of curves. While the pursuit of developing self-driving cars leaves many questions open, the models proposed by this paper show how cyber-physical systems can be used to develop verifiable, safe, and efficient models that can be used to advance automated vehicle technologies.

Equal work was contributed by both partners to this project.

References

[1] Doug Davis. Motion on a curve. http://www.ux1.eiu.edu/~cfadd/1150/05UCMGrav/Curve.html. Accessed: 2014-12-09.

[2] Dieter Foxy, Wolfram Burgardy, and Sebastian Thrun. The dynamic window approach to collision avoidance. Technical report, University of Bonn and Carnegie Mellon University, 1997.

[3] Junsung Kim, Hyoseung Kim, Karthik Lakshmanan, and Ragunathan Rajkumar. Parallel scheduling for cyber-physical systems: Analysis and case study on a self-driving car. Technical report, Real-Time and Multimedia Systems Laboratory, Carnegie Mellon University, Pittsburgh, PA Google Inc., Mountain View, CA, April 2011.

[4] Nina Kozlova. Newton's second law. http://physatwes.com/SecondLawHonors.aspx. Accessed: 2014-12-09.

[5] Jean-Claude Latombe, Lydia Kavraki, and Dan Halperin. Robot algorithms. Technical report, Stanford, Rice and Tev-Aviv University, 1999.

[6] Javier Minguez and Luis Montano. Nearness diagram (nd) navigation: Collision avoidance in troublesome scenarios. *IEEE Transactions on Robotics and Automation*, 20, February 2004.

[7] Zen-Chung Shih, Yuh-Sen Jaw, and Mei-Ling Hsu. Virtual roller coaster. Technical report, Natinal Chiao Tung University, 2002.