RollerCoaster Tycoon X: Like the original, but safer

Adriel Luo (aluo@andrew.cmu.edu)
Xue An Chuang (xchuang@andrew.cmu.edu)

Contents

Abstract

In this paper, we describe how we prove the safety of safety of roller coasters by modeling the track with mathematical functions and then breaking it down into individual segments described by a single function. We then proceed to prove properties of these segments individually before proving things of the coaster as a whole. Additionally, we have come up a graphical interface called *RollerCoaster Tycoon X* for designing roller coasters. *RollerCoaster Tycoon X* informs the user of the safety of the roller coaster at every step of the design and warns them if the coaster they are designing is unsafe.

Introduction

There are an estimated 4300 roller coasters in the world[1]. Roller coasters carry passengers at high speeds and through sharp twists and turns. Riders ride them to feel close to death, but coasters should not actually bring riders close to death. The safety of roller coasters is of utmost importance.

In this project, we make tractable the complex task of modeling and proving the safety of a roller coaster. We achieve this by breaking down a complex coaster into a simple track sections that together form the coaster, and then prove safety properties about the individual track section types to prove the safety of the whole roller coaster. Safety properties include that the coaster train stays on the track, does not roll backwards and reaches the end of the track. Not only does this approach make the original complex problem simpler; it also is more generalizable - we aren't limited to proving the safety of one particular roller coaster. We can prove the safety of any coaster that is made up of the track sections we have in our model.

Once we have proven the safety properties of individual track sections, we know what preconditions are required to make a particular track section safe. Next, we hope to create a tool, *RollerCoaster Tycoon X*, for designing roller coasters that will inform the designer as they go along whether the track created so far is safe, simply by checking that the conditions required are met.

Related work

The problem we are investigating is related to existing work in complex modular systems and component-based modeling. The basic idea is to break down a complex system into pieces that are individually manageable and can be reasoned about, so that the global system can be proven safe. In

[1] https://rcdb.com/census.htm

existing work[2], the safety of large traffic networks is verified by proving the safety of individual traffic flow components. A traffic network is decomposed into smaller parts, and proofs of safety of the parts is transferred to the entire network. We are doing something analogous with roller coasters by first proving the safety of individual track sections in order to prove the safety of the entire coaster. This approach allows proving of entire coaster tracks without requiring a complex proof about the entire coaster at once, similar to how the approach for traffic networks provides high-fidelity verification results without prohibitive analytic costs.

The verification of safety of roller coasters is also related to work in distributed hybrid systems[3]. Modern roller coaster tracks have block brakes that communicate with each other and the station to prevent trains from colliding. Coasters also have multiple trains, each of which is following its own dynamics. Furthermore, the station can choose to release new trains onto the track. Formally proving that the control of block brakes and the station ensures that trains do not collide requires proofs for distributed hybrid systems. In existing work, distributed car control is proven to be collision free. Similar techniques can be used to prove the safety of the distributed hybrid system in roller coasters.

Approach

We make several simplifying assumptions in our model of roller coasters. Firstly, we assume that gravity is the only force acting on the trains. We do not model other forces, such as friction. While this is not true in real life, it greatly reduces the complexity of modeling and proving safety properties. Ignoring friction also does not change the dynamics too much, because gravity is the main force driving all roller coasters.

We also make the assumption of unit gravity ($g = 1$). This reduces allows us to omit the variable g in our models while not invalidating the model, because we can simply scale physical dimensions so that the acceleration due to gravity is 1 unit.

Assuming that gravity is the only force allows us to model trains as unit masses, because all objects have the same acceleration due to the gravity, regardless of mass. This allows us to omit a variable for mass.

We also model a train as a point mass to avoid having to consider rotational physics and also to simplify detecting collisions (trains collide when they occupy the same point in space).

Finally, we model our coasters in two dimensions. This greatly simplifies the model. Even though real roller-coasters are three-dimensional, this remains an interesting problem. Many real-life coasters can be approximately modeled in two-dimensions by stretching out the track to lie in a single plane while keeping the hills and valleys, since gravity is the most pertinent force in roller coaster physics, and it works the same in both two and three dimensions. Long wooden coasters can be especially well modeled in 2D, because they travel for long distances in planar sections.

---

[2] Verified Traffic Networks: Component-Based Verification of Cyber-Physical Flow Systems
http://ieeexplore.ieee.org/document/7313220/
Change and Delay Contracts for Hybrid System Component Verification
https://link.springer.com/chapter/10.1007%2F978-3-662-54494-5_8

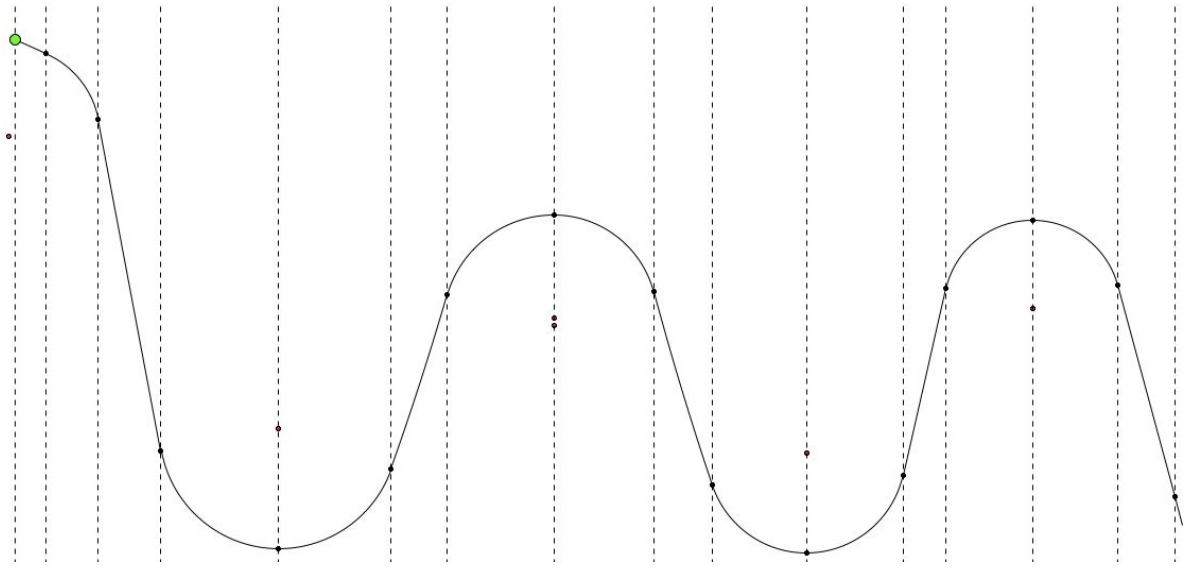
[3] A Complete Axiomatization of Quantified Differential Dynamic Logic for Distributed Hybrid Systems
http://www.lmcs-online.org/ojs/viewarticle.php?id=820

Figure 1: Wooden coasters have long planar sections

A train will therefore be modeled as a point mass with position (x, y), with gravity pointing in the downward direction, and speed v.

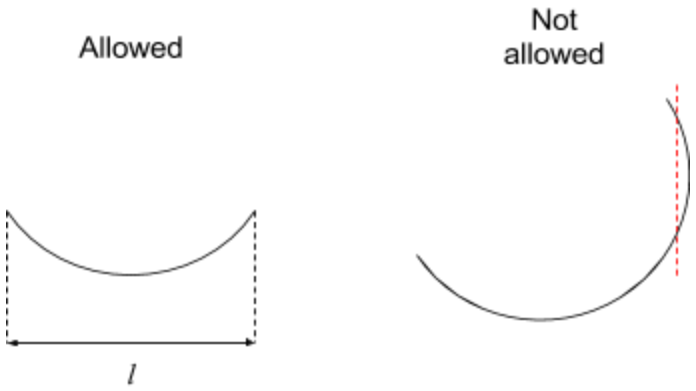
One insight that makes proving properties about coasters tractable is that they can be analyzed in components. We can break down a roller coaster track into components and prove the safety of components individually in order to prove the safety of the overall coaster. We can do this because a roller coaster is linear and the postconditions of any track section are the preconditions of the next one. In this manner, we can piece various track sections together to form the full track.

In particular, we model all track sections as either straight lines, arcs or parabolas. Just having these track sections gives us a surprising amount of expressive power for coasters. For instances, the middle section of the roller coaster in the Figure 1 can be modeled as a composition of straight lines and arcs:



Track sections will be modeled as equations of lines involving x and y. For instance, for a straight section of track, it can be modeled with $y = mx + c$. For an arc, we can use $(cx - x)^2 + (cy - y)^2 = r^2$.

One more assumption we make in modeling is that track sections never overlap in the y-direction. This means that track sections should not loop back over themselves. For lines of the form $y = mx + c$ and parabolas of the form $y = ax^2 + bx + c$, this is already the case. For arcs we must be careful to not allow the track section to go back on itself.



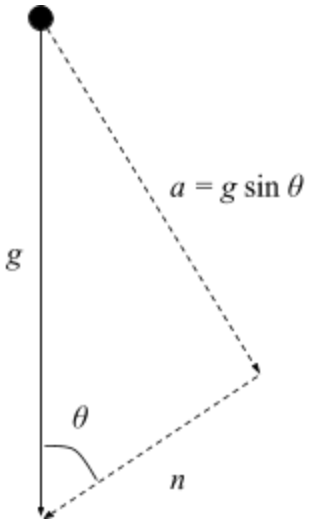
This restriction allows us to express that track sections have a fixed length that is captured by a variable l representing the extent of the track in the x-direction. A nice consequence of this is that each x-coordinate corresponds to a unique y-coordinate.

Modeling

In the this section, we shall show how we derived the continuous dynamics for the different models based off the mathematical equations that described their shape. This method of coming with the dynamics suggest that we should at least be able to prove basic properties such as the coaster never leaving the track.

Straight track

A straight track can be described with the equation of a line $y = mx + c$. The acceleration of a point mass on a slope is given by the component of gravity parallel to the slope:



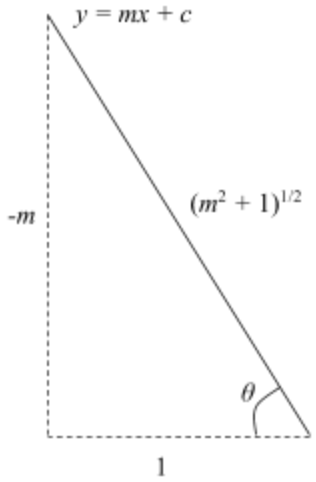
This is given by

$$a = g \sin \theta \qquad (1)$$

where θ is the angle between the slope and the horizontal.

$\sin \theta$ can be related to the slope of the line m:

$$\sin \theta = \frac{-m}{\sqrt{m^2+1}} \qquad (2)$$



Combining (1) and (2), we get:

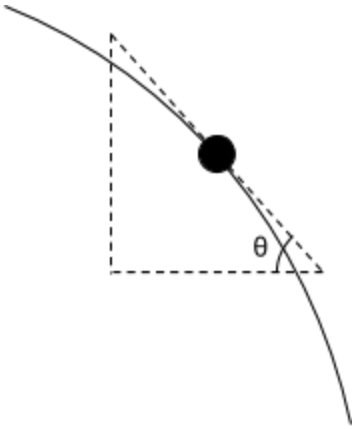
$$a = \frac{-gm}{\sqrt{m^2+1}}$$

The overall dynamics for a straight track is

```
{x′ = v * dx, y′ = v * dy, v′ = -m/((m^2+1)^(1/2))}
```

where $[dx, dy]^T$ is the unit vector representing the direction of motion of the train, such that $dy/dx = m$ and $dx > 0$. Since this is a straight track, the direction vector does not change, so dx and dy stay constant. The multiplicative factor of g has been omitted from the definition of v' due to our assumption of unit gravity.

Arc track



An arc track can be described with the equation of a circle

$$(cx - x)^2 + (cy - y)^2 = r^2$$

where (cx , cy) is the centre of the arc and r is its radius. The physics of an arc track section are not much more sophisticated than that of a straight track. One only needs to consider the instantaneous slope to figure out the acceleration. The expression for acceleration $a = -gm / (m^2 + 1)^{1/2}$ still applies, except that now m changes as well. m is the instantaneous gradient. Considering the top half of the circle, we have y in terms of x:

$$y = \sqrt{r^2 - (x - cx)^2} + cy$$

The instantaneous gradient is given by

$$m = \frac{dy}{dx} = \frac{-(x-cx)}{\sqrt{(r^2 - (x - cx)^2)}}$$

Substituting this into the expression for acceleration, we get:

$$a = \frac{g(x-cx)}{r}$$

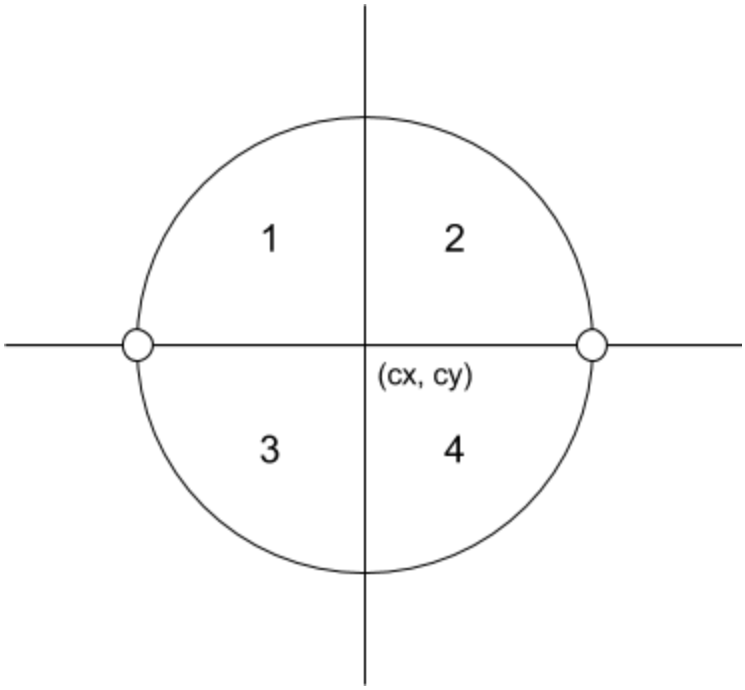
So the overall dynamics for an arc track (clockwise) is

```
{x′ = v*(y-cy)/r, y′ = -v*(x-cx)/r, v′ = (x-cx)/r}
```

The dynamics are similar for counterclockwise motion on an arc:

```
{x′ = -v*(y-cy)/r, y′ = v*(x-cx)/r, v′ = (cx-x)/r}
```

As with the straight line model, the multiplicative factor of g has been omitted from the definitions of v' due to our assumption of unit gravity.



Our proofs for the arc track involve us splitting the circle up into 4 distinct quadrants. The figure above illustrates how the quadrants are labelled. Note that we have open points at $(-r,\ 0)$ and $(r,\ 0)$. We have these restrictions as we do not allow the coaster to be at those points as they dictate that the adjacent section of the track to be completely vertical, something that we do not have in our model.

We split the arc into 4 distinct quadrants as this gives us more precision, allowing us to model more tracks based on where the arc starts and stops.

Parabolic track

The dynamics for parabolic track sections came from the realization that the dynamics of arbitrary curved motion are the same as those of circular motion, except with a changing instantaneous radius. Given a curve of the form $y = f(x)$, the instantaneous radius is given by

$$r = \frac{1+(f'(x)^2)^{3/2}}{f''(x)}$$ [4]

Where $f'(x)$ and $f''(x)$ are the first and second derivatives of y with respect to x respectively.

For a parabola of the form $y = f(x) = ax^2 + bx + c$, we have

$$f'(x) = 2ax + b$$
$$f''(x) = 2a$$

Then the instantaneous radius on a parabola is given by

$$r = \frac{1+((2ax+b)^2)^{3/2}}{2a}$$

The acceleration on a parabola can be derived in a similar manner to that for arc track by considering the instantaneous gradient:

$$a = \frac{-gf'(x)}{\sqrt{f'(x)^2+1}} = \frac{-(2ax+b)}{\sqrt{(2ax+b)^2+1}}$$

So a parabolic track can be modeled with these dynamics (under the assumption of unit gravity):

```
                {x' = v * dx,
                 y' = v * dy,
dx' = -v * dy * (2*a / (1 + (2*a*x + b)^2)^(3/2)),
dy' =  v * dx * (2*a / (1 + (2*a*x + b)^2)^(3/2)),
 v' =  -(2*a*x + b) / ((2*a*x + b)^2 + 1)^(1/2)}
```

The method described here allows us to model the dynamics on any curve of the form $y = f(x)$, not just parabolas, as long as the first- and second-order derivatives exist. For instance, one could model cubic curves if they wanted to.

Evolution Domain

For our models, we also have the concept of an evolution domain constraint, which is a boolean formula representing the region in which the specified continuous dynamics should hold. As we are modeling tracks defined by some mathematical function from which we derive our dynamics, we let the domain constraint be the section of the track which is described by that function, as specified by its x-coordinate. We hence introduce the variables x_0 and x_{end}, which denote the start and end of the track segment. We previously mentioned that we can characterize the length of each track section by a variable, l, and thus we have that $x_{end} = x_0 + l$. The continuous dynamics for our models now look like this:

[4] http://mathworld.wolfram.com/RadiusofCurvature.html

```
{x = f(x) & x0 <= x <= xend}
```

where $x = f(x)$ is the continuous dynamics dependent on the segment of the track under consideration. Note that this makes sense as an evolution domain constraint as it corresponds to a physical section of the track on which the train is traveling and hence the specified dynamics should only hold while the train is on that portion of the track.

Desired properties

Let us consider what it means for a coaster to be safe. There are several conditions that a roller coaster has to fulfill to be considered safe. Recall that we denote v, x, y to be the velocity, horizontal coordinate and vertical coordinate of the coaster respectively. For arc tracks, we denote (cx, cy) to be the centre of the circle the train is travelling on.

Energy is conserved

Since we want these coaster models to be as realistic as possible within the assumptions chosen, we would like to prove that its motion obeys the laws of physics. One of such laws that we seek to prove is law of conservation of energy. Since we neglect resistive forces in our assumptions, the coaster only has kinetic energy and gravitational potential energy. We thus have the following condition:

$$v^2 + 2y = v_0 + 2y_0$$

where v_0 and y_0 are the initial velocities and vertical coordinate of the system respectively.

Positive velocity

It is crucial that the trains only move in the forward direction on a track as moving backwards could have serious implications. For example, in the case of multiple trains on track at the same time, a train moving backwards could lead to an unwanted collision, putting lives at risk. We hence want the velocity to be strictly positive at all times:

$$v > 0$$

Train stays on track

Something that is less intuitive but still equally important to safety is that trains should remain on the track. That is, we want our dynamics to correctly model the motion of the coaster. If our model is correct, then trains should stay on the coaster as expected. For straight lines, we have:

$$y = mx + c$$

For arcs:

$$(cx - x)^2 + (cy - y)^2 = r^2$$

Train remains in its defined region

This is a property that is specific to arc tracks. Since we split up the arc into 4 distinct quadrants, we want to ensure that throughout the duration of evolution, the train remains in its specified quadrant. For instance, for a train evolving in the first quadrant, we have:

$x \leq cx \wedge cy < y$

Train reaches end of track
We also want to show that trains do reach the end of the track, because we don't want our riders to be stuck on the coaster. For straight tracks, this can be expressed as:

$x = x_0 + l$ or $x = x_{end}$

This formula states that there is a way for physics to evolve so that the coaster reaches the end of the track section. Since the only non-determinism in our model is in time, if there is a way for the train to reach the end of the track, then there is a time at which the coaster will be at the end of the track, so the coaster reaches the end of the track eventually.

Relating position and velocity
We would also like to know the relationship between the velocity and the position of the train. This is especially important as it would be vital to know that the train is not travelling too fast at some parts of the track, such as at the end. However, this property follows from our previous properties of strictly positive velocity and conservation of energy. Since we have the properties that $v > 0$ and $v^2 + 2y = v_0^2 + 2y_0$, we get that $v = \sqrt{v_0^2 + 2y_0 - 2y}$, which perfectly relates the velocity to the vertical coordinate. Since for each segment of the track we have unique *y*-coordinates, this property uniquely identifies the velocity of the train based on its position on the track. Hence, we get this property as a result of proving the previous properties.

Combining tracks
Once we identified the different individual track sections and proved properties about them, we are now ready to prove properties about the coaster as a whole. We model this by having a program that contains the continuous dynamics for all the different sections of the track that we have identified, accompanied by their appropriate evolution domain constraints. The decision about which dynamics to run is dependant on the position of the coaster and which evolution domain constraint it falls in. The safety properties for this combined model will be very similar to those of the individual model with the slight exception that for the property that we remain on the track, we have to case on the position of the train to determine which mathematical function it should be described by.

We then expect that we can separate the proof into separate branches and then use the same proof steps applied for the individual models to complete the proof.

Results
Preconditions
We aim to prove the desired properties given some initial conditions of the system. While some of initial conditions that each track segment requires varies, there are some constant conditions that we require of all tracks. For instance, we require a strictly positive initial velocity ($v_0 > 0$), a strictly positive length of track ($x_0 < x_{end} = x_0 + l$) and that the train initially remains on the track. For track segments where the train will end up at a greater *y*-coordinate than it initially started out at, we also require that the train has more than enough velocity such that it can reach the highest point ($v_0^2 > 2y_{peak} - 2y_0$) where y_{peak} is the highest *y*-coordinate the train will ever end up at throughout the duration of its evolution. Other preconditions include defining the bounds for the endpoints of the track segment based on the initial

coordinates and other physical properties of the track such as the centre of the circle for the arc case. As an example, we have reproduced the initial conditions required for a straight line track:

$$y_0 = mx_0 + c \land \frac{dy}{dx} = m \land dx^2 + dy^2 = 1 \land dx = 0 \land v_0 > 0 \land \frac{{v_0}^2}{2} > ml \land l > 0$$

Sample Model

Combining the ideas of preconditions, continuous dynamics, evolution domain and desired properties, we seek to prove something of the following form:

$$Pre \rightarrow [\alpha \,\&\, Q]Post$$

The above statement means that if we have some preconditions are specified by Pre and we allow the system to run according to the continuous dynamics specified in α for any amount of time as long as Q is satisfied, then all the desired properties in $Post$ hold. For instance, we have the following statement we would like to prove about straight line tracks:

```
          (y0 = m*x0 + c &      /* is on track initially */
       dy/dx = m &      /* correct initial direction vector */
              dx^2 + dy^2 = 1 &     /* unit vector */
 sinTheta = -m / ((m^2 + 1)^(1/2)) &   /* captures value of sinTheta
                     in terms of gradient m */
            x = x0 &   /* initial position and velocity */
            y = y0 &   /* initial position and velocity */
            v = v0 &   /* initial position and velocity */
             v0 > 0 &   /* positive velocity initially */
           dx > 0 &   /* travelling rightwards initially */
(v0^2)/2 > m*l &      /* coaster has enough initial kinetic energy to
                        reach end of track */
         l > 0 /* length of track is strictly positive */)
                                ->
                                 [
          {x' = v*dx, y' = v*dy, v' = sinTheta & x <= x0 + l}
          ](v > 0 & y = m*x + c & v^2 + 2*y = v0^2 + 2*y0)
```

Background

Our proofs utilize the tool KeymaeraX and thus the following section relies on the reader being able to understand some basic concepts of KeymaeraX and dL, the framework we use. Of particular relevance is the use of the differential invariants (DI) rule. The basic intuition of this rule is that if a relation holds between two bound variables of real numbers before the running of any ordinary differential equation (ODE) and we want to show that the same relation holds after the ODE, we just need to show that their rates of change fulfill some other condition. This is an especially useful rule to use, as it removes the need to solve the ODE, which can be a potentially complex endeavour for some systems. The identification and proving of key invariants turned out to be of vital importance in the proofs of our models, as they show important relationships between different parts of our system, such as energy and position, at all times.

We shall first discuss the proofs for the individual models (tracks defined by a single function) before first moving on to talking about proofs for combined models (tracks defined by more than one function).

Straight Line

For the case of the straight line track, we split up the safety properties based on which proof rules they would employ.

We surmise that the properties regarding staying on the track ($y = mx + c$) and energy being conserved ($v^2 + 2y = v_0^2 + 2y_0$) is something that should stay true regardless of the gradient of the slope, should we get the dynamics right. Thus it is expected that those two properties should prove via DI.

However, the property that velocity stays strictly positive ($v > 0$) is one that becomes less true in the case where we have a positive gradient and the direction of the acceleration is opposite to the direction of the movement of the coaster. Hence, DI would not work in this case. We note that the dynamics of this model is simple enough to solve for the solution to this particular set of differential equations then allow real arithmetic to figure it out.

We input this model into KeymaeraX and those predictions proved to be true. The first two properties proved directly with DI whereas the last one proved after solving for the solution to the ODE.

In addition, we proved for the straight line case that following the dynamics described by the model, the coaster can indeed reach the end of the track. Proving this just required solving the ODE and then relying on real arithmetic to show that there is indeed a length of time that the ODE can evolve for such that the coaster hits the end.

Arc Track

As previously mentioned in the modeling section, we broke up the analysis of arc tracks into 4 separate cases, one for each of the quadrants. Due to the different preconditions that are defined by the quadrants, we had to formulate different proof strategies for each of the different arcs that we were in. As it turned out, two out of the four were relatively long and complex whereas the other two were relatively simple. However, the safety properties of staying on the circle ($(cx - x)^2 + (cy - y)^2 = r^2$) and energy being conserved ($v^2 + 2y = v_0^2 + 2y_0$) throughout were proven via a simple DI for all four models. The discussion below thus assumes that these two properties have been proven.

A useful insight gained in the proving of these models is that some safety properties are actually really useful in proving other safety properties, and hence cutting these useful properties in earlier so as to have access to them in proving others, before going on to prove them, turned out to be an effective proof strategy. One of these central safety properties that allow us to prove others is that velocity is strictly positive throughout the duration of the coaster.

For the sake of brevity, we shall describe the proof strategy for only two quadrants, the third and the first. The proofs for the other two quadrants are very similar to these two and hence we choose to leave them out.

Third Quadrant

Since the only variables that affect the acceleration of the coaster is its x position relative to the centre of the circle, and we have domain constraints that indicate that in the third quadrant, the coaster has a non-negative acceleration, we can easily prove that velocity in this quadrants is safe via DI. Once we have this safe velocity, the condition that we remain within the third quadrant ($x \leq cx \wedge y < cy$) then proves via DI.

First Quadrant
In order to prove that the velocity is strictly positive, one might be tempted to apply the DI rule. However, in the first quadrant, the direction of acceleration is always opposite to the direction of travel and hence the application of the DI rule would fail. Instead, we use a differential auxiliary ("ghosts") to prove that although this condition gets less and less true as physics evolves, given the right preconditions, it will still remain true throughout. This auxiliary first requires showing that $v \neq 0$, which we prove by showing the equivalent $v^2 \neq 0$. This can be proven by the previously shown results of energy conservation which gave us that $v^2 = v_0{}^2 + 2y_0 - 2y$. Given that we have as a precondition $v_0{}^2 > 2y_{end} - 2y_0$, we just need to prove $y \leq y_{end}$ which we get after arithmetic simplification of the mathematical expression of staying on the circle of the circle and the domain constraint.

The conditions of remaining within the first quadrant ($x \leq cx \wedge cy < y$) were proven easily by taking the domain constraint and applying some transitivity rules to arrive at the desired conditions as well as using the strictly positive velocity previously proven as a differential cut to and then applying DI.

Proving that the coaster eventually reaches the end of the track ($x = x_{end}$) was significantly harder for arc tracks that it was for straight lines. This was due to the difficulty of finding a solution to the ODEs for arcs and then substituting in the appropriate time that makes condition true, which was the proof strategy adopted for straight lines. We could not come up with a valid proof strategy to prove this property for arcs by the end of this project.

Parabolic Tracks
Although we aimed to model parabolic tracks, the physics turned out to be significantly challenging, to the extent that simple properties such as the coaster staying on the track, was nontrivial, even for KeymaeraX. Hence, we decided to forgo modeling tracks described by such an equation. While this sacrificed some expressive power, we reasoned that most coasters can be reasonably modeled with a mixture of just straight lines and arcs and hence it was acceptable to focus on those two cases.

Combined Track
After successfully proving properties about individual models, we moved on to proving properties about combined models of tracks. However, we realized that once we broke down the model into branches that comprised only of an individual track segment (segment described by just one mathematical function), then the proof steps were very similar to the ones we applied for the models of just a single function. For that reason, we omit the proof strategy here.

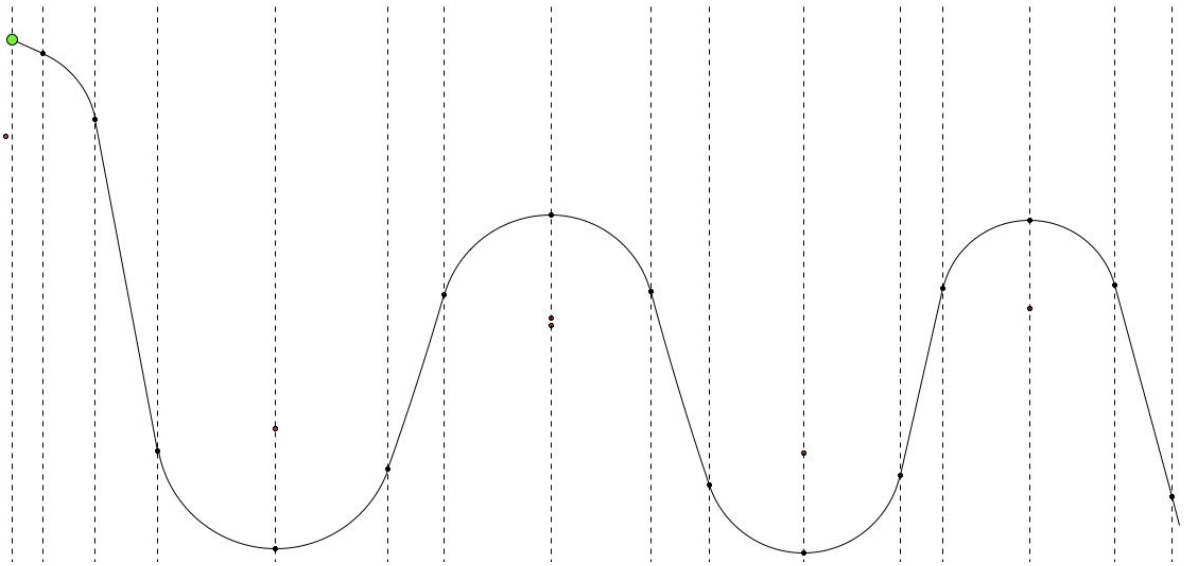
One thing to note is our progress on proving properties about combined tracks were limited by KeymaeraX. As mentioned above, the proof of straight line tracks requires the ODE solver and the proofs about the arc tracks require the use of differential auxiliaries. The latest version of KeymaeraX, while it supports the use of auxiliaries, is unable to solve the ODE for straight line tracks whereas previous versions of KeymaeraX that are able to solve the ODE for straight line tracks do not support the use of the auxiliary rule. Hence, this meant that we were not fully able to prove properties of combined coasters with a single version of KeymaeraX.

We settled on loading the same model in both versions of KeymaeraX, and proving as much as we can with that particular version. With the new version, we were able to prove all desired properties of the track segments described by an arc and with the old version, we could prove all desired properties of the

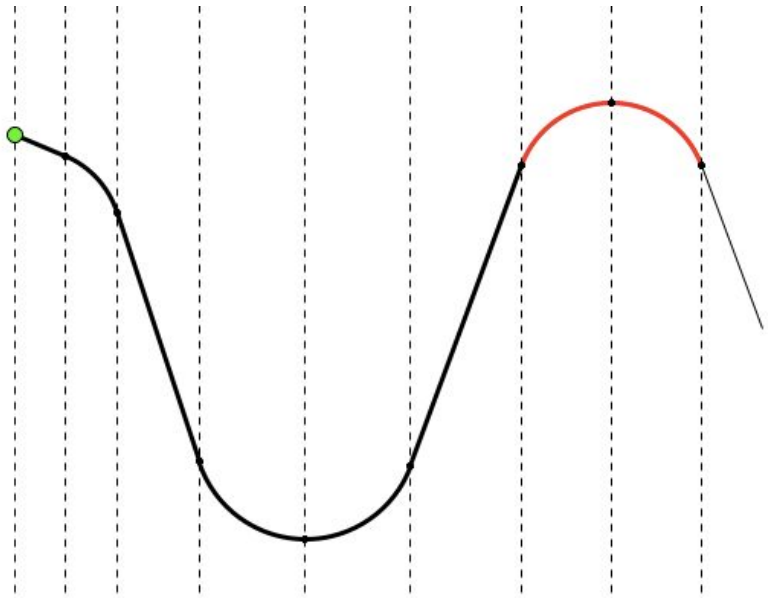
track segments described by a straight line. This leaves the only open branches yet to be proven to those of straight line tracks for the latest version and arc tracks for the older one. We hope that together, they show that we have a complete proof of the desired properties of this combined track, hindered only by the tool of choice.

RollerCoaster Tycoon X

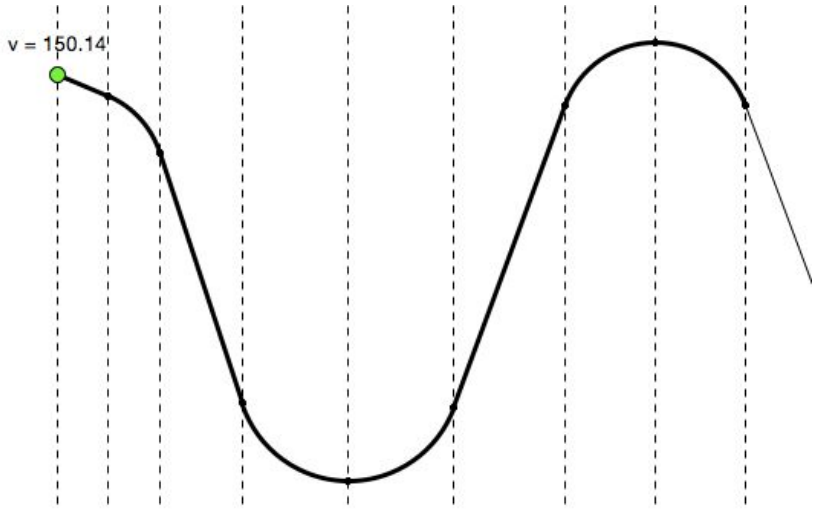
Now that we have the conditions required for a particular track section to be safe, we can write a tool to design safe roller coasters. To this end, we created RollerCoaster Tycoon X, a graphical interface for designing roller coasters using the Python Tkinter library.



Users are able to switch between placing a straight and an arc track by pressing the arrow keys. RCTX only permits allowable track sections to be placed (e.g. track sections are not allowed to loop back over themselves, etc.) For each new track section placed, RCTX checks whether the safety conditions are met. It highlights the track in red if it is unsafe. For instance, the track section highlighted in red here is unsafe because the train will roll backwards in that section.



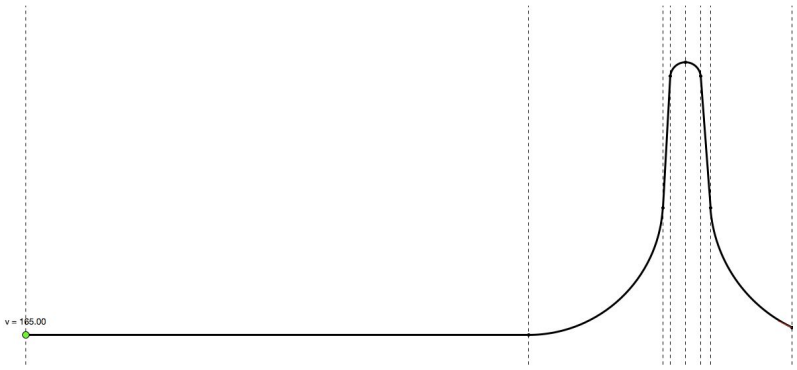
Users can remove unsafe track sections by pressing the Backspace key. Users can also decrease or increase the initial speed of the coaster by pressing the "-" and "=" keys. In this manner we can simulate coasters that are launched at the start of the track.



Finally, RollerCoaster Tycoon X has simulation capabilities to visualize the evolution of the system over time. The user presses the Spacebar to begin the simulation. However, as with all numerical integrations, it is subject to error and should not be used to conclude that a track is safe. Only if the safety conditions are satisfied is the track safe.

Modeling a real roller coaster

With RollerCoaster Tycoon X, we modeled the *Top Thrill Dragster* roller coaster in Cedar Point, Ohio. The *Top Thrill Dragster* consists of a horizontal launch at 120 mph, followed by a vertical climb over a 420-foot arc, and finally coming back down to another horizontal section.



We modeled the *Top Thrill Dragster* in our tool following to its dimensions, proving that it is safe (note how all track sections are black).

Future Work

While we have managed to prove significant properties regarding coasters modeled with the appropriate preconditions and assumptions, there are still a lot of avenues that this work can carry on in.

Concrete proof of combined model

As previously mentioned, we were unable to fully prove properties of tracks that are described by more than one function due to limitations in our choice of tools. While we were able to fully prove all the desired properties for all individual segments of tracks, this is unfortunately not the most useful proof as real roller coasters are definitely modeled by more than one function. Hence, we would like to be able to concretely prove the properties of combined models once our tool allows us to do so.

Modeling more track sections
The continuous dynamics of parabolic motion was too complex for us to be able to prove properties about it and we shifted our focus to just straight lines and arcs. However, we would like to look into proving desired properties about track sections that can be modeled by other functions as well so as to extend the expressive power of our models.

Eliminating assumptions
We made some underlying assumptions that simplified our models and reduced the complexity. The foremost of these was the absence of friction and other resistive forces. This is definitely not a realistic assumption and we would hence like to somehow incorporate friction into our model and prove safety properties.

Multiple trains
As mentioned in the Related Work section, real roller coasters have multiple trains running at once. An important safety condition for roller coasters is that these trains do not collide. Proving this involves modeling and proving the safety of the distributed hybrid system. It would also probably involve proving safety properties involving time.

Analyzing forces on riders
Safety conditions we haven't considered include that the forces on riders are not too great. Future work should analyze and prove that, at all times, the forces exerted on riders do not exceed certain thresholds.

Extending to three dimensions
In order for our models and proofs to be really practical, they need to be extended to the three dimensional world we live in. This means considering rotational physics and relaxing the point mass assumption - trains should have a non-zero volume.

RollerCoaster Tycoon X
As we increase the complexity of the model, RollerCoaster Tycoon X should increase in commensurate complexity. For instance, if we have safety properties for parabolic track sections, we can allow users to add that track type as well. If we safety properties for including friction in our model, RCTX should account for that too.

It would also be nice for RCTX to be able to output .kyx model files for the roller coasters that users have modeled, as well as tactics to automatically prove that the roller coasters are safe.

Conclusion
This project has managed to develop a model for roller coasters and identify a strategy to prove their safety under certain preconditions. We strategized to prove safety of a large coaster by first dividing it up into modular sections and then proving their safety individually. We then sought to prove a combined model comprised of multiple segments but were prevented from doing so due to unforeseen circumstances.

We also managed to develop a UI tool, based on the models we proved, that allows the user to design a roller coaster and will inform the user if the coaster is safe.

It our hope that roller coaster designers will make use of our work as a starting point for verifying that the roller coasters they design are safe. As emphasized at the beginning, the safety of roller coasters is of utmost importance in the interest of the preservation of human life. Thus, roller coasters not be considered safe just by running simulations. They should be provably safe, which we have taken some steps towards in this project.

Distribution of work

Equal work was performed by both project members

Appendix

Index of files

File name	Description
models/1stqcircular.kyx	Model for circular movement in the first quadrant
models/1stqcircular.kya	Proof for circular movement in the first quadrant
models/2ndqcircular.kyx	Model for circular movement in the second quadrant
models/2ndqcircular.kya	Proof for circular movement in the second quadrant
models/3rdqcircular.kyx	Model for circular movement in the third quadrant
models/3rdqcircular.kya	Proof for circular movement in the third quadrant
models/4thqcircular.kyx	Model for circular movement in the fourth quadrant
models/4thqcircular.kya	Proof for circular movement in the fourth quadrant
models/clockwisecircular.kyx	Model for general circular movement in the clockwise direction
models/clockwisecircular.kya	Proof for general circular movement in the clockwise direction
models/counterclockwisecircular.kyx	Model for general circular movement in the counter clockwise direction
models/counterclockwisecircular.kya	Proof for general circular movement in the counter clockwise direction
models/combinedcoaster.kyx	Model for combined coaster describing more than one function

models/combinedcoaster.kya	Partial proof for combined coaster describing more than one function
models/straightcoaster.kyx	Model for general straight line track
models/straightcoaster.kya	Proof for general straight line track
models/straightcoasterreachend.kyx	Model for straight line track that reaches the end
models/straightcoasterreachend.kya	Proof for general straight line track that reaches the end
RollerCoasterTycoonX.py	RollerCoaster Tycoon X application
coasters/TopThrillDragster.rctx	Top Thrill Dragster model file (open from within RollerCoaster Tycoon X)