Cheat Codes for Hybrid Games

Rachel Holladay (rmh@andrew.cmu.edu)

May 9, 2017

Abstract

Given the expressive power of hybrid games, our goal is to explore how easy it is to model realistic games and what insights can be gained from these models. We explore three children's games (Red Rover, Leap Frog and Red Light, Green Light) to provide examples of interesting multi-player, sometimes collaborative games. Throughout the modeling process we show what is easy to model and what is difficult. From each of our games, we were able to dig deeper into the connections made by hybrid games.

1 Introduction

Hybrid systems can be used model cyber-physical systems with discrete and continuous dynamics [1]. By introducing hybrid games, we can additionally model adversarial systems where multiple systems are have different, and possibly conflicting, goals [2, 3].

The flexibility of hybrid games allows us to accurately model the more complex, trickier scenarios that our cyber-physical systems could come in contact with. By allowing for the adversarial component we do not lose any of the gains made by work done on hybrid systems.

In fact, the differential game logic for hybrid games is strictly more expressive then the differential dynamic logic for hybrid systems [3]. This means that everything we can express with hybrid systems we can also express with hybrid games by syntactic embedding. Furthermore, hybrid games can express almost anything. More specifically they can express anything that can be inductively or co-inductively or mixed way be defined over the augmented structures of the reals, $\mathbb{R}$ [3].

In order to truly understand hybrid games, this paper aims to model real world scenarios via hybrid games and examine the modeling challenges that arise. In particular we model multi-player and collaborative games, since they push into an interesting domain. It is important to note that while we discuss and document modeling challenges, these games are expressible by the language, as described above. We do not focus on what is possible but rather what is easily and conveniently expressible.

In choosing what situations to model with hybrid games, we decided to look no further than the playground and model three classic children's games: 'Red Rover', 'Red Light, Green Light' and an adaption of 'Leap Frog'. For each game we first provide a brief summary of the game. We then attempt to model each game using the syntax described in Sec. 2.

For some of the games we are able to provide several models, on increasing complexity, and some models better represent the true game play. For each game, we found that modeling raised interesting questions about the nature of hybrid game modeling that we examine in turn.

A summary of our findings and contributions are listed below. Rather than one main insight, we present one insight per game, totaling to three.

Red Rover Red Rover is a multi-player two-team game competitive game. In developing a series of more complex models we see the difficulties of modeling arrive when we do not utilize private, member based information.

Red Light, Green Light Red Light, Green Light presents a two player game where we were able to fairly accurate model the game. By doing so we see how multi-stage models differ in hybrid systems versus hybrid games.

Leap Frog Leap Frog can be viewed a racing game with a collaborative element. Here we explored the challenges in accurately modeling the jumping and how this game connects to the mathematical game of Nim.

We believe that we can better understand and develop hybrid games by exploring real world examples. While we have chosen children's games and toys, we have lifted the machinery of hybrid games past toy examples. By working to develop more realistic models we can push cyber-physical modeling to new heights.

2 Background

In developing this game models throughout this paper we will build off of the the syntax and semantics discussed in [3]. In this section we will briefly summarize the relevant syntax.

We are modeling two-player games, where the players are referred to as 'Angel' and 'Demon'. Exactly one player, i.e. either Angel or Demon, will win a round of any game. Each game is composed of three components: a precondition, a series of statements composing the game, and the post-condition, or winning condition.

Our precondition is conjunction of logical statements that essentially set the stage for the game. They place bounds on required variables, such as guaranteeing that the distance to the finish line is non-zero, and initialize variables, such as setting all players to start at position zero.

Next, we have the heart of our model, a series of statements describing continuous and discrete dynamics. We will next describe the several operators we make use of. During each of these operations, one of our two players has the authority to make choices for non deterministic operators. Subsequent statements are separated by semicolons. Assignment follow as aspect ed, however, we can also assign to a start, $var := *$, where the player in control determines the value that is assigned. We can give the player in charge the choice of executing one of two statements with the choice operator, $\cup$.

We can change the power in control by wrapping any statements in the ^{d} operator. To insure a statement is true, we can force a player to execute a test $?(test)$, where the test is a logical expression. If the test is not true, the player fails the test and their opponent wins the gain. We will revisit this operator to discuss its use in reinforcing winning conditions. In order to execute continuous dynamics, we make sure of ordinary differential equations $(x' = y)$. The player in control determines how long the ODE evolves. We can place domain constraints, Q on our ODE, $(x' = y \& Q)$ to ensure that our ODE can only run while Q is true.

Finally we have a post-condition. In the case of games this corresponds to our winning condition. Our model can have one of two modalities. If our series of statements composing the model is contained within diamonds, <model>, then our winning condition is for Angel. Essentially we are checking if Angel could win the game. Otherwise, we are modeling to check if Demon could win using brackets, [model]. Considering our goal is to model fair games, we design our games such that the modality is largely irrelevant as both players are given equal opportunity to win. Therefore, for consistency, all of our models use the diamond modality and have a winning condition defined with respect to Angel.

In order to create fair games with realistic winning conditions, we make liberal use of tests. To illustrate why, let us consider the simple game of Angel and Demon racing to the finish line. Even using the diamond modality, the game does not only end when Angel decides or when Angel crosses the finish line to win. Instead, the game could end when Demon crosses the finish line. We would force Angel the test whether Demon has not crossed the finish line. If Demon does cross the finish line, Angel would fail this test, correctly allowing Demon to win. In all of our game model we strategically use tests to enforce the ending and therefore winning conditions of the game.

Figure 1: Red Rover presents a multi-player game with interesting dynamics [5]

With this tools we can now model and examine our three games closely and precisely.

3 Red Rover

The first game that we explore is Red Rover. After describing the game we present two models and discuss how the limitations of the language make it difficult to provide a realistic model of Red Rover.

3.1 Game Description

Red Rover is played by two lines of people holding hands (the "Angel Team" and the "Devil Team") that stand a two or three dozen feet apart [4]. The teams alternate turns where the have identical choices. Without the loss of generality, on Angel's turn, the children shout "Red Rover, Red Rover send X right over!" where X is a member of Devil's team. X then runs at the Angel's line and attempts to break the chain, as seen in Fig.1. If they succeed in breaking the line, they pick one of people from the link they broke to join them back on Devil's team. If they fail to break through, they join Angel's team. When one team has only one person, they will try to break the chain of the other team. If they fail, they lose and the game ends.

3.2 Attempt at Basic Model

We try to start off with an incredibly simple model, Model 1. We start off with at least four people playing (Line 1). Since we know the total number of players, we only need to count the number of players on Angel's team, playerCount. The game concludes when Angel has all the players on her team (Line 6). If Angel, at any point, has no players, then Demon must have all the players and Angel will therefore fail a test (Line 5), allowing Demon to win.

In our very simple model we want to leave whether a player pushes through the random chance. By doing so, Angel and Demon's turns become symmetric and we only need to model one. We represent this random change with boolean variable condition. However, in hybrid games we cannot have a random variable and we only have two players. Therefore the world cannot non-deterministically determine the condition. Allowing one of the players to make this choice would result in an incredibly boring game.

Therefore, we find that our simple model is actually unachievable. We next increase of level of realism and complexity in the following section.

3.3 Assuming Perfect Information

Our simple model of Red Rover turned out to be too simple. We therefore need a more complex model of determining if someone breaks through the line.

Model 1 Red Rover (Simple)

```
(teamCount $> 4$)
playerCount := teamCount;
< (
        (?(condition; playerCount += 1) $\cup$ (?($\neg$ condition; playerCount $-=$ 1));
        ?(playerCount > 0);
)*> playerCount = 2$*$teamCount
```

In Model 2 we present one method of determining this. Following its description we will discuss possible extensions and difficult limitations.

We say that each team player has a strength, which for now correlates to both the strength at which they run at the chain (on the offensive) and the strength at which they hold the chain (on the defensive). For each team we keep track of of the strength of of their minimum player, min_a and min_d, and the strength of their maximum player max_a and max_d. We make a simplifying assumption that the strengths of all other players are linearly interpolated between the minimum and maximum values (we will return to this assumption later).

We further assume that at each call, Angel will call over the maximum Demon Player, who will then run through the minimum strength Angel player (Line 4). If the Demon player breaks through we must adjust the team count (Line 5) and adjust for Demon losing its minimum value (Line 6). If ex-Demon player changes the Angel's minimum value, we adjust that as well (Line 8). If player does not break through, we do the symmetric operators for Demon adding someone to their team (Line 10).

We then repeat the same operations (in reverse) for Demon's turn.

One of the simplifying assumptions of Model 2 is that we assume the strategy of the players. We assume that they will call over the strongest player and run through the weakest player.

We can generalize this model to allow more freedom. We will sketch the model, instead of writing out the model explicitly. Instead of assuming that Angel (without the loss of generality) picks the weakest player, we could allow her to pick the ith strongest Demon player. That Demon player could then, instead of picking the weakest Angel link, could pick the jth weakest link.

This allows for a more realistic strategy. The tradeoff comes in the fact that updating our strengths becomes more complex. A simplifying way would be update the minimum and maximum value correspondingly if i or j is the first or last value and otherwise simply treat as though it is also a linearly interpolated point.

While this more complex model allows for freedom within our strategy it still makes two fundamental and limiting assumptions.

We assume that player's strengths are linearly interpolated. While there may exist groups of children for which this is true, it is hard to believe this holds in the general case. This modeling limitation stems from the fact that we are trying to, for ease, capture the individual strengthens through a group metric. Ideally we would keep track of each players strength. In fact, to be even more general we would have two notions of strength: the offensive strength, how hard they can throw themselves at the line, and defensive strength, how well they can withstand someone trying to break through. In order to do so we would need to generalized distributed systems into distributed games [6, 7].

A second, less obvious assumption is that we have assumed both players have perfect information of not only their strengthens but the opponent's strengths. This is not necessarily realistic for humans (or clever robots) so ideally we would like to have hidden information that encodes true strength versus perceived strength. However, hidden information is beyond the scope of this project.

Therefore we see that while Red Rover seems like a fairly simple children's game, modeling it correctly can prove to be quite challenging!

Model 2 Red Rover (Perfect Information)

```
(teamCount $> 4 \wedge max_a > 0 \wedge max_d > 0 \wedge min_a > 0 \wedge min_d > 0 \wedge max_a > min_a \wedge max_d > min_d) \Rightarrow$
playerCount := teamCount;
< (
if $max_a > min_d$ then
    playerCount = playerCount + 1;
    $min_d = (max_a - min_a) * (1/$playerCount$-1)$;
    if $min_d < min_a$ then
        $min_a = min_d$
else
    playerCount = playerCount - 1
    $max_a = (max_a - min_a) * ($playerCount$-2/$playerCount$-1)$;
    if $max_a \geq max_d$ then
        $max_d = max_a$
?(playerCount > 0);

if $max_d > min_a$ then
    playerCount = playerCount - 1;
    $min_a = (max_d - min_d) * (1/$playerCount$-1)$;
    if $min_a < min_d$ then
        $min_d = min_a$
else
    playerCount = playerCount + 1
    $max_d = (max_d - min_d) * ($playerCount$-2/$playerCount$-1)$;
    if $max_d \geq max_a$ then
        $max_a = max_d$
?(playerCount > 0);
)$^*$> playerCount = 2$*$teamCount
```

4 Red Light, Green Light

We next present the children's game "Red Light, Green Light". Following from the last section, we first provide a basic game description. We then present two models, which vary in complexity and allow us to discuss how the complexity of the model effects the complexity of the winning condition. Finally, we use this model to realize that concerns that arise with multi-stage hybrid systems are unimportant in multi-stage hybrid games.

4.1 Game Description

In Red Light, Green Light there is one person who is "It", who stands on one end of the playing field [8]. The rest of the players stand on the other end. The goal of the non-"It" players is to reach the side of the field where the "It" player is standing, as seen in Fig.2. However, the players can only move after the "It" player shouts "Green Light" and if they move after "Red Light" has been shouted, they must reset to the beginning.

4.2 Models

We present two models of Red Light, Green Light, with the first as a simple approximation and the second as a more complex, realistic understanding.

Our first model is given in Model 3. Our precondition (1) is composed of assumptions such as our

Figure 2: Red Light, Green Light is a group game where players attempt to make it to the other side of the field [9].

players start at zero, $p_a = 0 \land p_d = 0$ and our finish line being a nonzero distance away, $FL = 0$. We also set a maximum acceleration, $A = 0$ and timer $T = 0$.

We run in iterations where we first acceleration and then deceleration. Our end condition (Line 17) is that Angel has reached the end. However, we perform checks after each run such that if Demon (or Angel) reach the end (Line 7, Line 8, Line 15, Line 16), then it explodes on Angel (Demon), allowing Demon (Angel) to win.

At the start of each iteration of the game we let Angel and Demon pick their accelerations (Line 3), insuring they are in bounds (Line 4, Line 5). We next have our ODE where we let the players accelerate until one of them crosses the finish line or the maximum time T has elapsed (Line 6).

If neither of them have crossed the finish line, we repeat the process with the player decelerating to a stop (Line 9). By the rules of the game, if at the end of this ODE a player is still moving and they haven't passed the finish line, they are sent back to position 0 (Line 13, Line 14).

Model 3 Red Light, Green Light (Simple)

```
1:  $(FL > 0 \land A > 0 \land T > 0 \land p_a = 0 \land p_d = 0) \Rightarrow$
2:  $< ($
3:      $t := 0; a_a := *; (a_d := *)^d;$
4:      $?(a_a > 0 \land a_a < A);$
5:      $?^d(a_d > 0 \land a_d < A);$
6:      $(p_a' = v_a, v_a' = a_a, p_d' = v_d, v_d' = a_d, t' = 1, t < T \land p_a \leq FL \land p_d \leq FL \land v_a \geq 0 \land v_a \geq 0);$
7:      $?(p_d \leq FL);$
8:      $?^d(p_a \leq FL);$
9:      $t := 0; a_a := *; (a_d := *)^d;$
10:     $?(a_a < 0 \land a_a > -A);$
11:     $?^d(a_d < 0 \land a_d > -A);$
12:     $(p_a' = v_a, v_a' = a_a, p_d' = v_d, v_d' = a_d, t' = 1, t < T \land p_a \leq FL \land p_d \leq FL \land v_a \geq 0 \land v_a \geq 0);$
13:     $(?(p_a < FL \land v_a \neq 0); p_a = 0) \cup ?(p_a < FL \lor v_a = 0);$
14:     $(?^d(p_d < FL \land v_d \neq 0); p_d = 0) \cup ?(p_d < FL \lor v_d = 0);$
15:     $?(p_d \leq FL);$
16:     $?^d(p_a \leq FL);$
17: $)^* > (p_a \geq FL)$
```

While Model 3 models the game fairly well, it does make one simplifying assumption. It assumes that the two players run together, that one is always running while the other is. While this is not terribly problematic for the acceleration case, it is incredibly limiting for the deceleration case. If one player comes to a stop, it halts the ODE, preventing the other player from finishing its run. Additionally this places all of the power within Angel's hands, not allowing Demon to control how long she runs.

Therefore, we break up the ODEs to give the power back to each player. By doing so in Model 4 we involve a much more complicated termination condition.

Most of our set-up remains the same. However, now we have two ODEs where each player evolves

Model 4 Red Light, Green Light (Separate Running)

```
1: $(FL > 0 \land A > 0 \land T > 0 \land p_a = 0 \land p_d = 0) \Rightarrow$
2: $< ($
3:     $t_a := 0; t_d := 0; a_a := *; (a_d := *)^d;$
4:     $?(a_a > 0 \land a_a < A);$
5:     $?^d(a_d > 0 \land a_d < A);$
6:     $(p_a' = v_a, v_a' = a_a, t_a' = 1, t < T \land p_a \leq FL \land v_a \geq 0);$
7:     $(p_d' = v_d, v_d' = a_d, t_d' = 1, t < T \land p_d \leq FL \land v_d \geq 0)^d;$
8:     $?((p_a < FL \land p_b < FL) \lor (p_a > FL \land p_b < FL) \lor (p_a > FL \land p_b > FL \land t_a < t_b));$
9:     $?^d((p_d < FL \land p_a < FL) \lor (p_d > FL \land p_a < FL) \lor (p_d > FL \land p_a > FL \land t_d < t_a));$
10:    $t_a := 0; t_d := 0; a_a := *; (a_d := *)^d;$
11:    $?(a_a > 0 \land a_a < A);$
12:    $?^d(a_d > 0 \land a_d < A);$
13:    $(p_a' = v_a, v_a' = a_a, t_a' = 1, t < T \land p_a \leq FL \land v_a \geq 0);$
14:    $(p_d' = v_d, v_d' = a_d, t_d' = 1, t < T \land p_d \leq FL \land v_d \geq 0)^d;$
15:    $(?(p_a < FL \land v_a \neq 0); p_a = 0) + +?(p_a < FL \lor v_a = 0);$
16:    $(?^d(p_d < FL \land v_d \neq 0); p_d = 0) + +?(p_d < FL \lor v_d = 0);$
17:    $?((p_a < FL \land p_b < FL) \lor (p_a > FL \land p_b < FL) \lor (p_a > FL \land p_b > FL \land t_a < t_b));$
18:    $?^d((p_d < FL \land p_a < FL) \lor (p_d > FL \land p_a < FL) \lor (p_d > FL \land p_a > FL \land t_d < t_a));$
19: $)^* > (p_a \geq FL)$
```

separate and controls their evolution (Line 6, Line 7). By evolving each player separate we create a more complicated termination condition (Line 8, Line 9).

Without the loss of generality, let's consider when Angel has not lost. There are three conditions. The first is that neither Angel nor Demon passed the finish line, hence the game must continue. The second is that Angel passed the finish line but Demon did not, allowing Angel to win. The third is that Angel and Demon both passed the finish line but Angel did so first, making her the winner.

By creating a more realistic model, we significantly increased the termination complexity. However, excitingly, we were able to develop the termination condition for basic races.

4.3 Multi-Stage Game

Both of our models for Red Light, Green Light present an interesting structure. Both of them have a discrete section, a continuous section and then another round of discrete and continuous sections. Therefore this makes our system multi-stage.

Within the context of hybrid systems, multi-stage present an issue. For hybrid systems we want to ensure that our safety property is true at all times. It is necessary but not sufficient to know that we we stopped we were not in a pit of lava. We would also like to know that we were never in a pit of lava. In multi-stage systems, we do not necessarily prove our property is true for all time and instead remove many of the viable cases.

In hybrid systems there are multiple ways to handle this, such as splitting up the model into multiple cases that are proved separately or reconfiguring the model to check for safety conditions after each round of continuous modeling. The still more complex method of handling this is to enable differential temporal logic, which involves a new set of semantics [].

The question becomes, do we need a similar bag of tricks for games? In short, no. In hybrid systems, we need to insure our system is safe at all times. Our post-condition is a safety condition, and possibly also an efficiency condition. In hybrid games, our post-condition is a winning condition. Therefore, we do not need to or want to insure that our winning condition is always true. If fact, to do so would be lose the point of the game. Therefore, it seems as though multi-stage hybrid games do not need to leverage any kind of differential temporal game logic.

Figure 3: Leap Frog, here demonstrated by children, can be interpreted as a two-player collaborative game [11].

5 Leap Frog

We are able to formulate leap frog as a two person game, which we describe in Sec. 5.1. We then present one detailed model and provide the explanation around, particularly our method for modeling the jump. Finally, we conclude by drawing the connection between this version of leap frog and the mathematical game of Nim.

5.1 Game Description

Leap frog is a children's game where people jump over each other's crouched backs [10], as seen in Fig.3. We re-interpret this as a two-person collaborative game. Angel and Demon start at $x = 0$ and their goal is to be the first one to the finish line, at $x = F$. However, they can only move forward by jumping over each other's backs. Angel and Demon alternate symmetric turns. Without the loss of generality, on Angel's turn she decides the acceleration she wants to run at for her running jump. Demon then decides how far to crouch. The faster Angel moves, the further she will go. However, by crouching less Demon limits her distance.

5.2 Model

The most interesting aspect of model leap frog became the choice of how to model the jump. Before detailing this design we first overview the entire model. First, Demon decided how far they want to crouch over (Line 3) and Angel decides how far they want to jump, subject to some constraints discussed later. Next Angel jumps over Demon (Line 7). If by doing so Angel cross the finish line, Angel wins (Line 8). We then repeat this for Demon, where Angel decides how much to crouch, Demon decides how far to go and then Demon to jump over Angel (Line 13). Finally, we check if Demon's jump has pushed them to win (Line 14). The game ends when either Angel or Demon cross the finish line. Similar to in Red Light, Green Light (Sec. 4.3), we have a multistage game.

As mentioned, the most interesting and challenging aspect of this model is accurately modeling the jump. We made the modeling choice to consider our jump as a ellipse centered around the person being jumped over, as seen in Fig.4. For the sake of this explanation, without the loss of generality, we assume Angel is jumping over Demon. We use an ODE to model the ellipse as seen in (Line 7). The test at (Line 9) ensures that Angel runs the ODE long enough to actually jump over Demon by forcing her position to be

Model 5 Leap Frog

```
$(FL > 0 \land a > 0 \land H > 0 \land T > 0 \land x_a = 0 \land x_d = 0 \land y_a = 0 \land y_d = 0 \land yaxis > 0) \Rightarrow$
$< ($
        $t := 0; v := 0; (h := *)^d;$                                                          ▷ Angel's Jump
        $?^d(h > 0 \land (h + yaxis) < H);$
        xaxis := ∗;
        $?(axis + (h + yaxis) \leq H);$
        $(x'_a = -(v/xaxis) * (y - h), y'_a = (v/yaxis) * (x - centerx), v' = a, t' = 1 \& t \leq T \& v \geq 0);$
        $?^d(x_a < FL)$
        $?(x_a > x_d);$
        $t := 0; v := 0; h := *;$                                                               ▷ Demon's Jump
        $?(h > 0 \land (h + yaxis) < H);$
        $?^d(axis + (h + yaxis) \leq H);$
        $(x'_d = -(v/xaxis) * (y - h), y'_d = (v/yaxis) * (x - centerx), v' = a, t' = 1 \& t \leq T \& v \geq 0)^d;$
        $?(x_d < FL); ?^d(x_d > x_a);$
$)^* > (x_a \geq FL)$
```

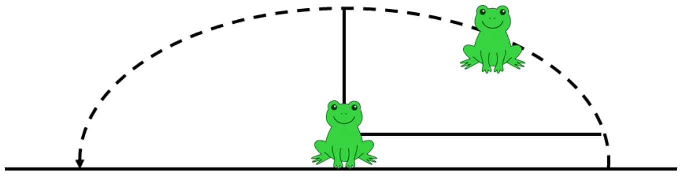

Figure 4: We model our jump as an ellipse centered around the player being jumped.

further long from Demon. The removes the physically impossible case of Angel pausing the jump midair or allowing Angel to not jump, skipping their turn.

In order to use an elliptical motion we have to define four variables: the center of the ellipse, in (x, y), and the length of the x and y axis. We will first discuss the center of the ellipse. With respect to the x coordinate, our jump is centered Demon's x position. The y coordinate the situation is slightly more complicated. The y-height is controlled by how far Demon decides to crouch. Intuitively this makes sense because when we jump over Demon's back, how far they crouch moves the y center of the ellipse.

Having established the center of the ellipse we now discuss the shape, which is determined by the length of the x and y axis. For first consider the y-axis, the height of the ellipse. The y-axis corresponds to tracing out how far Angel moves above Demon's back. We assume the distance between Demon's back and Angel's body is constant since this is determined by the length of each player's arms.

The last variable to specify is the x-axis of the ellipse, essentially the width of the ellipse. The width to the ellipse determines how far the player travels, which is critical to the game play. We can think of how to model the width from three different Angels.

One way to think about it is that in accurately modeling our jump, we want to constrain how far we travel in the x direction based how far we are forced to travel in the y direction.

We can also think of this from the game play perspective. So far Demon has affected Angel's jump based on their height. We would also like the give Angel some level of control over her jump. However, given unlimited control, we would nullify the effect Demon's choice has. Therefore, we must give Angel some choice, constrained by Demon's choice.

Finally, if we think about this game from the perspective, the jumper pushes themselves, which intuitively acts on the x direction.

Therefore, we see that we want the Angel to decide the x-axis with some limits based on Demon's height. Angel, during her jump, is traveling half the circumference of the ellipse. Ideally we would limit the distance traveled by backwards solving for some constant distance. However, as seen in (1), the equation of a circumference is incredible complex.

$$C = 4a \int_0^{\pi/2} \sqrt{1 - e^2 \sin^2(\theta) d\theta} \tag{1}$$

Therefore we are going to approximate our ellipse by saying the sum of the x-axis and the y height is limited to some nonzero value k. We enforce that $k - y > 0$ so that x can always have some non-negative value.

5.3 Connection to NIM

This version of leap frog presents an interesting game that is both collaborative and competitive. The goal of the game is to cross the finish line first. However, to do so you need to work together with the other person, which assists them in winning the game. Therefore, at first you both want to make fast progress towards the goal, making the task collaborative. However, as the players get closer and closer, the task becomes for competitive as you do not want to assist the other player so much that they are allowed to win the game and beat you.

With this formulation of Leap Frog we can draw parallel to an even more fun game: Nim. Nim is a mathematical strategy game played between two people [12]. The game is played with multiple heaps of objects, which we will call stones. At each players turn they can remove some number of stones from a particular heap. They must remove at least one, and there are varying limits on how many they can maximally remove. All the stones removed during a particular player's turn must come from the same stack. There are two variants of this game: the misere play where whoever takes the last object loses, and normal play, where whoever takes the last object wins. What makes Nim a particularly interesting game, aside from its thrilling game play, is that it has been solved for any number of initial heaps and stones. In fact, we can predict which player will win and what winning moves are possible to the player.

We can consider our game of Leap Frog as a normal play game of Nim with one stack where how far you move in the x direction, towards the finish line, corresponds to how many pieces you remove. In this case, we are discretizing the jump distance. The complication comes in that in Nim, each player controls how many pieces it removes versus in our game, both players control how much a particular player jumps. One way around this would be to force the player being jumped over to pick a fixed height, therefore giving the jumper full control over how far they travel. While this removes some of the interesting strategy aspects, it does not severely impact how realistic the model is. Furthermore, by making such an assumption we can apply the winning strategies information from Nim to Leap Frog.

6 Discussion

In this project we examined three children's games: Red Rover, Red Light, Green Light and Leap Frog. For each game we modeled them as hybrid games, examining the extent of our ability to easily and accurately model the games. Each game brought out an interesting element. For Red Rover, we saw how far we can get by modeling the individuals on the teams as team units. Ultimately we saw the limitations with this method and complexities in this seemingly simple game.

We next explored a two player Red Light, Green Light game that allowed us to formally define the winning conditions of a race like game. The setup of our model also showed us that while multi-stage systems can be tricky in hybrid systems, that same difficulty is not true in hybrid games. We concluded with Leap Frog, which we framed as a collaborative game in which two player are leaping over each other to try to cross a finish line. The most challenging aspect of modeling this came from properly defining the jumping procedure, which we did via an ellipse. We were also able to make the connection between this form of leap frog and the mathematical strategy game Nim.

We lifted hybrid games into realistic children's games, thus providing non-toy examples, ironically, through toys. We used each game as chance to probe deeper into modeling and the games themselves.

Acknowledgements

I want to thank Professor Andre Platzer and Brandon Bohrer for their helpful discussion and advice. I greatly appreciate the direction they provided throughout the course of this project. I would also like to thank Brian Okorn for his insights into Red Rover.

Appendix: Notes to Course Staff

In my initial proposal I had expected to model these children games and then develop operators that assist in the modeling process. However, while working, the result of the project turned in an unexpected direction. Even providing basic models proved more challenging that expected and through modeling there were interesting insights. However, none of these insights pointed towards developing a new operator. Overall, the project was more dominated by modeling then what I had initially anticipated. Therefore the contributions of my final projects are quite different then what was suggested in the proposal. Despite this change, I hope that the course staff finds my findings as interesting as I did.

References

[1] A. Platzer, "Logic & proofs for cyber-physical systems," in *International Joint Conference on Automated Reasoning*, pp. 15–21, Springer, 2016.

[2] A. Platzer, "A complete uniform substitution calculus for differential dynamic logic," *Journal of Automated Reasoning*, pp. 1–47, 2016.

[3] A. Platzer, "Differential game logic," *ACM Transactions on Computational Logic (TOCL)*, vol. 17, no. 1, p. 1, 2015.

[4] American Grandparents Association, "How to play red rover," 2015. [Online; accessed April 18, 2017].

[5] Giraffe Boards, "R1: Red rover vs hanabi," 2017. [Online; accessed April 18, 2017].

[6] A. Platzer, "Quantified differential dynamic logic for distributed hybrid systems," in *International Workshop on Computer Science Logic*, pp. 469–483, Springer, 2010.

[7] A. Platzer, "A complete axiomatization of quantified differential dynamic logic for distributed hybrid systems," *arXiv preprint arXiv:1206.3357*, 2012.

[8] Games Kids Play, "Red light / green light." [Online; accessed April 18, 2017].

[9] Icebreaker Ideas, "Red light / green light – game rules and variations," September 21, 2015. [Online; accessed April 18, 2017].

[10] Oxford English Dictionary, "Leap frog." [Online; accessed April 18, 2017].

[11] Flikr, "Leapfrog game 1950s," March 10, 2008. [Online; accessed April 18, 2017].

[12] C. L. Bouton, "Nim, a game with a complete mathematical theory," *The Annals of Mathematics*, vol. 3, no. 1/4, pp. 35–39, 1901.