A CPS Analysis of Pong

Milda Zizyte and Felix Hutchison

December 9, 2014

1 Abstract

To ensure a quality product, it is imperative that video game developers minimize glitches in their games. This paper presents a method to formalize models of the object interactions and physics that video games rely on in order to mathematically prove their correctness. To illustrate this, we prove that the ball in a game of pong does not leave the screen with a series of controller models that increase in complexity. In all but the most complicated case, this is done with an automated proof solver. The last step is proved by hand, and from this we suggest several proof rules to be added to the automated proof software in order to assure automatic formal proofs for games in the future.

2 Introduction

Often, the complexities of video games stem from a need to reproduce some amount of physics in the gaming environment, and have the objects in the game behave consistent to these physics. The market demands quality in the industry, but, often, players still manage to find glitches in the physics implementations of these games because of faulty assumptions made by the developers. As these glitches can make gameplay counterintutive or downright illogical, developers may want some way of ensuring that the objects in their games are consistent with physics. The intent of this project is to provide an illustrative example of how this can be achieved by formalizing the model of the game and using automated proofs to formally ensure that a game behaves according to physics and intuitive play.

The goal of this CPS project is to model and prove the correctness of increasingly more complex paddle controllers in the game of the game of Pong. While Pong itself is not a physical system, it is meant to mimic the dynamics of a game of table tennis. "Pong is a two-dimensional sports game that simulates table tennis. The player controls an in-game paddle by moving it vertically across the left side of the screen, and can compete against either a computer-controlled opponent or another player controlling a second paddle on the opposing side. Players use the paddles to hit a ball back and forth." [1] This gives way to three main objects in a system modeling pong: the ball and the two paddles. The

paddles have vertical motion controlled by an AI or a player, and the ball has physics which respond to it hitting the walls of the arena and the ping pong paddles.

For our formalisms, we use the language of linear differential logic, or $D\mathcal{L}$. The systems are modeled as hybrid programs, where control is dependent on discrete assignment, nondeterministic choice, repetition, test, and ordinary differential equations. Properties are governed by first order logic and modalities. The automated proof solving software KeYmaera provides a framework for creating proofs within this formal system automatically or semi-automatically with manual interaction. We use this software to prove properties of our model, in conjunction to showing several properties by hand.

Using this tool, we prove that our model of pong ensures that the ball obeys physics of the walls, and that following the ball with the paddles keeps the ball on the playing screen. Using non-automated formal proof, we further show that the ball can be kept on the screen by the player by removing strict starting conditions. This last step creates a requirement for proof rules unavailable in KeYmaera, which we prove semantically and argue for the addition of in order to automate further game verification.

3 Related Work

Surprisingly, there are not currently any papers analyzing paddle controllers in Pong, or at least none that we could find. However, there does exist some work exploring how to formalize games. [3] outlines a framework for specifying a formal model of a game before game development, and then auto-generating code based on this model. This method features an Event-B model that defines invariants in these games, and the example game the authors write is a very clear cyberphysical system - a model of a car. This paper shows that some attempts at formalizing games to ensure correct physics have been made, but makes no discussion on the actual formal proofs of the models they use.

At least one other work formally outlines reachability conditions for generated Mario levels [4]. The paper is an example of how formal methods can be used to show playability of a certain game given initial conditions. While the paper makes no mention of hybrid languages for cyberphysical systems, it motivates our work to prove that certain starting configurations of playable.

4 Terminology

As illustrated in Figure 1, we model the screen as a rectangle with upper-left origin and width $Width$ (abbreviated as W) and height $Height$ (abbreviated as H). The boundaries are at the upper and lower walls (black). The ball (red) is a point with location (b_x, b_y) and linear velocity vector (bv_x, bv_y). The player and opponent paddles (blue) are line segments with y-centers P_y and O_y and velocities Pv_y and Ov_y, respectively. Both paddles have length $2 \cdot W_p$.

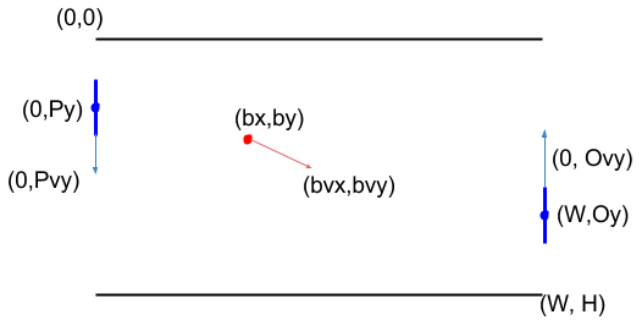

Figure 1: The pong playing screen

The main property were interested in is that of the controlled paddle being always able to return the ball and prevent a point from being scored. Since the opponent paddle is assumed to be perfect, this property can be rephrased as "no point is scored". For an actual measurable property we can instead check the position of ball and ensure it stays out of the scoring box. Since we want this to always hold true, it should be expressed as a safety property (an always). To prove this property in KeYmaera, we express this as the dL formula

$$[\alpha^*]ourScoringBox < ball < opponentScoringBox.$$

We also have the opposite case, where the controller is not sufficient. In this situation, we want to show that a point can be scored. This corresponds to the liveness property

$$\langle\alpha^*\rangle(ball < ourScoringBox \vee ball > opponentScoringBox).$$

5 Outline of goals

The first stepping stone in this project is to get the ball bouncing correctly off the walls. This is mostly just getting the dynamics of α, the differential equations, correct.

Once we've shown the correctness of the ball movement in an empty course, we introduce the player paddles and prove that the ball still remains in the course. We then move on to proving perfect play under these control schema.

6 Methodology

6.1 Ball physics

The control here has the ball move linearly and change direction upon hitting a wall. If it hits a vertical wall, its horizontal velocity flips sign, and if it hits

a horizontal wall, its vertical velocity flips sign. The ball is allowed to start anywhere inside the screen and moves with a nonzero velocity.

We show two properties. First, that the ball always stays inside the court(equation 1). Second, that the velocity of the ball remains constant, even with directional changes (equation 2).

$$\alpha = if(b_x = 0 \& bv_x < 0)\ then\ bv_x = -bv_x;$$
$$...; if(b_y = H \& bv_y > 0)\ then\ bv_y = -bv_y;$$
$$(b'_x = bv_x, b'_y = bv_y \& b_x \geq W \& 0 \leq b_y \leq H) \cup$$
$$... \cup (b'_x = bv_x, b'_y = bv_y \& 0 \leq b_x \leq W \& b_y \geq H)$$

$$\Gamma, bv_x^2 + bv_y^2 = v^2 \rightarrow [\alpha^*]bv_x^2 + bv_y^2 = v^2 \quad (1)$$

$$\Gamma, 0 \leq b_x \leq W \wedge 0 \leq b_y \leq H \rightarrow [\alpha^*]0 \leq b_x \leq W \wedge 0 \leq b_y \leq H \quad (2)$$

In order to prove that the ball doesn't exit, we need to give it the actual possibility of exiting. At the very least, we need an "inner physics" evolution domain to describe the area inside of the screen, and an "outer physics" evolution domain to describe the area outside of the screen. The two evolution domains share a boundary such that it is possible to cross from one evolution domain to the next between repetitions of the control.

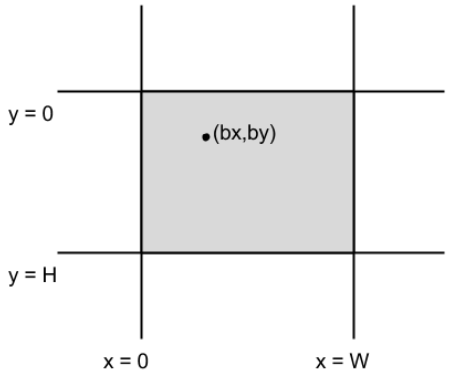

Figure 2: Evolution domains for Step 1

In fact, we use 9 separate evolution domains to describe the areas the ball could go, as delineated by the borders of the screen. Figure 2 illustrates these evolution domains. The gray area corresponds to inner physics and the white areas correspond to outer physics. While first trying to prove these properties, we tried to define upper physics as a single evolution domain

$$b_x \leq 0 \vee b_x \geq W \vee b_y \leq 0 \vee b_y \geq H,$$

where b_x and b_y describe the x and y coordinates of the ball. However, this did not work, as certain branches of our control loop would, for example, restrict only b_x at the boundary and leave b_y free. This is problematic if, say, $b_x = 0$ and b_y is left free, as we did not have the necessary information in our proof that b_y would stay within the required safe range.

This, combined with the branching that results from checking if the ball is at any of the four boundaries, results in a proof that has 2701 branches. This makes sense, as each of the "if" statements adds exponential complexity on the outside of the proof, and each of the nine evolution domains creates further branching within each of the big branches, but is still cumbersome. Thankfully, for the simple model of a ball bouncing off the walls, most of the proof completes automatically, but for a proof that would require manual intervention, this would quickly become cumbersome if not intractable to prove.

In subsequent steps we will be using the back walls as part of our safety conditions, so we no longer need to consider their interactions with the ball. However, we will be introducing the player paddles, which interact with the ball in a similar manner, leaving us with a similar number of differential equations to consider.

6.2 Introduction of Player Paddles

The player paddles are constrained to movement in the y-direction, and have a width $2W_p$ less than the height of the court. If we let the paddles move discontinuously to block the ball every time the ball gets to the paddle movement axis, in an event driven manner, we should be able to demonstrate that the ball always remains between the paddles. After showing this, we can use roughly the same proof to show that no point is ever scored.

The main difference in the behavior is that we have gotten rid of the back walls. Instead we substitute in the behavior of the paddles, where the ball only bounces when it hits the paddle between the paddle's width ($2 \cdot W_p$). If it misses the paddle then it continues straight, which we show as a violation of the safety property $b_x \geq 0$.

$$\alpha = if(b_x = 0 \& bv_x < 0 \& p_y + W_p \geq b_y \geq p_y - W_p) \; then \; bv_x = -bv_x;$$

$$...; if(b_y = H \& bv_y > 0) \; then \; bv_y = -bv_y;$$

$$(b_x' = bv_x, b_y' = bv_y \& b_x \geq W \& 0 \leq b_y \leq H) \cup$$

$$... \cup (b_x' = bv_x, b_y' = bv_y \& 0 \leq b_x \leq W \& b_y \geq H)$$

The other addition we make to our hybrid program is the inclusion of the paddle controllers Player (P_y) and Opponent (O_y). For the purposes of this step, which allows cheating by making sure the paddles are always at the height of the ball, we just use nondeterministic assignment to enforce the position of the paddle.

$$\beta = P_y := b_y; O_y := b_y;$$

$$\Gamma, 0 \leq b_x \leq W \wedge 0 \leq b_y \leq H \rightarrow [(\beta, \alpha)^*]0 \leq b_x \leq W \wedge 0 \leq b_y \leq H \quad (3)$$

Equation 3 proves automatically in KeYmaera, requiring 2236 branches. Note that the number of evolution domains has been reduced to three: the inner physics evolution domain, and the areas directly to the left and to the right of the screen.

6.3 Ball-Follower Paddle

Since a paddle that automatically places itself in the right place isn't very interesting, we will introduce continuous behavior to the paddle that we are writing the control schemes for. This also means we have to pick what control scheme we want to examine. The first one will be a y-position follower. This controller simply tries to keep the paddles y-position to that of the ball. We will show that these dynamics are correct (eq. 4), and then that this controller is capable of perfect play (eq. 5).

$$\beta = Pv_y := bv_y; O_y := b_y;$$

$$\alpha = if(b_x = 0 \& bv_x < 0 \& p_y + W_p \geq b_y \geq p_y - W_p) \ then \ bv_x = -bv_x;$$

$$...; if(b_y = H \& bv_y > 0) then bv_y = -bv_y;$$

$$(b'_x = bv_x, b'_y = bv_y \& b_x \geq W \& 0 \leq b_y \leq P_y) \cup$$

$$... \cup (b'_x = bv_x, b'_y = bv_y \& 0 \leq b_x \leq W \& P_y \geq b_y \geq H)$$

$$\Gamma, P_y = b_y \rightarrow [(\beta; \alpha)^*]P_y = b_y \quad (4)$$

$$\Gamma \rightarrow [(\beta; \alpha)^*]0 \leq b_x \leq W \wedge 0 \leq by \leq H \quad (5)$$

6.3.1 The proof

This is the section where we first notice the importance of event ordering. Now that the player paddle is trying to respond to the ball, it is important that physical events that discretely change the state of the ball (e.g. a wall bounce) occur before the controllers observe the state and react. To fix this, we split α into two parts:

$$\alpha_1 = if(b_x = 0 \& bv_x < 0 \& p_y + W_p \geq b_y \geq p_y - W_p) \ then \ bv_x = -bv_x;$$

$$...; if(b_y = H \& bv_y > 0) \ then \ bv_y = -bv_y;$$

$$\alpha_2 = (b'_x = bv_x, b'_y = bv_y \& b_x \geq W \& 0 \leq b_y \leq P_y) \cup$$
$$... \cup (b'_x = bv_x, b'_y = bv_y \& 0 \leq b_x \leq W \& P_y \geq b_y \geq H)$$

and prove equations 4 and 5 over runs of $(\alpha_1; \beta; \alpha_2)^*$ instead.

These properties then prove automatically, in under 10 minutes of compute time (see appendix A.3). This give us an indication that these techniques may be not only theoretically applicable to analysis of game physics interactions, but practicable too.

6.4 Fixed Speed Paddle

However, a paddle that perfectly matches the ball is relatively uninteresting, so let's add another constraint. Here, the paddle velocity is fixed, and greater than that of the ball. We will show that the controller can still keep the ball above the paddle (eq. 6), that it can always get the ball there, (eq. 7), and that it can play perfectly (eq. 8). To make the controller respond appropriately, we need to add the y-crossing of paddle and ball to the evolution domains.

$$\beta = if(b_y > P_y)\ then\ Pv_y = Pv_{max}$$
$$elseif(b_y < P_y)\ then\ Pv_y = -Pv_{max}; O_y := b_y;$$

$$\alpha = if(b_x = 0 \& bv_x < 0 \& p_y + W_p \geq b_y \geq p_y - W_p)\ then\ bv_x = -bv_x;$$

$$...; if(b_y = H \& bv_y > 0)\ then\ bv_y = -bv_y;$$
$$(b'_x = bv_x, b'_y = bv_y \& b_x \geq W \& 0 \leq b_y \leq P_y) \cup$$
$$... \cup (b'_x = bv_x, b'_y = bv_y \& 0 \leq b_x \leq W \& P_y \leq b_y \leq H)$$

$$\Gamma, |P_y - b_y| < W_p \rightarrow [(\alpha; \beta)^*]|P_y - b_y| < W_p \tag{6}$$
$$\Gamma \rightarrow \langle(\beta; \alpha)^*\rangle|P_y - b_y| < W_p \tag{7}$$
$$\Gamma \rightarrow [(\beta; \alpha)^*]0 \leq b_x \leq W \wedge 0 \leq b_y \leq H \tag{8}$$

6.4.1 Keeping the ball over the paddle

Equation 6 describes the controller ensuring that the ball stays over the paddle. Since this is the differential invariant we use to show the safety property of 8, we will prove these at the same time.

The proof file can be found in appendix A.4 and proves automatically. In some cases, however, manual intervention will greatly speed up the solution. For example, when the ball is at a boundary and interacting with the wall or paddle, it may be necessary to chose an intermediate time (t_0) of 0, for the ODE solve, to give us the information we need for the branch.

$$\text{ODE solve}\ \frac{\forall t \geq 0((\forall 0 \leq t_0 \leq t.[x := \psi(t_0)]H) \rightarrow [x := \psi(t)]\phi)}{[x' = \theta]\phi}$$

Since this term may not arise immediately from the quantifier elimination, it can be beneficial to select this value manually using the *forall left* rule.

$$\begin{array}{r} \forall L\ \dfrac{t \geq 0, [x := \psi(0)]H \vdash [x := \psi(t)]\phi}{t \geq 0, \forall 0 \leq t_0 \leq t.[x := \psi(t_0)]H \vdash [x := \psi(t)]\phi} \\ \rightarrow r\ \dfrac{t \geq 0 \vdash (\forall 0 \leq t_0 \leq t.[x := \psi(t_0)]H) \rightarrow [x := \psi(t)]\phi}{} \\ \forall r\ \dfrac{\forall t \geq 0((\forall 0 \leq t_0 \leq t.[x := \psi(t_0)]H) \rightarrow [x := \psi(t)]\phi)}{} \\ \text{ODE solve}\ \dfrac{}{\vdash [x' = \theta]\phi} \end{array}$$

The property proves in 34,285 steps (1227 of which were manual interventions, to assist when Quantifier Elimination encountered difficulty). Part of the reason for the largeness of this proof is the controller decisions and physical interactions that occur. in this particular proof, there were 6 if-then-else statements, resulting in 27 branches, with 9 possible evolution domains in each branch. This lead to a total of 3846 branches, many of which were similar. This is an example of a situation where lemmas would greatly assist the ease of proof.

6.4.2 Getting the ball over the paddle

While it is useful to know that the paddle control can keep the ball over top of the paddle, if it can't get the ball there then this property is useless. What we'd like to show is that the paddle controller can always get the ball over the paddle no matter where it is when the opponent hits it. This property is expressed as

$\Gamma \rightarrow \langle(\beta;\alpha)^*\rangle\phi$

Where $\phi \equiv |Py - by| \leq P_{width}$ and Γ, β, and α remain the same as for equations 6 and 6. In this case, some formulas may be omitted for brevity, as can be seen in appendix A.5. The justification for these removals is given in the code, though the proof does not suffer if the formulas are not omitted.

Unfortunately, this property is not provable in KeYmaera. The problem comes from the loop. For diamond properties over loops the proof rule needed is convergence.

$$\frac{\Gamma \vdash \exists n.\psi(n) \qquad \forall n > 0.\psi(n) \rightarrow \langle\beta;\alpha\rangle\psi(n-1) \qquad \exists n \leq 0.\psi(n) \rightarrow \phi}{\Gamma \vdash \langle(\beta;\alpha)^*\rangle\phi}$$

The problem is that convergence relies on a strict progress step of 1 every time. While this can be scaled by multiplying n by some δ, we still need a fixed step advancement. This presents an issue when dealing with evolution domains. Since evolution domains restrict how much a system can evolve, it's not always possible to pick a trace through one that progresses by at least δ. For example,

in this case, we can always end up in a situation where $by = \epsilon$ such that ϵ is sufficiently small that the differential equation α can't evolve for long enough before hitting the evolution domain boundary $by \geq 0$ to guarantee progress by δ, for any δ for any ψ.

This means that we need to use new proof rules that can provide us with a way to navigate past the problem. Here we introduce two new rules: Convergence Substitution, and Loop Segmentation.

Convergence Substitution, lets us use a convergence function $\psi(n)$ that doesn't necessarily prove ϕ. To do this, we introduce an easier convergence goal C, which can be proved with $\psi(n)$, and a loop invariant F, which can be used in conjunction with C to prove the original property ϕ. (Soundness of this rule is in appendix B)

$$\text{Convergence Substitution } \frac{\Gamma \vdash [\alpha^*]C \qquad \Gamma \vdash \langle\alpha^*\rangle F \qquad F, C \vdash \phi}{\Gamma \vdash \langle\alpha^*\rangle\phi}$$

The other rule we will introduce is Loop Segmentation. This rule allows us to split a loop into segments of size n. This is useful when you can make guarantees about behavior over n iterations that can't be made about a single one.

$$\text{Loop Segmentation } \frac{\Gamma \vdash \langle(\alpha^n)^*\rangle\phi}{\Gamma \vdash \langle\alpha^*\rangle\phi}$$

Using these two new rules, we will outline the skeleton of a proof for the property that the paddle controller can always get the ball over the paddle. The full proof would contain far to many branches to be practical on paper, due to the if conditions of paddle controller. If the above rules were added to KeYmaera it would be possible to prove this property in a more rigorous manner than outlined here.

$$\text{Convergence Substitution } \frac{\Gamma \vdash [(\beta;\alpha)^*]C \qquad \Gamma \vdash \langle(\beta;\alpha)^*\rangle F \qquad F, C \vdash \phi}{\Gamma \vdash \langle(\beta;\alpha)^*\rangle\phi}$$

Where

$C \equiv \frac{|p_y - b_y|}{H} \leq \frac{b_x}{W}$

$F \equiv b_x \leq 0$

Here we use convergence substitution to let us pick a function that is much easier to prove the convergence of. Since bx is unaffected by bouncing or paddle control, we can more easily prove convergence of $bx \leq 0$. Additionally we introduce the invariant C that is needed to show that ϕ is true. The intuition is that we're using C as a property that will show ϕ under convergence and F as a way of showing progress of convergence.

$$\text{QE}\ \frac{*}{F, C \vdash \phi}$$

This part is trivial, which is good. Selection of C and F should in general be chosen to ensure that this part is trivial.

Next up is proving the invariant C branch. This is all normal proof techniques and will be skipped for brevity (the ifs create a lot of branches). Suffice to say the invariant conditions of Γ, and the occasional differential cut, are sufficient to prove C by differential weaken in the end of every branch.

$$\text{LI}\ \frac{\text{QE}\ \dfrac{*}{\Gamma \vdash C} \qquad \text{DC,DW}\ \dfrac{*}{\text{some formulas} \vdash [\alpha]C}\ \text{;,if, ...}\ \dfrac{}{C \vdash [(\beta;\alpha)]C}}{\Gamma \vdash [(\beta;\alpha)^*]C}$$

More interesting is the convergence branch. If we start trying to prove this as a direct convergence property we'd attempt something like $bx \leq \delta n$. The problem is that we can always have an $by = \epsilon$ such that ϵ is sufficiently small that the differential equation α can't evolve for long enough to guarantee bx can decrease by at least δ after one iteration. For example, if we have $bvy = bvx = 1$, a $by = \frac{\delta}{2}$ makes it impossible for bx to decrease by δ, as we run into the evolution domain constraint $by \geq 0$.

$$\frac{\text{QE}\ \dfrac{*}{\underset{n\in\mathbb{N}}{\exists}\ b_x < \delta n} \qquad \dfrac{\text{Here be dragons}}{\underset{n>0}{\forall}\ b_x < \delta n \rightarrow \langle(\beta;\alpha\rangle b_x < \delta(n-1)} \qquad \text{QE}\ \dfrac{*}{\underset{n\leq 0}{\exists}\ b_x < \delta n \rightarrow x \leq 0}}{\Gamma \vdash \langle(\beta;\alpha)^*\rangle F}$$

In order to prove this property, we need to evolve for two iterations. One to take us to the wall, and then one to evolve for as long as we want (until we hit the opposite wall). After two iterations by is guaranteed to be able to evolve by at least H and thus bx can evolve by at least $bvx|\frac{H}{bvy}|$. This gives us a guarantee of progress.

$$\text{Loop Segmentation}\ \frac{\text{as above} \qquad \dfrac{\dfrac{}{\forall n > 0. b_x < \delta n \rightarrow \langle\beta;\alpha;\beta;\alpha\rangle b_x < \delta(n-1)}\ \text{...}}{\Gamma \vdash \langle(\beta;\alpha;\beta;\alpha)^*\rangle F} \qquad \text{as above}}{\Gamma \vdash \langle(\beta;\alpha)^*\rangle F}$$

The rest follows from these using existing proof rules (by selecting the path that evolves first to the wall and then to the opposite wall). While our discussion of the property ends here, this skeleton of a proof can be fleshed out in KeYmaera by adding the two proof rules discussed above.

7 Conclusions and Future Work

We have shown that it is possible to use dL models to formally prove several main aspects of pong. In particular, the methods we have employed show what initial conditions can guarantee that game dynamics behave in a certain way. In this paper specifically, we have shown how a pong ball can be proven to stay within bounds of the playing screen, that having paddles follow the ball and removing vertical walls preserves this property, that a paddle that follows the ball can prevent the player from losing, given certain initial conditions. Similar formal models can be applied to any other video game.

We acknowledge, however, that this approach is still developmental. In particular, the discussion in section 6.4.2 shows the need for more rules in the KeYmaera tool to assist with automation. However, as progress in this space continues, the needed rules will be naturally discovered, and once they are proved semantically, they can be added to KeYmaera.

Once development of KeYmaera has reach a sufficient point, the next step is to begin trying to prove interactions in actual on market video games. This will involve finding ways to automate the extraction of a model from interaction code, and techniques for invariant generation.

In this paper, we have shown an immensely powerful technique for proving formal properties of game physics, and the applicability of formal proofs to this problem space. With added features to proof automation tools, the ability to automatically prove these properties will also increase. We maintain that formal guarantees are the best way to ensure high quality products, not only by proving that objects in the game obey physics, but by showing that players cannot win and lose in unexpected and unintuitive ways.

References

[1] *Pong*, Wikipedia, http://en.wikipedia.org/wiki/Pong

[2] *Pong*, Game Mechanics Wiki, http://gamemechanics.wikia.com/wiki/Pong

[3] Perchy, S. and Catano, N, *Formal Software Development of Android Games,* http://www.lix.polytechnique.fr/ perchy/home/articles/formal_game.pdf

[4] Adelhart, K. and Kargov, N., *Mario Game Solver*, International Technological University, Masters Thesis, http://130.226.142.164/documents/reports/AdelhartKargov12.pdf

A Key files

A.1 Key file for the simple physcis

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

Name: Milda Zizyte & Felix Hutchison
KeYmaera version: 3.6.12
Backends used (Mathematica, Z3, ...): Mathematica
Nodes: 20617
Branches: 2701
Interactive steps: 52
Arithmetic Solver: 47.723
Time: 1077.339s
Proof completes (Y/N): Y

*/

\functions {
R Width;
R Height;
R vel;
}

\programVariables {
R bx;
R by;
R vx;
R vy;
}

/* Prove that the ball stays in bounds when bouncing off
    the walls */

\problem{

/* INITIAL CONDITIONS */
/* Linear non-zero velocity and non-empty playing space
    with ball inside */

(Width > 0 & Height > 0 & bx > 0 & by > 0 & bx < Width &
    by < Height & vx^2 + vy^2 = vel^2 & vel > 0)
->
\[
```

```
  (
    /* CONTROL */
    /* Bouncing off the walls. We check velocity to avoid
        oscillating
    in control forever */

    (
       if (bx = 0 & vx < 0) then (vx := -vx) fi;
       if (bx = Width & vx > 0) then (vx := -vx) fi;
       if (by = 0 & vy < 0) then (vy := -vy) fi;
       if (by = Height & vy > 0) then (vy := -vy) fi
    );

    /* CONTINUOUS DYNAMICS */
    /* The nine evolution domains correspond to the
       middle area of the
    field and the areas directly above, below, to the
       left, to the right,
    and the four corners outside the field. We need these
    separate domains in order for the proof to work. */

    ({bx' = vx, by' = vy & bx >= 0 & bx <= Width & by >=
       0 & by <= Height} ++
     {bx' = vx, by' = vy & bx <= 0 & by >= 0 & by <=
        Height} ++
     {bx' = vx, by' = vy & bx <= 0 & by <= 0} ++
     {bx' = vx, by' = vy & bx <= 0 & by >= Height} ++
    {bx' = vx, by' = vy & bx >= Width & by >= 0 & by <=
       Height} ++
    {bx' = vx, by' = vy & bx >= Width & by <= 0} ++
    {bx' = vx, by' = vy & bx >= Width & by >= Height} ++
    {bx' = vx, by' = vy & bx >= 0 & bx <= Width & by <=
       0} ++
    {bx' = vx, by' = vy & bx >= 0 & bx <= Width & by >=
       Height}))*@invariant(vx^2 + vy^2 = vel^2 & bx >= 0
        & bx <= Width & by >= 0 & by <= Height)  /* Loop
       Invariant */
\]
(bx >= 0 & bx <= Width & by >= 0 & by <= Height) /*
   Safety Condition */
}
```

A.2 Key file for equations 3

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

```
    Statistics.

Name: Milda Zizyte & Felix Hutchison
KeYmaera version: 3.6.16
Backends used (Mathematica, Z3, ...): Mathematica
Nodes: 18832
Branches: 2236
Interactive steps: 0
Arithmetic Solver: 47.723
Time: 1077.339s
Proof completes (Y/N): Y

*/

\functions {
R Width;
R Height;
R vel;
}

\programVariables {
R bx;
R by;
R vx;
R vy;

R paddleWidth;
//R Px;
R Py;
//R Ox;
R Oy;
}

/* Prove that the paddles following the ball
keeps the ball in the play screen */

\problem{

/* INITIAL CONDITIONS */
(Width > 0 & Height > 0 & bx > 0 & by > 0 & bx < Width &
    by < Height & vx^2 + vy^2 = vel^2 & vel > 0 &
    paddleWidth > 0) /* & Px = 0 & Ox = Width) */
->
\[
  (
    /* CONTROL */
```

```
    Py := by;
    Oy := by;

    /* PHYSICS EVENTS */
    (
    /* Bounce off the paddles */
       if (bx = 0 & by > Py - paddleWidth &
            by < Py + paddleWidth & vx < 0) then (vx := -
               vx) fi;
       if (bx = Width & by > Oy - paddleWidth &
            by < Oy + paddleWidth & vx > 0) then (vx := -
               vx) fi;
    /* Bounce off the top and bottom walls */
       if (by = 0 & vy < 0) then (vy := -vy) fi;
       if (by = Height & vy > 0) then (vy := -vy) fi
    );

    /* CONTINUOUS DYNAMICS */
    /* Evolution domains correspond to play screen area
       and the areas
       Directly to the sides. We disregard the areas
          above and below
       for simplicity, as we proved wall bouncing in (1)
          */
    ({bx' = vx, by' = vy & bx >= 0 & bx <= Width & by >=
       0 & by <= Height} ++
     {bx' = vx, by' = vy & bx <= 0 & by >= 0 & by <=
        Height} ++
    {bx' = vx, by' = vy & bx >= Width & by >= 0 & by <=
       Height}))*@invariant(vx^2 + vy^2 = vel^2 & bx >= 0
        & bx <= Width & by >= 0 & by <= Height)  /* Loop
       Invariant */
\]
(bx >= 0 & bx <= Width & by >= 0 & by <= Height) /*
   Safety Condition */
}
```

A.3 Key file for equations 4 and 5

```
/* REQUIRED INFORMATION:

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

Name: Milda Zizyte & Felix Hutchison
KeYmaera version: 3.6.16
```

```
Backends used (Mathematica, Z3, ...): Mathematica
Nodes: 13692
Branches: 1223
Interactive steps: 0
Arithmetic Solver: 143.34
Time: 226.524
Proof completes (Y/N): Y

*/

\functions {
R Width;
R Height;
R vel;
}

\programVariables {
R bx;
R by;
R vx;
R vy;

R paddleWidth;
//R Px;
R Py;
R Pvy;
//R Ox;
R Oy;
}

\problem{

/* INITIAL CONDITIONS */
(Width > 0 & Height > 0 & bx > 0 & by > 0 & bx < Width &
    by < Height & vx^2 + vy^2 = vel^2 & vel > 0 &
    paddleWidth > 0 & Py = by) /* & Px = 0 & Ox = Width)
    */
->
\[
  (

     /* WALL PHYSICS EVENTS */
        if (by = 0 & vy < 0) then (vy := -vy)
        else (if (by = Height & vy > 0) then (vy := -vy)
            fi)
        fi;
```

```
    /* PADDLE CONTROL */
    /* Player paddle tracks the ball */
    Pvy := vy;
    Oy := by;


    /* PADDLE PHYSICS EVENTS */
    if (bx = 0 & by > Py - paddleWidth &
          by < Py + paddleWidth & vx < 0) then (vx := -
              vx)
       else (if (bx = Width & by > Oy - paddleWidth &
          by < Oy + paddleWidth & vx > 0) then (vx := -
              vx)
           fi)
        fi;

    /* CONTINUOUS DYNAMICS */
    ({bx' = vx, by' = vy, Py' = Pvy & bx >= 0 & bx <=
       Width & by >= 0 & by <= Height} ++
     {bx' = vx, by' = vy, Py' = Pvy & bx <= 0 & by >= 0 &
         by <= Height} ++
    {bx' = vx, by' = vy, Py' = Pvy & bx >= Width & by >=
       0 & by <= Height})@invariant(Py = by) /* Diff
       Invariant*/
    )*@invariant(bx >= 0 & bx <= Width & Py = by)  /*
       Loop Invariant */
\]
(bx >= 0 & bx <= Width) /* Safety Condition */
}
```

A.4 Key file for equations 6 and 8

```
/* REQUIRED INFORMATION:

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

Name: Milda Zizyte & Felix Hutchison
KeYmaera version: 3.6.16
Backends used (Mathematica, Z3, ...): Mathematica
Nodes: 34285
Branches: 3846
Interactive steps: 1227
Arithmetic Solver: 2958.415
Time: 2469.39
```

```
Proof completes (Y/N): Y

*/

\functions {
R Width;
R Height;
R vel;
}

\programVariables {
R bx;
R by;
R vx;
R vy;

R paddleWidth;
//R Px;
R Py;
R Pvy;
R Pvmax;
//R Ox;
R Oy;
}

\problem{

/* INITIAL CONDITIONS */
(Width > 0 & Height > 0 & bx > 0 & by > 0 & bx < Width &
    by < Height & vx^2 + vy^2 = vel^2 & vel > 0 &
    paddleWidth > 0 & Pvmax^2 > vy^2 & Pvmax > 0 & by >=
    Py - paddleWidth &
            by <= Py + paddleWidth) /* & Px = 0 & Ox =
                Width) */
->
\[
  (

     /* WALL PHYSICS EVENTS */
        if (by = 0 & vy < 0) then (vy := -vy)
        else (if (by = Height & vy > 0) then (vy := -vy)
            fi)
        fi;

     /* PADDLE CONTROL */
     if (by > Py) then (Pvy := Pvmax)
```

```
    else (Pvy := -Pvmax)
    fi;
    /* Opponent Paddle moves non-deterministically to be
       where it needs to be */
    Oy := by;


    /* PADDLE PHYSICS EVENTS */
    if (bx = 0 & by >= Py - paddleWidth &
          by <= Py + paddleWidth & vx < 0) then (vx :=
             -vx)
          /* rebound off of the player paddle */
       else (if (bx = Width & by >= Oy - paddleWidth &
          by <= Oy + paddleWidth & vx > 0) then (vx :=
             -vx)
          /* rebound off the opponent paddle */
         fi)
       fi;

    /* CONTINUOUS DYNAMICS */
    (
/* first three diff eqs are the ones that govern real
   behavior */
     {bx' = vx, by' = vy, Py' = Pvy & bx >= 0 & bx <=
        Width & by >= 0 & by <= Py - paddleWidth} ++
     {bx' = vx, by' = vy, Py' = Pvy & bx >= 0 & bx <=
        Width & by >= Py - paddleWidth & by <= Py +
        paddleWidth} ++
     {bx' = vx, by' = vy, Py' = Pvy & bx >= 0 & bx <=
        Width & by >= Py + paddleWidth & by <= Height} ++
/* These three diff eqs govern behavior past the player
   paddle. Nothing
     happens in them, cause the ball provably never ends
        up there. */
     {bx' = vx, by' = vy, Py' = Pvy & bx <= 0 & by >= 0 &
         by <= Py - paddleWidth} ++
     {bx' = vx, by' = vy, Py' = Pvy & bx <= 0 & by >= Py
        - paddleWidth & by <= Py + paddleWidth} ++
     {bx' = vx, by' = vy, Py' = Pvy & bx <= 0 & by >= Py
        + paddleWidth & by <= Height} ++
/* These three diff eqs govern behavior past the opponent
    paddle. Nothing
     happens in them, cause the ball provably never ends
        up there. */
     {bx' = vx, by' = vy, Py' = Pvy & bx >= Width & by >=
         0 & by <= Py - paddleWidth} ++
```

```
      {bx' = vx, by' = vy, Py' = Pvy & bx >= Width & by >=
          Py - paddleWidth & by <= Py + paddleWidth} ++
      {bx' = vx, by' = vy, Py' = Pvy & bx >= Width & by >=
          Py + paddleWidth & by <= Height}
      )@invariant(by >= Py - paddleWidth & by <= Py +
          paddleWidth & Pvmax^2 > vy^2) /* Diff Invariant*/

    )*@invariant(bx >= 0 & bx <= Width & by >= Py -
        paddleWidth & by <= Py + paddleWidth & Pvmax^2 >
        vy^2)  /* Loop Invariant */
\]
(bx >= 0 & bx <= Width) /* Safety Condition */
}
```

A.5 Key file for equations 7

```
/* REQUIRED INFORMATION:

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

Name: Milda Zizyte & Felix Hutchison
KeYmaera version: 3.6.12
Backends used (Mathematica, Z3, ...): Mathematica
Nodes:
Branches:
Interactive steps:
Arithmetic Solver:
Time:
Proof completes (Y/N): N

*/

\functions {
R Width;
R Height;
R vel;
}

\programVariables {
R bx;
R by;
R vx;
R vy;

R paddleWidth;
```

```
//R Px;
R Py;
R Pvy;
R Pvmax;
//R Ox;
R Oy;
}

\problem{

/* INITIAL CONDITIONS */
(Width > 0 & Height > 0 & bx = Width & by > 0 & vx < 0 &
    by < Height & vx^2 + vy^2 = vel^2 & vel > 0 &
    paddleWidth > 0 & Pvmax^2 > vy^2 & Pvmax > 0 & (-
    Height * (Width/vel)) <= Pvmax & Py < Height & Py > 0)
     /* & Px = 0 & Ox = Width) */
->
\<
  (

    /* WALL PHYSICS EVENTS */
       if (by = 0 & vy < 0) then (vy := -vy)
       else (if (by = Height & vy > 0) then (vy := -vy)
           fi)
       fi;

    /* PADDLE CONTROL */
    if (by > Py) then (Pvy := Pvmax)
    else (Pvy := -Pvmax)
    fi;
    /* Opponent Paddle moves non-deterministically to be
       where it needs to be */
    Oy := by;

    /* We don't need paddle interactions since it's only
       traveling on the court. */

    /* CONTINUOUS DYNAMICS */
    (
/* first three diff eqs are the ones that govern real
    behavior. since we're
     only looking at the court , these are the only  */
     {bx' = vx, by' = vy, Py' = Pvy & bx >= 0 & bx <=
         Width & by >= 0 & by <= Py - paddleWidth} ++
      {bx' = vx, by' = vy, Py' = Pvy & bx >= 0 & bx <=
          Width & by >= Py - paddleWidth & by <= Py +
```

```
        paddleWidth} ++
      {bx' = vx, by' = vy, Py' = Pvy & bx >= 0 & bx <=
        Width & by >= Py + paddleWidth & by <= Height})
    )*
\>
(by >= Py - paddleWidth & by <= Py + paddleWidth) /*
    Termination Condition */
}
```

B Soundness proofs

$$DC_{\langle\rangle}\ \frac{\Gamma \vdash [x' = \theta \,\&\, H]C \qquad \Gamma \vdash \langle x' = \theta \,\&\, (H \wedge C)\rangle\phi}{\Gamma \vdash \langle x' = \theta \,\&\, H\rangle\phi}$$

Assume $\Gamma \vdash [x' = \theta \,\&\, H]C$ and $\Gamma \vdash \langle x' = \theta \,\&\, (H \wedge C)\rangle\phi$ are valid.

Then, for all ν, $\nu \models \Gamma \rightarrow \langle x' = \theta \,\&\, H \wedge C\rangle\phi$.

Thus, for all ν such that $\nu \models \Gamma$, there exists an $\omega \in \rho(x' = \theta \,\&\, H \wedge C)(\nu)$ such that $\omega \models \phi$.

Since $\omega \in \rho(x' = \theta \,\&\, H \wedge C)(\nu)$ must also belong to $\rho(x' = \theta \,\&\, H)(\nu)$ by definition of evolution domains.

Thus for all ν such that $\nu \models \Gamma$, there exists an $\omega \in \rho(x' = \theta \,\&\, H)(\nu)$ such that $\omega \models \phi$.

So, $\Gamma \vdash \langle x' = \theta \,\&\, H\rangle\phi$ is valid. (soundness)

Also, for all ν, $\nu \models \Gamma \rightarrow [x' = \theta \,\&\, H]C$.

Thus, for all ν such that $\nu \models \Gamma$, for all $\omega \in \rho(x' = \theta \,\&\, H)(\nu)$, $\omega \models C$.

So, for all $\omega \in \rho(x' = \theta \,\&\, H)(\nu)$ where $\nu \models \Gamma$, $\omega \in \rho(x' = \theta \,\&\, H \wedge C)(\nu)$.

This means that if $\Gamma \vdash \langle x' = \theta \,\&\, H\rangle\phi$ is valid , then $\Gamma \vdash \langle x' = \theta \,\&\, (H \wedge C)\rangle\phi$ is valid. (completeness)

$$\text{Convergence Substitution}\ \frac{\Gamma \vdash [\alpha^*]C \qquad \Gamma \vdash \langle\alpha^*\rangle F \qquad F, C \vdash \phi}{\Gamma \vdash \langle\alpha^*\rangle\phi}$$

Assume $\Gamma \vdash [\alpha^*]C$ and $\Gamma \vdash \langle\alpha^*\rangle F$ and $F, C \vdash \phi$ are valid

Thus, for all $\omega \in \rho(\alpha^*)(\nu)$ such that $\nu \models \Gamma$, $\omega \models C$

It is also the case that there exists $\omega_0 \in \rho(\alpha^*)(\nu)$, such that $\nu \models \Gamma$ and $\omega_0 \models F$.

This ω_0 satifies both F and C by the above.

Lastly, it is the case that for any state ν that satisfies F and C, $\nu \models \phi$

Thus $\omega_0 \models \phi$

So there exists a state $\omega \in \rho(\alpha^*)(\nu)$ where $\nu \models \Gamma$ such that $\omega \models \phi$

Thus $\Gamma \vdash \langle\alpha^*\rangle\phi$ is valid.

$$\text{Loop Segmentation } \frac{\Gamma \vdash \langle (\alpha^n)^* \rangle \phi}{\Gamma \vdash \langle \alpha^* \rangle \phi}$$

Assume $\Gamma \vdash \langle (\alpha^n)^* \rangle \phi$

For all states ν, there exists $\omega \in \rho((\alpha^n)^*)(\nu)$ such that $\omega \models \phi$.

Since all $\omega \in \rho(\alpha^n)^*)(\nu)$ also belong to $\rho(\alpha^*)(\nu)$ by definition of α^*, we can state that for all states ν, there exists $\omega \in \rho(\alpha^*)(\nu)$ such that $\omega \models \phi$.

Thus $\Gamma \vdash \langle \alpha^* \rangle \phi$ is valid.