CPS analysis of Pong

Felix Hutchison
Milda Zizyte

Motivation

- Game physics is hard
  - Even when your physics engine is good.

Motivation

- Interactions combine in interesting ways

Motivation

- You may want to make guarantees of certain conditions (e.g. player altitude above ground) for things to function (e.g. AI algorithm)

- Can we use CPS techniques, like dL, to make these guarantees?

Why Use Differential Dynamic Logic?

- Formal guarantees
  - High assurance for high exposure products like videogames
- Great for event based interactions and continuous dynamics
  - Like physics simulation

KeYmaera Hybrid Verification Tool

- Automated and interactive theorem prover for dL
- All the following proofs will prove automatically
  - No team of formal methods experts required!
  - Though in some cases manual interventions were used to speed the process.

Let’s prove some game physics!

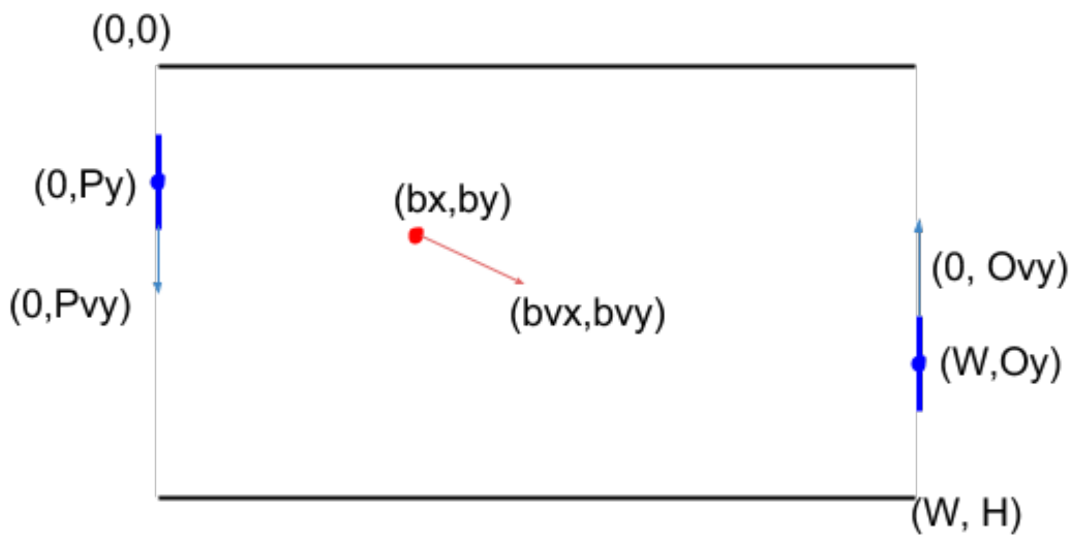
But…

- We're broke grad students, we can't afford real video games
  - Train simulator and DLC totals to over \$4000

- So we'll look at Pong
  - Plenty of free versions with source available

Pong program model

- Ball has constant speed in each direction
- Paddles move at the far ends of the court



Based on http://gamemechanics.wikia.com/wiki/Pong and http://en.wikipedia.org/wiki/Pong

Preliminaries

- Make sure our physics is doing what we think
  - Ball bouncing and paddle interactions

- Even this is non-trivial!
- Some bugs in ordering of events:
  - Paddle interactions vs. paddle control algorithm.

Basic AI

- Ball follower
  - Controller A) Matches ball velocity
  - Controller B) Moves at a fixed speed faster than the ball, keeps ball above the paddle

- Can we prove perfect play with these controllers?
  - I.e. Against an infallible opponent, can we assure no point is scored

$$\Gamma \rightarrow [(\beta, \alpha)^*] 0 \leq bx \leq Width$$

Controller A

- Does this work?

$$\Gamma, Py = by \rightarrow [(\beta, \alpha)^*] Py = by$$
$$\beta \equiv \{Pvy := bvy\};$$

- Does this ensure perfect play?

$$\Gamma, Py = by \rightarrow [(\beta, \alpha)^*] 0 \leq bx \leq Width$$

- Unsurprisingly, yes.
  - Proof takes 226.524 seconds (+ 143.34 seconds in Mathematica)
  - 13692 proof steps
  - 1223 branches
  - Mostly symmetric/similar braches
    - Lemmas will greatly speed up proof

Controller B

- If the Ball is over the paddle, can we keep it there?
- Can we get the ball over the paddle every time?
- Does this ensure perfect play?

Controller B

- If the Ball is over the paddle, can we keep it there?

$$\Gamma, F \rightarrow [(\beta, \alpha)^*]F$$

$$\beta \equiv \{\text{if } (Py > by)$$

$$\text{then } (Pvy := Vel)$$

$$\text{else } (Pvy := -Vel)\};$$

$$F \equiv Py - Pw \leq by \leq Py + Pw$$

Controller B

- Since this again trivially shows perfect play, we can do that too.

$$\Gamma, F \rightarrow [(\beta, \alpha)^*]F, 0 \leq bx \leq Width$$

- Proves automatically again
  - Proof takes: 2469.39 (+ 2958.415) seconds
  - 34285 proof steps
  - 3846 branches
    - Again, mostly symmetrical

Controller B

- Can we get the ball over the paddle every time?

$$\Gamma \rightarrow <(\beta, \alpha)^{*}> F$$

- Unfortunately this may not be provable in KeYmaera as it is.
  - Loop convergence (induction) won't work because there's no guaranteed possibility of progress
  - E.g. The ball stops within epsilon of hitting the wall, then it can only progress at most epsilon in this iteration.

Controller B

- So KeYmaera doesn't help, but is it dL provable?
- Yes!
  - Using Convergence Substitution, and Loop Segmentation for <> modality
  - Full proof, and soundness for the above rules, in the paper
- And these rules can be added to KeYmaera

Conclusion: Are CPS tools useful for this type of analysis?

Some drawbacks:

- Still developmental
- Additional features needed
  - But all are implementable or in progress

But more importantly:

- Immensely powerful
- Formal guarantees are the best way to ensure high quality products
- Planned improvements give great benefits to the speed of automation

Questions?

Next Steps

- ModelPlex
  - Runtime verification of model assumptions
  - Automatically generated formal monitors from proof

- In this case assumptions are
  - Physics engine
  - Interaction assumptions
  - Bounds/initial conditions

Added Rules

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

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