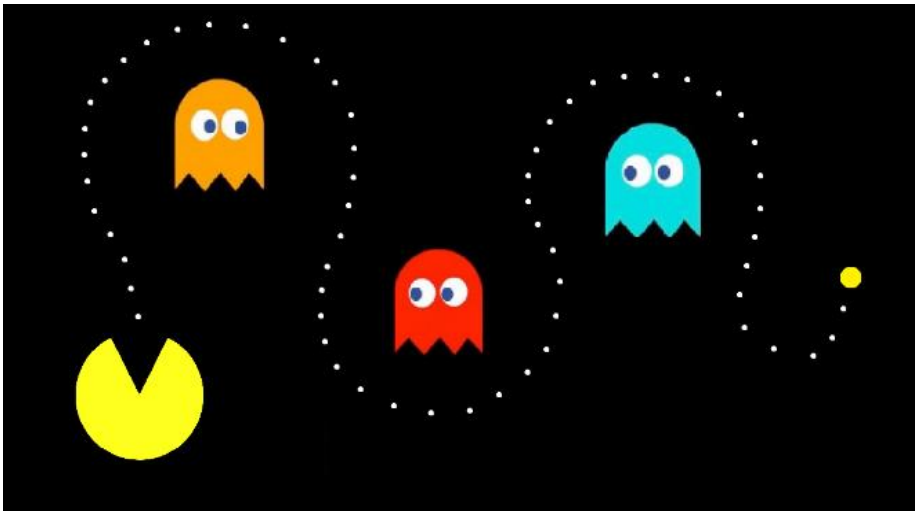
Safe Path Planning for an Autonomous Agent in a Hostile Environment

a.k.a

Save PacMan!

Jimit Gandhi and Astha Prasad

May 2, 2016



Abstract

Robot interaction with dynamic environments is becoming more and more commonplace in today's world. This paper explores a two dimensional world with an autonomous agent moving about a constant plane riddled with dynamic ghost agents. The environment is made increasingly hostile and difficult to maneuver by giving the ghost agents progressively sophisticated dynamics. Here, we present the approach adopted in modeling the environment as well as the dynamics and constraints of all the involved agents. We develop the necessary safety and efficiency conditions needed to design the controller that ensures collision avoidance between our agent and ghosts. Finally, we present a formal proof guaranteeing the safety of our agent using KeymaeraX, a theorem prover for differential dynamic logic (dL).

1. Introduction

This project is inspired by the cherished arcade game of PacMan which has been an integral part of our childhood. We have all seen times when we would get greedy for dots and end up making a bad decision, leaving PacMan surrounded by ghosts and nowhere to go.

The concept behind PacMan poses an excellent framework to build models that are based on safety critical dynamics. PacMan can be thought of as a robot or rover that is constantly in motion, thrown into a world with random hostile agents (obstacles) that it must avoid. Rather than humans controlling the movements of PacMan, we have set out to build an intelligent controller that understands the dynamics of the environment as well as the constraints of PacMan's movements, and can thus ensure his safety.

As we delved deeper into this problem, we recognized that it could be mapped to many existing real-world Cyber-Physical systems we see today, especially in the field of robotics. Specifically, a system where there is one bot in an environment with many static and moving obstacles. It is, however, necessary to know the behavior of the moving obstacles in order to define safe controls. A drone flying in an airspace surrounded by other aerial vehicles would require similar controls for safety.

In real world systems, timely detection of an impending collision largely depends on the quality of sensors and the speed of computation of the system on board. Hence, we have modeled our system to be time triggered, ensuring that is takes into account the lag that we see in real life between physical changes in the environment and their detection and interpretation by the cyber system.

Our agent is modeled to be quite similar to PacMan wherein it is constantly moving about in a 2 dimensional plane. The world that we have modeled, however, is grid less and the agent is allowed to move along the perimeter of circles of random radii. The idea is to introduce harmful ghost agents with increasingly sophisticated dynamics and prove the safety of the system at each step up. To start off, we first model a ghost as simple static obstacle and then move on to give it some constant velocity. We then give it the capability to accelerate and brake arbitrarily. Finally, we model a system with more than one ghosts and prove the condition for safety at all these levels.

2. Related Work

For many years now, games have been a test bed for Artificial Intelligence and Machine Learning algorithms. The game of PacMan was proven to be an NP-Hard problem by Giovanni Viglietta of the University of Pisa, Italy [1]. There has been a lot of research in coming up with strategies and computing the complexity of games from the 80s and 90s.

Many algorithms have been developed in which the PacMan agent turns out to be victorious. One such algorithm employs a

simple tree search method [2] wherein a decision made by PacMan is expanded to generate subsequent states. The ghost's decision for each of the expanded states gives birth to further states and so on, to generate a decision tree. These algorithms require high computational power as they grow exponentially with the depth of the decision tree. The final strategy or decision for PacMan is determined by a heuristic cost function. Monte Carlo tree search is another framework employed to play PacMan [3]. The algorithm is used to find an optimal path for an agent at each turn which determines the path to take based on randomized simulations.

There is also ongoing research on safe obstacle avoidance for autonomous robotic ground vehicles [4] at Carnegie Mellon University which studies the safety properties to ensure passive safety (where all collisions are avoided while the robot is moving) and a stronger passive safety (where a certain buffer is maintained to ensure sufficient maneuvering distance between obstacles. While the study presented in our paper only explores safety against the worst possible case, it would be interesting to look into strategies where a buffer is maintained between agents.

To make matters interesting and more real world, we modified certain cyber physical features such as making the environment grid less, giving our PacMan some dynamics and more freedom to move about in the plane.

For years, mathematicians have been conducting research on proving the completeness of such problems and we thought it would be interesting to explore some of the nuances of this PacMan game. This project aims to figure out where the major challenge related to a similar set-up lies and to relate this problem to real world scenarios. It is as old and as interesting as it could be.

3. Basic System Model

The PacMan game is very complicated as it depends on numerous things such as the maze design, the number of ghosts, the decisions of the ghost. Proving the hybrid program design will be very difficult as the number of states in the decision tree grows exponentially due to the above mentioned factors.

The power of proving Cyber-physical systems using sequent calculus is that one can break a very complex hybrid program into small simple hybrid programs and then prove these simplified hybrid programs separately. Proving them separately eventually proves the complex hybrid program since they have been derived from the latter.

We use similar approach with the PacMan problem breaking it into simplified hybrid programs and developing a strong foundation of the system proving them. We can then add on complexities we mentioned and verify them as well. In order proceed with the proof nonetheless, we first start off with the simplest of system and prove the hybrid program design for safety and

efficiency. Then we keep adding complexities which account for maze design as well as the complex dynamics for the ghosts.
But before we get into details of our proof methodology, we should first describe PacMan's dynamics and the ghost dynamics and the world as well.

PacMan

- Since our PacMan is not exactly the PacMan of the game but rather a simulated version of a real world robot we will change the dynamics and the world in which PacMan moves.
- PacMan is moving in a 2D plane. He is capable of circular dynamics and can accelerate and brake with values A or B.
- The control decision of the PacMan will be time triggered as event driven hybrid programs don't work well with real world robots. The time in Time-triggered hybrid programs depends on the time delay in which the sensor data or the information about the world is received. Let's call this time 'T'.
- The PacMan has the ability to accelerate with a constant value and brake with a constant value but for at most time T before it gets a new updated data about the world.
- The PacMan has some radius of its own and has to avoid spinning about its own axis.
- There will be only one PacMan.

Ghost

- A ghost is moving in the same 2D plane with a linear dynamics.
- A ghost can either be static, moving with constant velocity or accelerate or decelerate randomly.
- If the ghost decides randomly to accelerate or decelerate it will do so for at most time T.
- The ghost can be one or two or more.
- Ghost also has its own radius and the game is over if it collides with the PacMan.

Interaction and Controls

- PacMan knows the location (XY coordinates) of the ghost at the control decision and chooses to accelerate or brake accordingly along a circle with a random radius generated at least once every time T.
- PacMan has to satisfy the two conditions to remain safe and they are
  - ❖ Should not collide with the ghost considering radius of itself and the ghost
  - ❖ Should never overrule circular dynamics.
- PacMan will also know the velocity of the ghost at every point of update which is at most time T interval.

Maze or world map

- It will be a 2 dimensional world and we will have static ghosts as obstacles with finite radius.
- Initially we will emulate real world scenario with few static obstacles and few moving ghosts.

- In order to imitate the exact same maze design as that of PacMan game one can simply increase the number of obstacles and limit the size of the 2 dimensional world and with these defined one can have the desired maze design.
- Since imitating the exact same design of the PacMan maze will make the problem all the more complicated, we will initially start with just one obstacle and then increase them accordingly.

Now that we have defined our agents and their interaction in the world we shall begin with our ideal PacMan scenario which would be the desired final goal

Ideal PacMan Scenario: One PacMan, 3-5 ghosts, many obstacles forming the maze design (>50).
Since the above scenario is very complex and will require longer time for theorem provers to prove, we de-scope our desired goal for this paper and then mention the other desirable goals in the future work part.
Note that our major effort here is not to prove the PacMan controller to be safe. It is more of an effort to study the PacMan scenario and determine the properties of the entire system and their agents and based on that build some foundations on designing of the controller.

End Goal Scenario: A PacMan agent in a 2 dimensional world, a few (3) ghosts moving around in the world with a few obstacles (3-5).
To achieve the above end goal we divide and simplify the problem further

- Multiple ghosts only
- Multiple obstacles only

Again we further simplify our problem by first proving

- Single ghost followed by two ghosts and so on
- Single obstacle followed by two obstacles and so on

The program that we design for ghost shall have the following different variants

- Ghost static (for obstacle)
- Ghost is moving with constant velocity
- Ghost has random acceleration and deceleration.

Note that in all of the above scenarios, the design of our controller will be same. It will be designed generically such that it works for all the above scenarios.
The above divisions of the final end goal has led us to prove the following systems.
We have divided our end goal scenario into certain small milestones and we can be sure that proving them will lead us to prove the end goal entirely.

The following are the steps or milestones that we need to prove.

1. Circular dynamics of our Pacman in the 2 dimensional world with no obstacles.

2. Circular dynamics of our Pacman in 2 dimensional space with one static obstacle.

3. Circular dynamics of our Pacman in 2 dimensional space with one single ghost moving with constant velocity.

4. Circular dynamics of our Pacman in 2 dimensional space with one single ghost which accelerates or brakes randomly.

5. Circular dynamics of our PacMan in 2 dimensional space with multiple obstacles.

6. Circular dynamics of our Pacman in 2 dimensional space with multiple ghosts moving with constant velocity.

7. Circular dynamics of our Pacman in 2 dimensional space with multiple ghosts which accelerate or brake randomly.

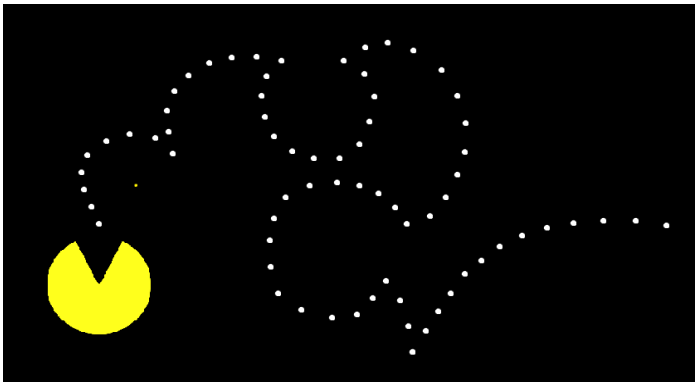


Figure 2: PacMan in 2D world- No obstacles, no ghosts

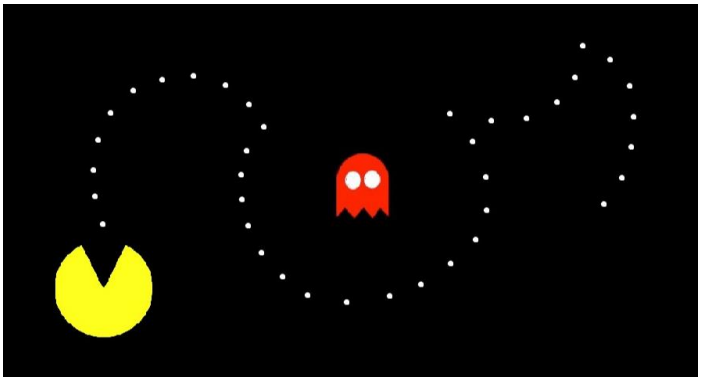


Figure 1: PacMan in 2D world-Static ghost (can be considered as an obstacle)

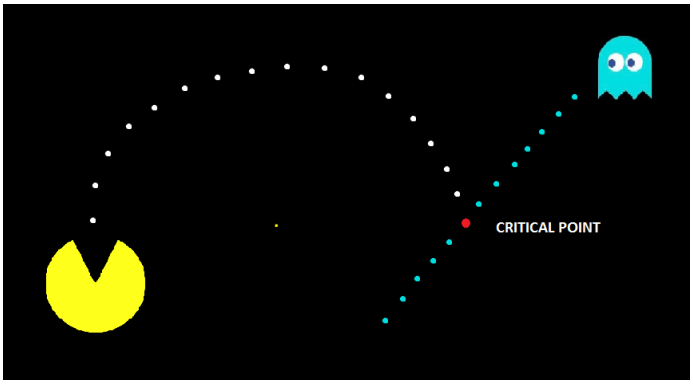


Figure 3: PacMan in 2D world- Single ghost moving

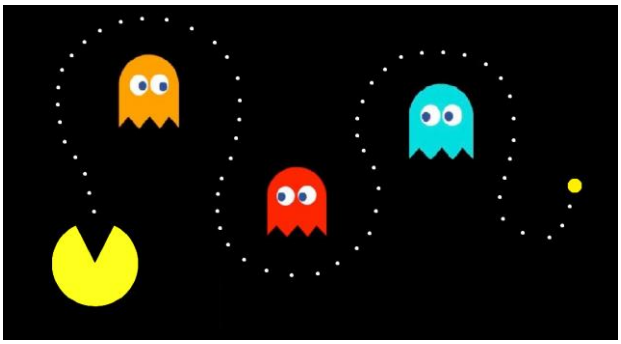


Figure 4: PacMan in 2D world- Multiple obstacles

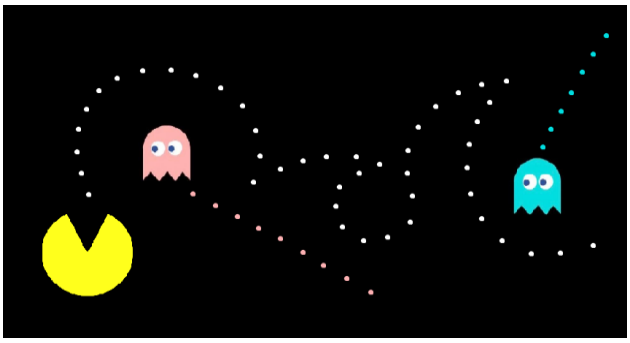


Figure 5: PacMan in 2D world- Multiple Ghost moving

3.1 Description of Safety and Efficiency

The system will be proved using dL logic initially because for first step of the project we will not be giving any control decisions to the ghost. We will need properties such as differential cuts, differential invariants for circular dynamics and differential weakening for the first step.

Safety

For safety we calculate the distance ghost will cover in time T and for the new center and track radius the PacMan calculates the distance it will cover for time T. If the two do not collide then the decision will be safe and PacMan will proceed with its dynamics.

Another way to approach is the geometrical approach which is shown in the figure.
So we know the straight line equation (where velocity direction would give us the slope and position would give us the intercept of the line) which is of the form **y=mx+c.** Now whenever PacMan takes the control decision, we know the circle equation (new radius and the new center. We can then calculate and test if this condition is valid for x direction

?(xghost+velghost*T-xPacMan)^2
>(v*T+0.5*A*T^2+vT^2/2*B)^2

and similarly for y. And at the same time we can also calculate the actual x and y position the PacMan will end up in if it accelerates and then stops using the same kinematics equation i.e

x PacMan_future= xPacMan+
(v*T+0.5*A*T^2+vT^2/2*B)

and similarly for yPacMan_future. If this point (xPacMan_future, yPacMan_future) at which the robot will finally come to a stop satisfies the line equation for the ghost y=mx+c in time T or in other words we run this test

?(yPacMan_future =
m*xPacMan_future + c)

we will know if the PacMan has danger of colliding with the ghost or not. If our new controller radius and centers are chosen such

that they pass the test then the controller is implemented or else it is simply discarded and a new value is chosen

Once we prove that, then we plan to add complexity of the project by giving the ghost the power to make decisions such as accelerating and braking for at most time T but the ghost will have straight line dynamics and also will be able to switch its velocity direction. Here we will require the dGL properties like dual operator or we can convert the dGL into simplified dL hybrid programs.

Safety for PacMan will be when it makes the choice of not colliding with the ghost for the worst case taking into consideration that is will accelerate or brake and assuming the velocity direction to be towards him. The choice for ghost for constant velocity case will be something like

(vGhost = vGhost or vGhost = -vGhost)

We can add control decisions by giving the Ghost the choice between acceleration and deceleration equally in both x and y directions and it can have negative velocity as well

(acc= +A or acc= -B)

Hybrid Game extension

We plan to go step by step. Since KeymaeraX is not tried and tested much for dGL we transformed it into simplified dL version. So we write the generic hybrid game for our system for one ghost and one PacMan case. Do a hand written proof up until we convert it to a dL hybrid program and then proceed to prove that hybrid program on KeymaeraX.

So basically the controller of PacMan makes the control choice whether to accelerate or decelerate based on the ghosts location and velocity. To check if the decision is safe the PacMan's controller takes into account the prediction of ghost position in two consecutive time intervals of T. Since the ghost has the choice to either accelerate with A or decelerate with B, it has to make two decisions for the two consecutive time interval of T. So below is the tree of how the game proceeds for the two consecutive time intervals T i.e. 0 to 2T. Thus there are only four possible combinations for ghosts to make up to two time intervals of accelerations and decelerations. Based on our derivation from hybrid game to n simple hybrid program where n is the number of possible combinations that the ghost can make in terms of accelerations and decelerations. In this case it is 4. So now we have to prove just 4 differential hybrid programs for single PacMan- ghost hybrid game.

In the derivation where we breakdown the hybrid game to hybrid system we have this four combinations as $\beta 1, \beta 2, \beta 3\ and\ \beta 4$. All are the choices of the ghost.

Now we can separately prove them where
αcorresponds to the controller program of the PacMan
β1 corresponds to (acc:=A for t=0 to T and acc:=A for t=T to 2T)
β2 corresponds to (acc:=A for t=0 to T and acc:=-B for t=T to 2T)
β3 corresponds to (acc:=-B for t=0 to T and acc:=A for t=T to 2T)
β4 corresponds to (acc:=-B for t=0 to T and acc:=-B for t=T to 2T)

$$\frac{\Gamma \vdash \langle[\beta_1][\alpha]\rangle^*P \quad \Gamma \vdash \langle[\beta_2][\alpha]\rangle^*P \quad \Gamma \vdash \langle[\beta_3][\alpha]\rangle^*P \quad \Gamma \vdash \langle[\beta_4][\alpha]\rangle^*P}{\Gamma \vdash \langle\, [\beta_1 \cup \beta_2 \cup \beta_3 \cup \beta_4]^d\, [\alpha]\rangle^*\, P}\,(u)$$

$$\frac{\Gamma \vdash \langle\, [\beta_1 \wedge \beta_2 \wedge \beta_3 \wedge \beta_4]^d\, [\alpha]\,\rangle^*\, P}{\Gamma \vdash \langle\, [\beta_1 \wedge \beta_2 \wedge \beta_3 \wedge \beta_4]^d\, ;[\alpha]\rangle\, *\, P}\,(;)$$

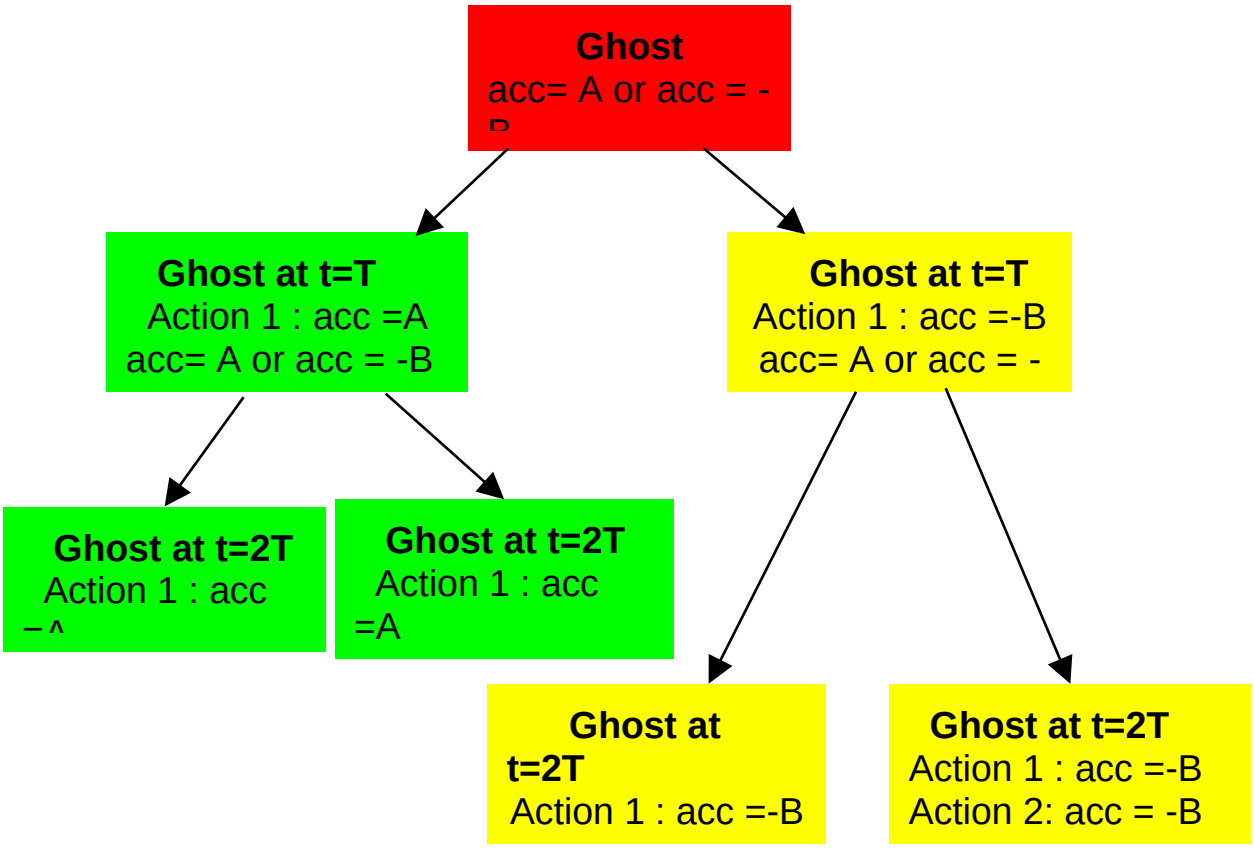


Figure 6: Action tree progression for ghost from t = 0 to t=2T

Assumptions

Our assumption would be that the initial positions of both PacMan and ghost would not be same as that would cause the PacMan to lose instantly as the game starts.

While checking for the condition of collision for control decision, one assumption we make is to separate a single complex circular dynamics into two 1-Dimensional linear dynamics case. We do this because it is hard to model using trigonometric functions and parametric coordinates. This assumption is valid as according to Newton's laws, all we care about is displacement and not the actual path to be taken. This assumption also helps increase the efficiency of the system as it is now decoupled into two one dimensional motion equation.

Our assumption that PacMan has circular dynamics is because, first of all it makes 2D traversing more interesting and second we have already proved it during the assignment for a Wall-E and Eva. So it is a good place to start with. We would build the complexities of the game from this foundation. Also adding the dynamics makes it more of a cyber-physical system than just a normal game (which could be boring).

We also plan to have obstacles instead of a maze as when we increase the number of obstacles it could eventually become a maze and PacMan will have less places to go to.

The other assumption is that we have ghost with linear dynamics because if we give same dynamics and control decision to both ghost and PacMan then the game would never end and the decision tree will explode as it is NP hard problem.

4. Other Variants

The model described above is a basic implementation of the problem. We plan to expand the model by increasing the complexity in several ways. The other variants for this project are outlined below:

- Include mines/dangerous locations that can blow up when PacMan travels over them, decreasing PacMan's speed for some time, say '3T'. This condition will further limit PacMan's chances of winning/avoiding the approaching ghosts.
- Introduce obstacles, i.e. locations where PacMan is not allowed to go. These will be implemented as simple static ghosts.
- Model the system using dGl, i.e. allow the ghosts to make control decisions based on the knowledge of PacMan's location.

5. Final Deliverables

Our final goal is to model and prove a system that:

- Implements circular dynamics for PacMan
- Implements linear dynamics for multiple ghosts
- Introduces obstacles (static ghosts) for PacMan to avoid
- Introduces hybrid games and forms basis for further development

Since Keymaera is not tried and tested much for dGl, we plan to at least make a model for the PacMan game and decide the winning regions based or states for both PacMan and one ghost.

Final Remarks and Future Work

Similarity with MiniMax algorithm

Many algorithms have been tried to solve the PacMan problem. Each of the algorithm explores the future states based upon the actions the opposite player takes and a decision tree is formed. The most commonly used algorithm for game theoretic approach is the MiniMax Algorithm.

The principle of this algorithm is basically forming a 'maximin' value for a player during its turn. Calculating the maximin value of a player is done in a worst-case approach: for each possible action of the player, we check all possible actions of the other players and determine the worst possible combination of actions. Our controller is also designed based on the worst case combination of actions that ghost can take in the form of choosing algorithms. Many books [1] have the proof of MiniMax algorithm mathematically which says that this algorithm guarantees optimal strategy only if there exists one.

Future Work

This is just the basic foundation where hybrid games just meet with hybrid systems. The further extensions of this would be to have a complex control strategy for the ghosts as well and then test the controller's robustness to this. Another possibility would be to increase the number of obstacles greatly in order to generate the maze design such as that of real PacMan game. In this case however since the PacMan will have restricted movement, there will be regions where ghosts will have a winning strategy for instance, the PacMan is stuck between two ghosts and has nowhere to go. Proving or coming up with a very robust controller for those kind of situations require to study the worst case possible actions of multiple agents together. The combinations of the actions will be of the order$((2^{actions})^{number\ of\ agents})$. This will lead to a very large decision tree and hence PacMan is a prime example of a NP hard problem.

References

[1] Giovanni Viglietta. *Gaming is a hard job, but someone has to do it!* In Proceedings of the 6th International conference on Fun with Algorithms, June 2012

[2] David Robles and Simon M. Lucas, Senior Member, IEEE *A Simple Tree Search Method for Playing Ms. Pac-Man,* 2009 IEEE Symposium on Computational Intelligence and Games

[3] Tom Pepels, Mark H. M. Winands, Member, IEEE, and *Marc Lanctot Real-Time Monte Carlo Tree Search in Ms Pac-Man,* IEEE Transactions on Computational Intelligence and AI in Games, Vol. 6, No. 3, September 2014

[4] *On Provably Safe Obstacle Avoidance for Autonomous Robotic Ground Vehicles* Mitsch et al. and related

[5] R. D. Luce and H. Raiffa.,Games and Decisions, John Wiley, New York, 1957.