Fuzzy Logic versus simple Rules Based system for controlling an AI video game car

Jay Bretherton 1800231 Report for MAT301

# Introduction

## Overview

This project aims to evaluate the performance of a fuzzy logic system in solving a simple video-game oriented problem. To achieve this, an implementation of a fuzzy logic system that was written by this author was tested against a very simple control class which implements a rules-based solution to the same problem.

The aim is to prove if a more complex fuzzy inference system out-performs a more naive solution. The devised test is to drive a Kart around a lap of a track in the fastest time, while avoiding crashing into the walls of the track.

## Techniques

The solution, naturally, implements a fuzzy logic inference system as this is the system this experiment aims to test, and it implements a very simple rules based inference system as a baseline comparison.

The fuzzy system defines fuzzy numbers as being in a combination of five states (Tutorials Point, n.d.): Large negative, Medium negative, Close to zero, Medium positive, and Large positive. It then uses these fuzzy numbers to allow the construction of logical statements that do not need precise “crisp” values in their antecedent and consequent phrases. This means logical rules can be built up without having to define exact scenarios where they would come into effect. Furthermore, it contains systems for taking in crisp input data and converting it into fuzzy numbers (fuzzifying) and taking fuzzy output numbers and turning them back into crisp numbers that can be used by the rest of the solution (de-fuzzifying). This allows the system to be integrated into the solution in exactly the same way as the rules based system.

By comparison, the rules based system is intentionally far simpler. It consists of a series of evaluations comparing the data to exact numerical values and a list of Boolean logical rules. It then evaluates those rules and returns an output for the system. An RBS was chosen as the control it has the closest mapping onto a fuzzy rules system, with the main difference being the lack of fuzzy numbers – the key characteristic of fuzzy systems.

## Description of Solution

The solution implements an example microgame developed by (Unity Technologies, 2021) int which each AI controls a kart and attempts to drive it round a course. There is a sensor class which gathers information about the current state of the kart, and a controller class for each AI system. Each controller class takes data from the sensor and formats it before it is used as the input for its AI system. It then takes the suggested output given by its AI and validates it before passing this to the class which controls the Kart.

The example includes a countdown timer, but there are several “checkpoints” throughout the course which add time on to this. For this reason, each solutions’ time performance is recoded externally. The number of wall collisions will be counted manually by the individual carrying out the test.

## Hypothesis

The hypothesis held is that the fuzzy system will outperform the rules based system in both the time-to-complete of the lap, and in the number of collisions.

# Background

## Fuzzy Logic

Fuzzy logic is built off fuzzy set theory first proposed by (Zadeh, 1965). This model attempts to emulate the way natural logic is performed by people, where strict and exact values are not used but instead they use “fuzzy” values such as “tall, large, cold, or few”. It is particularly useful whenever discussing systems where statements can have degrees of truth and the system’s state can be in more than one form at once (Cintula, 2017). Because of this, fuzzy logic can be used in very complex systems so long as each attribute of the system is expressible on a scale of large negative to large positive These are then used to build up a rules base which is used by the fuzzy logic inference engine to determine the behaviour of the system for a given input.

Fuzzy systems implement logic through fuzzy sets logic rules which are different from their Boolean counterparts. Logical “And” is represented as a Minimum function of its two arguments. Logical “Or” is represented as a Maximum function. Logical Not is gained by taking 1 – value. (Guru99, n.d.).

An idealised fuzzy logic system is outlined below.

Crisp Input

Rules

Inference Engine

Fuzzifier

De-Fuzzifier

Fuzzy Input

Fuzzy Output

Crisp Output

Figure

As can be seen in Figure 1 here, crisp input is passes to the Fuzzifier which produces a fuzzy input. This is then passed to the inference engine which uses the rules set and fuzzy logic to produce Fuzzy output. This is then passed to the De-Fuzzifier which produces a crisp output.

## Simple Rules Based System

Rules based systems are in many ways identical to fuzzy rules based systems, as fuzzy logic systems are a subset of rule based systems. RBS use a list of rules in much the way a fuzzy logic system would but uses crisp numbers throughout. This means that each logical statement must be made with exact values for each rule (Deep AI, n.d.).

# Methodology

## Experiment Overview

## Unity Setup and Integration

### Kart Controller

The kart controller implements (Unity Technologies, 2021)’s KartGame.KartSystems.BaseInput interface which KartGame.KartSystems.ArcadeKart (the class that moves the kart) uses to gather its commands. The Kart controller can change which if the AI systems it uses at any time and switches between them in order to test both systems.

### Kart Sensor

The kart sensor class can be queried to discover the current state of the kart. It returns the Karts current speed, but also fires out raycasts forward, left, and right. The sensor returns if it hit a wall (implicitly), the distance to the wall it hit, and the surface normal of that wall. This information communicates all the AI would need to know to keep the car driving on the track and turn if there is a bend in the road coming up.

### Fuzzy Kart Control

The fuzzy kart control class is used by the kart controller when it is in the mode to test the fuzzy AI. This class is responsible for taking data from the sensor and normalising it into a form the fuzzy AI will accept as crisp input. This means taking the speed; forward, left, and right raycast distances; and the forward raycast surface normal and normalising them to a [-1, 1] scale.

The speed (which can never be negative or invalid) was mapped 0 to 0 and 10 (the max speed) to 1, with all negative values being unused.

The forward distance was mapped 0 to 0, 10 (the max length of the ray) to 1, and the state where the ray hit nothing was encoded as -1.

The same for the left and right rays as forward, except their max length is 5 so this was mapped to 1 instead.

For the surface normal, the signed angle difference between the surface normal and the negative forward of the kart was taken. As this angle cannot have a larger magnitude than 90°, this value was mapped to 1 and -90° to -1. In the event that the ray hit noting, this event was encoded as 0 as this would represent no action.

All intermediate values were linearly interpolated between the maximum, neutral and minimum.

When the fuzzy system does not have any rules at all that apply to the given inputs, it returns an (ISO, 2020) NaN representing no suggestion for the output value. In this case the fuzzy kart control ignores the output and does the safest option of not driving or turning.

## Rule based system Kart Control

The rules based system kart control class is used by the kart controller when it is in the mode to test the RBS AI. This class also takes in data from the sensor and evaluates it based on its hard coded rules.

This class implements a rules based system AI. As it exists to compare the fuzzy solution to, it was not made to be flexible or expandable, but instead implements the same rules that the fuzzy system does, but with hard coded explicit values. This allowed the entire system to be implemented very quickly. Below is an illustrative code snippet from (RBSKartController.cs, line 90). *[Some rules have been omitted for brevity]*

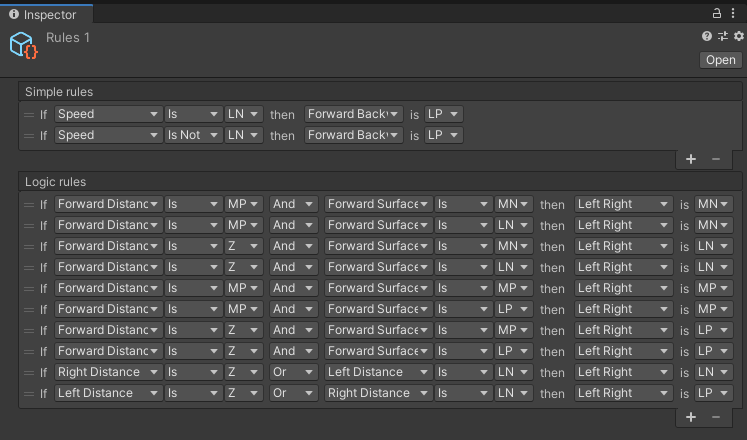


Figure

As can be seen, this system is very simple when compared to the several classes required to implement the fuzzy solution.

## Fuzzy Rules

For ease of use of the system, a unity scriptable object with a custom editor UI was developed for use when tuning the fuzzy rules.



Figure

The simple rules consist only of antecedent and consequent whereas the logical rules represent rules that have a logical (and / or) relationship between the who antecedent statements. These statements can be made up of “is” statements or “is not” statements, where the fuzzy logical Not of the input is taken.

The values LN, MN, Z, MP, and LP represent Large Negative, Medium Negative, Close to Zero, Medium Positive, and Large Positive, respectively.

## System Design Fuzzy Logic

## Defuzzification

Upon initial research, the most popular method for defuzzification seemed to involve finding the centroid of the area under the output graph. This method seemed overly complex, involving expressing the output data as a mathematical curve and the summation of several integrations (Topperly, 2020). This was deemed too complex a solution to implement and therefore out of scope, so alternative methods were devised.

An excel spreadsheet prototype[[1]](#footnote-1) was created to test different defuzzification algorithms. The devised algorithms were as follows.

#### Clamped weighted sum

This method simply takes the input values and multiplies them by the value of their membership category   
(-1, -0.5, 0, 0.5, 1) for (LN, ML, Z, MP, LP) respectively. It then clamps this between [-1, 1].

#### Weighted average

This method takes a weighted sum (as described above) then takes the mean average of these values.

#### Exact value from biggest input

Simply take the value of the category that has the maximum value.

#### Weighted sum by input magnitude

Weighted sum divided by Pythagorean magnitude of the input values.

#### Weighted sum by input sum

Also known as centre of mass formula for point masses (isaacphysics, n.d.). This formula takes the weighed sum of the inputs and divides them by the unweighted sum of the inputs.

### Graphs

## Code Implementation – Fuzzy System

# Results

## Raw Data

## Direct Comparison

## Adjustments and Results

# Discussion

## Results Overveiw

## Causes

## Outliers and Adjustments

## Explanations of failures

# Conclusion

## Hypothesis

## Critical Analysis

# References

Cintula, P. C. G. F. a. C. N., 2017. *Fuzzy Logic.* [Online]   
Available at: https://plato.stanford.edu/archives/fall2017/entries/logic-fuzzy  
[Accessed 2021].

Deep AI, n.d. *Rule Based System.* [Online]   
Available at: https://deepai.org/machine-learning-glossary-and-terms/rule-based-system  
[Accessed 2021].

Guru99, n.d. *Fuzzy Logic Tutorial: What is, Architecture, Application, Example.* [Online]   
Available at: https://www.guru99.com/what-is-fuzzy-logic.html#:~:text=The%20term%20fuzzy%20logic%20was,from%20control%20theory%20to%20AI.  
[Accessed 2021].

isaacphysics, n.d. *Centre of Mass.* [Online]   
Available at: https://isaacphysics.org/concepts/cp\_centre\_mass  
[Accessed 2021].

ISO, 2020. ISO/IEC/IEEE International Standard - Floating-point arithmetic. *ISO/IEC 60559:2020(E) IEEE Std 754-2019,* pp. 1-86.

Topperly, 2020. *Solved Examples - Defuzzification Method | Fuzzy Logic.* [Online]   
Available at: https://www.youtube.com/watch?v=jhCFouPg3SA  
[Accessed 2021].

Tutorials Point, n.d. *Artificial Intelligence - Fuzzy Logic Systems.* [Online]   
Available at: https://www.tutorialspoint.com/artificial\_intelligence/artificial\_intelligence\_fuzzy\_logic\_systems.htm  
[Accessed 2021].

Unity Technologies, 2021. *Karting Microgame.* [Online]   
Available at: https://assetstore.unity.com/packages/templates/karting-microgame-150956  
[Accessed 2021].

Zadeh, L., 1965. Fuzzy sets. *Information and Control,* 8(3), pp. 338-353.

1. please see “Defuzzification Methods Experimentation.xlsx” included with this submission [↑](#footnote-ref-1)