# Complex Game Systems – Design

Fuzzy Logic State Machine Unity Proposal

By Jacob Cooper

# System

The fuzzy state machine is an extension to standard finite-state machines where it uses fuzzy logic to determine behaviours. Instead of processing crisp values, both the inputs and outputs are fuzzy values. This is advantageous for AI, as its intelligence comes from the feeling of the ai making choices based on the graphical formulars used to determine the desired desire level.

The goal is to implement a modular Fuzzy State Machine into unity where the user can create and use custom states ([see here](#_Integration)) to manipulate the algorithm a give the outcome of the AI a more realistic taste.

# Mathematical Operations

The mathematical order see [Fuzzy Inference System](#_Fuzzy_Inference_System).

## Logical Operators:

The fuzzy "and" is written as:

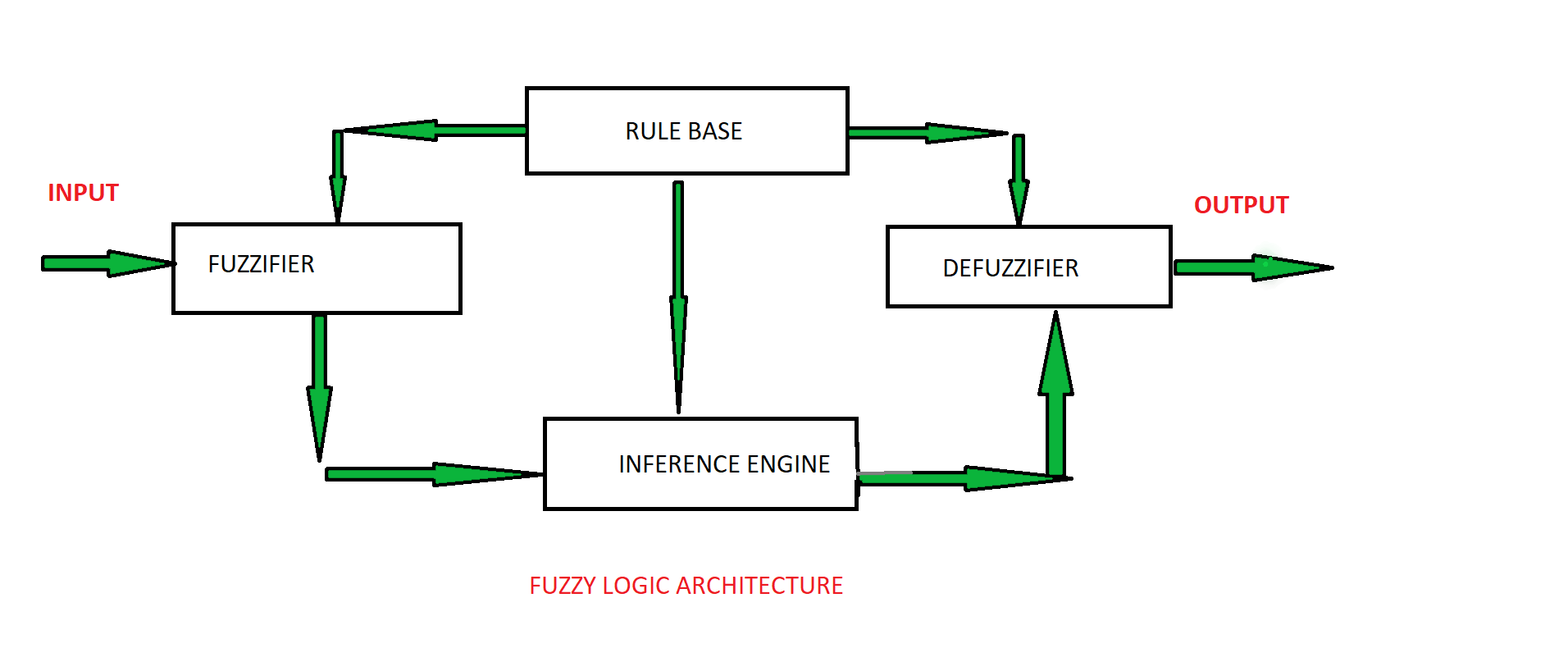


And “or” is written as:  
Undisplayed Graphic

Where µA is read as "the membership in class A" and µB is read as "the membership in class B". There way I will be computing “and” and “or” will be using Zadeh’s methods (the most common):

1. min(uA(x), uB(x)) - By taking the minimum of the two (or more) membership values.
2. max(uA(x), uB(x)) - This technique computes the fuzzy "or" by taking the maximum of the two (or more) membership values.

# Advanced Algorithms

A fuzzy state machine algorithm will have to be included which calculates fuzziness and states all under a set of rules and then outputs the correct desirable state to the AI to execute. 

## What is crisp/fuzzy/membership?

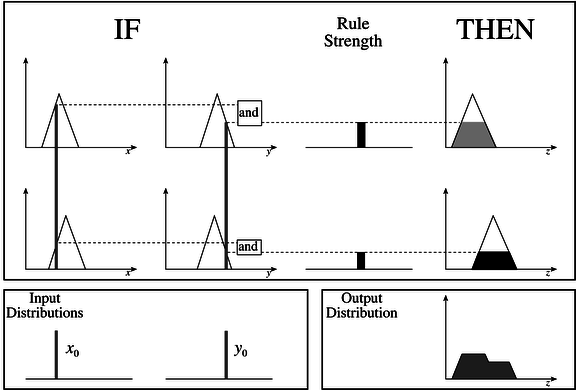
Crisp sets are simply a true or false, black and white interaction, the element is either a member of the set or not. However, Fuzzy sets allow elements to be *partially* in a set. Each element is given a degree of membership in a set. This membership value can range from 0 (not an element of the set) to 1 (a member of the set). A membership function is the relationship between the values of an element and its degree of membership in a set.

## Fuzzy Inference System (Mamdani)

I will be hopefully trying to implement the Mamdani Fuzzy Inference system. The order of operations are shown below:

1. Determining a set of fuzzy rules.
2. Fuzzifying the inputs using the input membership functions.
3. Combining the fuzzified inputs according to the fuzzy rules to establish a rule strength.
4. Finding the consequence of the rule by combining the rule strength and the output membership function.
5. Combining the consequences to get an output distribution.
6. Defuzzifying the output distribution (this step is only if a crisp output class is needed, don’t know yet).

The following is a more detailed description of this process.



## Creating fuzzy rules

Fuzzy rules are found in Mathematical Operations under logic, always written in this form:  
***if*** *(input1 is membership function1)* ***and/or*** *(input2 is membership function2)* ***and/or*** *�.* ***then*** *(output is output membership function).*

For example:

***If*** *temperature is high* ***and*** *humidity is high* ***then*** *room is hot.*

There would have to be a membership functions that define what we mean by high temperature (input1), high humidity (input2) and a hot room (output 1). This process of taking an input such as temperature and processing it through a membership function to determine what we mean by “high” temperature called fuzzification.

# Modularity

To make my system modular, I will be creating a Fuzzy State Machine object which can be edited using a custom node graph editor which will then be implemented onto objects.

The user will be able to inherit from two node classes to manipulate the graph editor/AI which include:

* FuzzyLogic – Create your own fuzzy logic script which allows manipulation of states and variables inputted into the script.
* FuzzyState – Fuzzy state which will be manipulated by the fuzzy logic script, which then will be passed off to an AI script which will execute this state during runtime.

Neither of these classes will inherit mono behaviour as their executions exist solely through an AI script.

The graph editor will give the user the ability to create states, a fuzzy logic calculator, and variable inputs which will all output necessary information which can be accessed later by another script which uses this component. Allowing for a customizable fuzzy state machine which can be implemented anywhere and used by any AI.

The graph editor is created using *Graph View* a unity experimental backend library which assisted in the development of the shader graph editor. The editor will be opened up through Windows/Fuzzy State Machine Editor or by simply double clicking the Fuzzy State Machine object.

The graphs object data will be mounted onto a loader component which can be accessed by getting the fuzzy state machine loader component on the game object,where you can grab/manipulate fuzzy data, in order to simulate fuzzy logic on an AI.

Hopefully later in development if there is time left, there will be a Fuzzy Logic Graph visualiser which will during runtime output a desirability graph chart in the component inspector to visualise the next most likely fuzzy logic state outcome.

This system will be exported into a unity package and can be loaded in any unity application by simply importing it.

# Integration

Integration of my system will be intricate, and will include:

## Importing

To import the *“Fuzzy State Machine.unitypackage”*, you can double click the package or in unity, go to Assets/Import Package and import the package.

## Creation

To create a fuzzy state machine, you right click on the inspector, “*Superior/FuzzyStateMachine/New Graph”*, once down you can open it in the Fuzzy State Machine editor by double clicking it or dragging it into the editor itself found in “*Windows/FSME”*. Once created and opened in the editor, right clicking the background will prompt a popup where you can crate one of three machine controllers under the *“Create/”* tab which include Variable, State and Logic.

The three types of state controllers are easy to use, you create a Fuzzy Variable in the inspector by right clicking and navigating to *“Superior/FuzzyStateMachine/Variable”*. Once created with its data you can drag it into a fuzzy variable controller, this will act as a variable inputter from outside, and a variable output for the inside, very useful for the state logic later on. FuzzyState is done in a similar fashion, however you will need to make a new script and inherit “*Superior.FuzzyStateMachine.FuzzyState”* for the editor to pick it up, this allows for highly custom AI states, for fuzzy logic node this is exactly the same, but you inherit *“Superior.FuzzyStateMachine.FuzzyLogic”* instead.

When creating state’s and logic the goal is that your script is fully reusable on any AI, as the state/logic doesn’t impact the game itself, it impacts the script that reads and uses the data.

## Integrating Scripts

On the game object you want to implement AI, you add the *“Fuzzy State Machine Loader”* component to it, then drag your recently created fuzzy state machine graph into the loader or click the drop down and find the graph you wish to use, once mounted, upon runtime, all of the states data is exposed, however it will not run or do anything as of yet.

Inside of the Graph Editor there will be a final node called *“Output”* this will output fuzzy data and fuzzy states out to the component loader.

You will need a script that grabs the fuzzy loader component *“FuzzyStateMachineLoader”*, and by using *SetVariable(“{variable name}”, variable data)* to set data, *GetState()* to grab the current decision of the state machine and *Calculate()* to produce a state to be used by GetState, after calculating and getting a desirable state you can run the state on your AI and perform such task.

You can also use *GetData()* which will return all the variables fuzzified though the fuzzy logic, this is useful for visualisation of the state machine.

# Additional Libraries

The project will be using the Unity Version 2020.1.10f. It will require the use of the experimental backend library of Graph View which is already included in unity which will be used to create the fuzzy state machine node graph.

# Useful Links

[https://www.cs.princeton.edu/courses/archive/fall07/cos436/HIDDEN/Knapp/fuzzy001.htm#0](https://www.cs.princeton.edu/courses/archive/fall07/cos436/HIDDEN/Knapp/fuzzy001.htm%230)

<http://www.cs.cmu.edu/Groups/AI/html/faqs/ai/fuzzy/part1/faq.html>