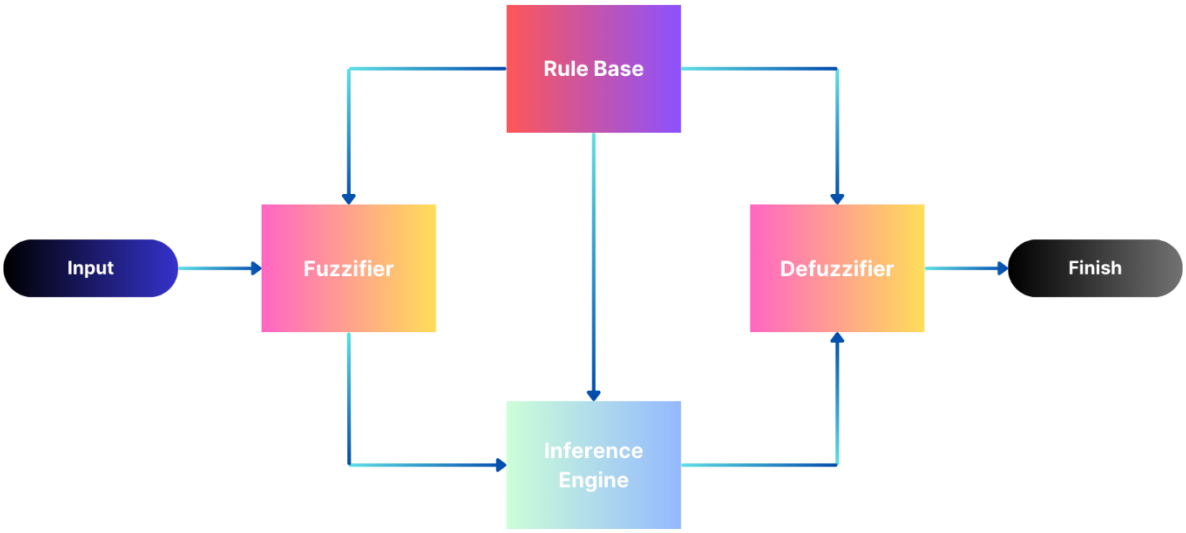
Assignment 1

We are going to see the implementation of fuzzy logic. But before we get into it, what is fuzzy logic? Fuzzy logic is a form of multi-valued logic that deals with reasoning that is approximate rather than fixed and exact. It is designed to handle vagueness and uncertainty, making it suitable for situations where precise mathematical models may be difficult to define. Fuzzy logic algorithms help solve problems by considering all available data and taking the best possible decision for the given input. They imitate the way humans make decisions, which often involve partial truths and intermediate possibilities. Fuzzy logic is based on the concept of membership functions and is implemented using fuzzy rules, which are if-then statements that express the relationship between input variables and output variables in a fuzzy way. The output of a fuzzy logic system is a fuzzy set, which is a set of membership degrees for each possible output value.

Architecture:



1. Rule Base: The rule base contains the rules and membership functions that regulate or control decision-making in the fuzzy logic system. It includes the IF-THEN conditions used for conditional programming and controlling the system. The rules are provided by experts and are based on linguistic information.
2. Fuzzifier: The fuzzifier component transforms raw inputs into fuzzy sets. It converts crisp input values into fuzzy sets, which are then processed by the control system. The fuzzification process involves mapping input values to fuzzy sets based on membership functions.
3. Inference Engine: The inference engine aids in determining the degree of compatibility among fuzzy inputs and rules. It selects which rules must be executed for the current input field based on the percent match. The applicable rules are then integrated to produce the control actions. In a fuzzy logic system, the inference system and the database combined are referred to as the controller.
4. Defuzzifier: The control system processes the fuzzy sets and rules to produce a certain output. It integrates the fuzzy sets and rules to make decisions based on the input data and the defined rules. The output of the control system is a fuzzy set, which represents the system's decision or action.

Application:

a simple fuzzy logic-based application for vehicle speed control using the scikit-fuzzy library in Python. It defines membership functions, rules, and simulates the control system to compute the acceleration based on input speed and distance values to control the acceleration of a vehicle based on input speed and distance values. Fuzzy logic allows for approximate reasoning and decision-making, making it suitable for control systems where precise mathematical models may be difficult to define.

Implementation:

Import necessary libraries

```
import numpy as np
import skfuzzy as fuzz
from skfuzzy import control as ctrl
```

✓ 2.2s

```
# Create Antecedent and Consequent objects ···
```

Define membership functions

```
speed['low'] = fuzz.trimf(speed.universe, [0, 30, 60])
speed['medium'] = fuzz.trimf(speed.universe, [40, 70, 100])
speed['high'] = fuzz.trimf(speed.universe, [80, 110, 130])

distance['close'] = fuzz.trimf(distance.universe, [0, 50, 100])
distance['medium'] = fuzz.trimf(distance.universe, [50, 100, 150])
distance['far'] = fuzz.trimf(distance.universe, [100, 150, 200])

acceleration['low'] = fuzz.trimf(acceleration.universe, [0, 0.3, 0.6])
acceleration['medium'] = fuzz.trimf(acceleration.universe, [0.4, 0.5, 0.7])
acceleration['high'] = fuzz.trimf(acceleration.universe, [0.6, 0.8, 1.0])
```

✓ 0.0s

Define rules

```
rule1 = ctrl.Rule(speed['low'] & distance['close'], acceleration['high'])
rule2 = ctrl.Rule(speed['medium'] & distance['medium'], acceleration['medium'])
rule3 = ctrl.Rule(speed['high'] & distance['far'], acceleration['low'])
```

✓ 0.0s

Create control system

```
acceleration_ctrl = ctrl.ControlSystem([rule1, rule2, rule3])
acceleration_simulation = ctrl.ControlSystemSimulation(acceleration_ctrl)
```

✓ 0.0s

Input values

```
acceleration_simulation.input['speed'] = 70
acceleration_simulation.input['distance'] = 120
```

✓ 0.0s

Compute acceleration

```
acceleration_simulation.compute()
```

✓ 0.0s

Output

```
print(acceleration_simulation.output['acceleration'])
```

✓ 0.0s

```
0.537142857142857
```

Explaination:

1. Importing Libraries: The code begins by importing the necessary libraries, including numpy and skfuzzy.

2. Creating Antecedent and Consequent Objects: Antecedent and Consequent objects are created to represent the input which in this case are speed, distance and output which will be calculated here. This is acceleration in this case.

3. Defining Membership Functions: Membership functions are defined for the input and output variables. These functions represent the degree of membership of a value in a fuzzy set. The degree of membership of a value ranges from 0 to 1 in a fuzzy set. This tells us the how much associated a value is to a fuzzy set.

4. Defining Rules: Rules are defined to map the input variables to the output variable based on fuzzy logic. These rules capture the fuzzy relationships between the input and output variables

5. Creating Control System: A control system is created using the defined rules and membership functions. This control system represents the fuzzy inference system.

6. Computing acceleration by simulating the control system: The control system is simulated with specific input values for speed and distance. The output acceleration is computed based on the fuzzy inference system's rules and membership functions.

7. Output: The computed acceleration value is printed as the output of the fuzzy logic-based system.

Logic:

The logic behind this application is to use fuzzy logic to control the acceleration of a vehicle based on input speed and distance values. Fuzzy logic allows for approximate reasoning and decision-making, making it suitable for control systems where precise mathematical models may be difficult to define. The rules in the fuzzy inference system capture the relationships between the input and output variables, allowing the system to reason with imprecise, incomplete, or distorted data and make decisions that resemble human decision-making.

References:

1. https://www.researchgate.net/publication/325165961_A_fuzzy_logic-based_anticipation_car-following_model
2. https://pythongeeks.org/fuzzy-logic-system-in-ai/
3. https://www.youtube.com/watch?app=desktop&v=rln_kZbYaWc
4. https://scikit-fuzzy.readthedocs.io/en/latest/
5. https://pypi.org/project/scikit-fuzzy/

Thank you