Name: Kaushik Kotian **Class: D20B/29**

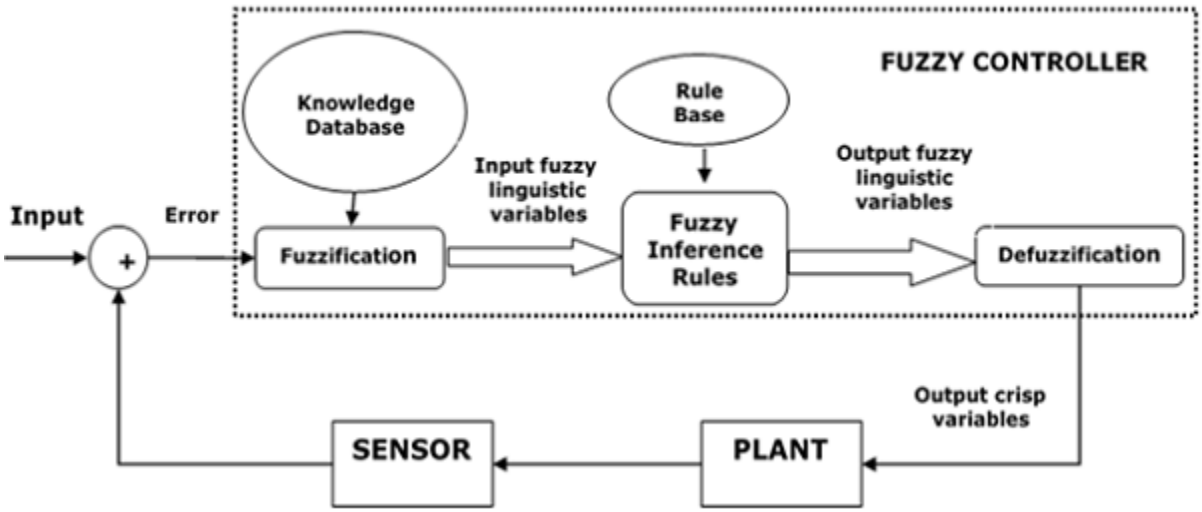
AIDS Lab EXPERIMENT NO. 06

Aim: To design a Fuzzy control system using Fuzzy tool/library.

Theory:

Fuzzy Logic Control (FLC) represents a vibrant area of research that leverages fuzzy set theory, fuzzy reasoning, and fuzzy logic. Its applications span a wide array of fields, from industrial process management to biomedical devices and financial systems. Unlike traditional control methods, FLC excels in addressing complex and poorly defined problems, often managed effectively by skilled human operators who may not fully understand the system's underlying dynamics.

The following diagram shows the architecture of Fuzzy Logic Control (FLC).



Followings are the major components of the Fuzzy logic control as shown in the above figure:

1. Fuzzifier: This component transforms crisp input values into fuzzy values, enabling the system to interpret real-world data.
2. Fuzzy Knowledge Base: This repository contains knowledge about the fuzzy relationships between inputs and outputs, including the membership functions that characterize the input and output variables relevant to the controlled system.
3. Fuzzy Rule Base: This element encapsulates the operational knowledge of the specific domain, outlining the rules that guide decision-making.
4. Inference Engine: Serving as the core of the FLC, the inference engine mimics human reasoning by performing approximate reasoning based on the fuzzy rules.
5. Defuzzifier: This component converts the fuzzy outputs generated by the inference engine back into crisp values for practical application.

The design process for a Fuzzy Logic Control system involves several key steps:

1. Identification of Variables: Determine the input, output, and state variables pertinent to the system being analyzed.
2. Fuzzy Subset Configuration: Divide the universe of discourse into multiple fuzzy subsets, each labeled with a linguistic term, ensuring comprehensive coverage of all elements.
3. Obtaining Membership Functions: Develop membership functions for each fuzzy subset identified in the previous step.
4. Fuzzy Rule Base Configuration: Create a fuzzy rule base that establishes relationships between the fuzzy inputs and outputs.
5. Fuzzification: Initiate the fuzzification process to translate crisp inputs into fuzzy values.
6. Combining Fuzzy Outputs: Utilize fuzzy approximate reasoning to identify and merge the fuzzy outputs.
7. Defuzzification: Conclude the process by converting the fuzzy outputs into a crisp output.

Fuzzy Logic Control systems are widely utilized across various industrial and commercial sectors. They have demonstrated significant effectiveness, particularly in managing nonlinear, time-varying, and complex systems, often outperforming traditional control approaches.

Code and Output:

```
!pip install scikit-fuzzy
from skfuzzy import control as ctrl
import skfuzzy as fuzz
import numpy as np
```

Using scikit-fuzzy we will generate a Control System that will estimate how long it will take to wash one load of clothes. Our inputs will be known as Antecedents and Outputs will be known as Consequents in a scikit-fuzzy controller.

Antecedents (Inputs):

1. type_of_dirtiness:

- Universe (crisp value range): Type of dirtiness in terms of percentage 1 to 100
- Fuzzy set (fuzzy value range): NonFat, Medium, Fat

2. degree_of_dirtiness:

- Universe (crisp value range): Degree of dirtiness in terms of percentage 1 to 100
- Fuzzy set (fuzzy value range): Low, Medium, Fat

Consequents (Outputs):

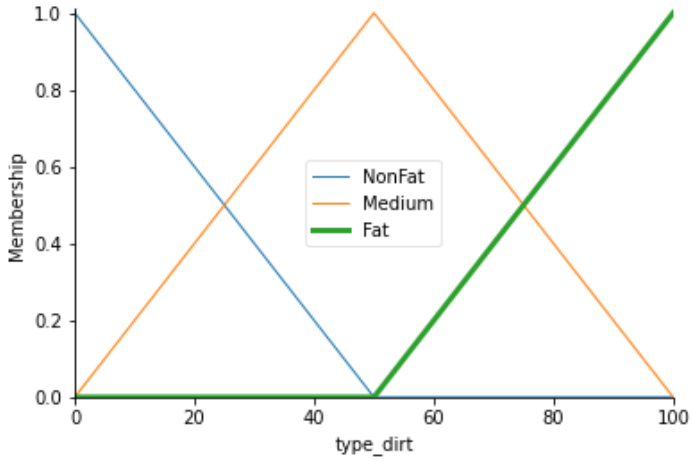
Name: Kaushik Kotian **Class: D20B/29**

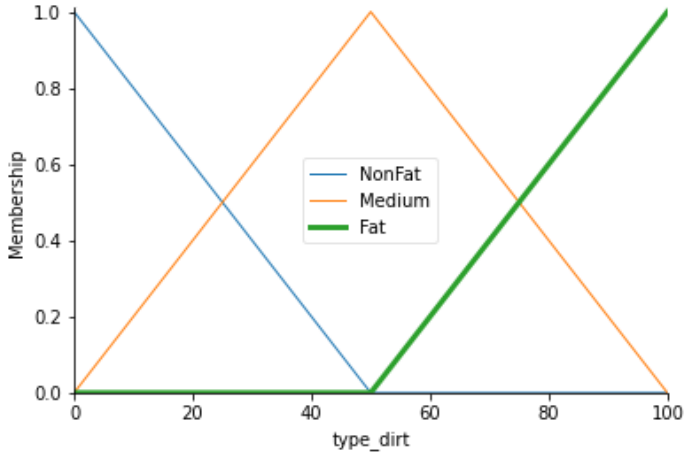
1. wash_time:

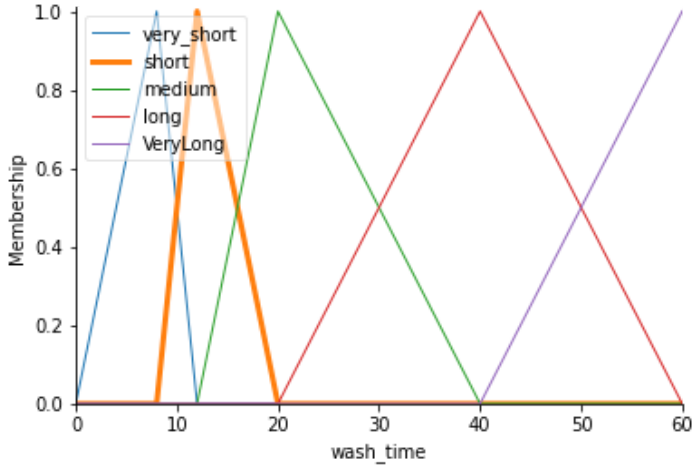
- Universe: According to type_of_dirtiness and degree_of_dirtiness program will determine how long it would take to wash one load of clothes. Output is generated in the format of minutes between (1 to 60)
- Fuzzy set (fuzzy value range): VeryShort, Short, Medium, Long, VeryLong

```
class washing_machine:
        degree_dirt = ctrl.Antecedent(np.arange(0, 101, 1), 'degree_dirt')
        type_dirt = ctrl.Antecedent(np.arange(0, 101, 1), 'type_dirt')
        wash_time = ctrl.Consequent(np.arange(0, 61, 1), 'wash_time')
        degree_names = ['Low', 'Medium', 'High']
        type_names = ['NonFat', 'Medium', 'Fat']

        # Outputting them into auto-membership functions
        degree_dirt.automf(names=degree_names)
        type_dirt.automf(names=type_names)

        # You can see how these look with .view()
        degree_dirt['Medium'].view()
        type_dirt['Fat'].view()

        # Washing Time Universe
        wash_time['very_short'] = fuzz.trimf(wash_time.universe, [0, 8, 12])
        wash_time['short'] = fuzz.trimf(wash_time.universe, [8, 12, 20])
        wash_time['medium'] = fuzz.trimf(wash_time.universe, [12, 20, 40])
        wash_time['long'] = fuzz.trimf(wash_time.universe, [20, 40, 60])
        wash_time['VeryLong'] = fuzz.trimf(wash_time.universe, [40, 60, 60])
        wash_time['short'].view()

        # Rule Application
        rule1 = ctrl.Rule(degree_dirt['High'] | type_dirt['Fat'], wash_time['VeryLong'])
        rule2 = ctrl.Rule(degree_dirt['Medium'] | type_dirt['Fat'], wash_time['long'])
        rule3 = ctrl.Rule(degree_dirt['Low'] | type_dirt['Fat'], wash_time['long'])
        rule4 = ctrl.Rule(degree_dirt['High'] | type_dirt['Medium'], wash_time['long'])
        rule5 = ctrl.Rule(degree_dirt['Medium'] | type_dirt['Medium'], wash_time['medium'])
        rule6 = ctrl.Rule(degree_dirt['Low'] | type_dirt['Medium'], wash_time['medium'])
        rule7 = ctrl.Rule(degree_dirt['High'] | type_dirt['NonFat'], wash_time['medium'])
        rule8 = ctrl.Rule(degree_dirt['Medium'] | type_dirt['NonFat'], wash_time['short'])
        rule9 = ctrl.Rule(degree_dirt['Low'] | type_dirt['NonFat'], wash_time['very_short'])
        rule1.view()

        # Washing Control Simulation
```

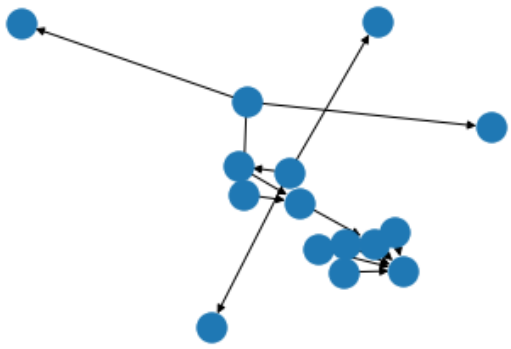
```
        washing_ctrl = ctrl.ControlSystem([rule1, rule2, rule3, rule4, rule5, rule6, rule7, rule8,
rule9])
        washing = ctrl.ControlSystemSimulation(washing_ctrl)
```







Name: Kaushik Kotian **Class: D20B/29**



```
def fuzzify_laundry(fuzz_type, fuzz_degree):
        washing_machine.washing.input['type_dirt'] = fuzz_type
        washing_machine.washing.input['degree_dirt'] = fuzz_degree
        washing_machine.washing.compute()
        washing_machine.wash_time.view(sim=washing_machine.washing)
        return washing_machine.washing.output['wash_time']
```

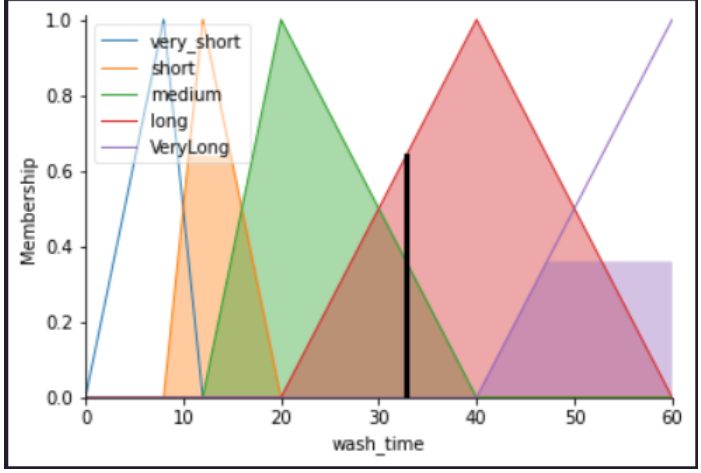
```
def compute_washing_parameters(type_of_dirt, degree_of_dirt):
        if type_of_dirt < 0.0 or type_of_dirt > 100.0:
        raise Exception("Invalid Type of Dirtiness: %lf" %type_of_dirt)
        if degree_of_dirt < 0.0 or type_of_dirt > 100.0:
        raise Exception("Invalid Degree of Dirtiness: %lf" %degree_of_dirt)
        type_fuzzy = fuzzify_laundry(type_of_dirt, degree_of_dirt)
        return type_fuzzy
```

Once the output is computed all together, we can visualize it.

```
if __name__ == "__main__":
        type_of_dirt = float(input("Enter Type of Dirtiness [0-100]"))
        degree_of_dirt = float(input("Enter Degree of Dirtiness [0-100]"))
        washing_parameters = compute_washing_parameters(type_of_dirt, degree_of_dirt)
        print(washing_parameters)
```

Name: Kaushik Kotian **Class: D20B/29**

```
Enter Type of Dirtiness [0-100]50
Enter Degree of Dirtiness [0-100]68
32.7886891153378
```



Inputs that we put in were type_of_dirtiness and degree_of_dirtiness around 50 and 68 respectively, according to that washing time is generated around approximately 32.7886891153378 minutes, we can round that up to 33 minutes.

Conclusion:
Thus we studied an overview of what Fuzzy Logic Control is, implemented the Fuzzy control system on a Washing Machine and found out the washing time generated after inputting type and degree of dirtiness.