Date: 23-10-2023

Artificial Intelligence and Its Applications

3rd Semester

Assignement-1

Q1. Explain inheritance in detail in knowledge representation.

Q2. There is a monkey at the door in a room. In the middle of the room a bunch of banana is hanging from the ceiling. The monkey is hungry and wants to get the banana. but he cannot stretch high enough from the floor. At the window of the room there is a box. Represent the information used in the above-mentioned problem in predicate logic.

Q3. Consider the following clausal form:
isa(X, living _ thing) <- isa(X, animate)
isa(X, animate) <- isa(X, human)
isa(X, human) <- isa(X, man)
isa(Jay, man)
Represent forward reasoning inference.

Q4. Explain in detail about forward chaining and backward chaining with algorithms and examples.

Q5. Design an Aircraft Landing Control Problem using a fuzzy control system.

NAME: KARAN BHATIA
BATCH: 110T-B1
ENROL.NO.: 01819011722

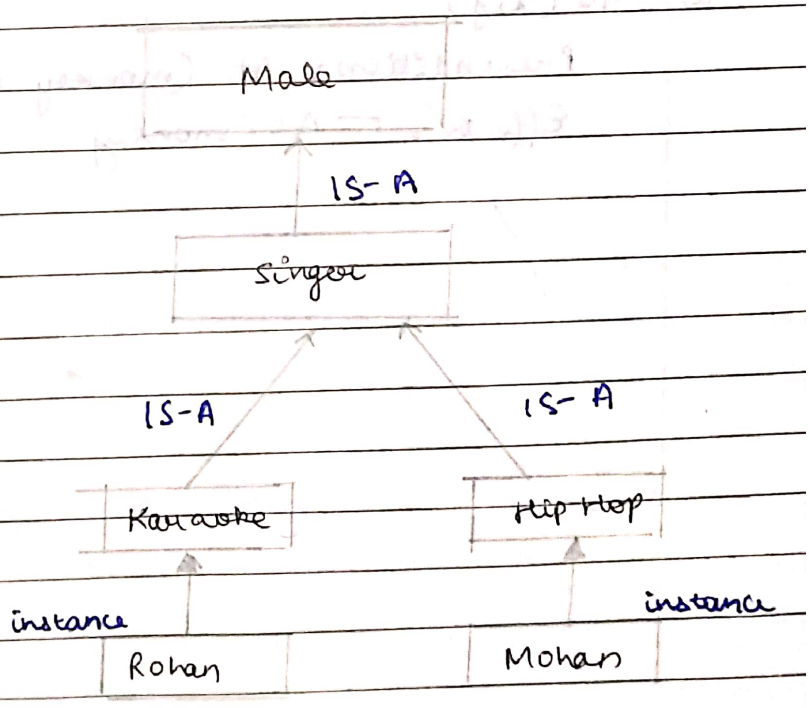
DOMS | Page No.
Date / /

Artificial Intelligence & Its Applications
Assignment-1

Q.1 Explain inheritance in detail in knowledge representation

(1) In inheritable knowledge all data must be stored into hierarchy of classes
(2) All classes should be arranged in generalized form of hierarchical manner
(3) In this approach we apply inheritance property
(4) Elements inherit values from other members of a class
(5) This approach contains inheritable knowledge which shows a relation b/w instance and class and it is called instance relation.
(6) Every individual frame can represent the collection of attributes and its value.
(7) In this approach, objects & values are represented in Boxed nodes.
(8) We use arrows which point from objects to their values

Example:



A°2 Let $M(x)$ represent "x is a monkey"
Let $B(x)$ represent "x is bunch of bananas"
Let $H(x)$ represent "x is hungry"
Let $F(x,y)$ represent "x is able to reach y"
Let $W(x)$ represent "x is a window"
Let $D(x)$ represent "x is a door"
Let $P(x)$ represent "x is a box"

1. There exists a monkey at the door:
$$\exists x (M(x) \wedge D(x))$$

2. A bunch of bananas is hanging from the ceiling:
$$\exists x\, B(x)$$

3. The monkey is hungry: $\exists x (M(x) \wedge H(x))$

4. The monkey cannot stretch high enough from the floor to reach the banana: $\forall x \forall y ((M(x) \wedge B(y)) \rightarrow \neg F(x,y))$

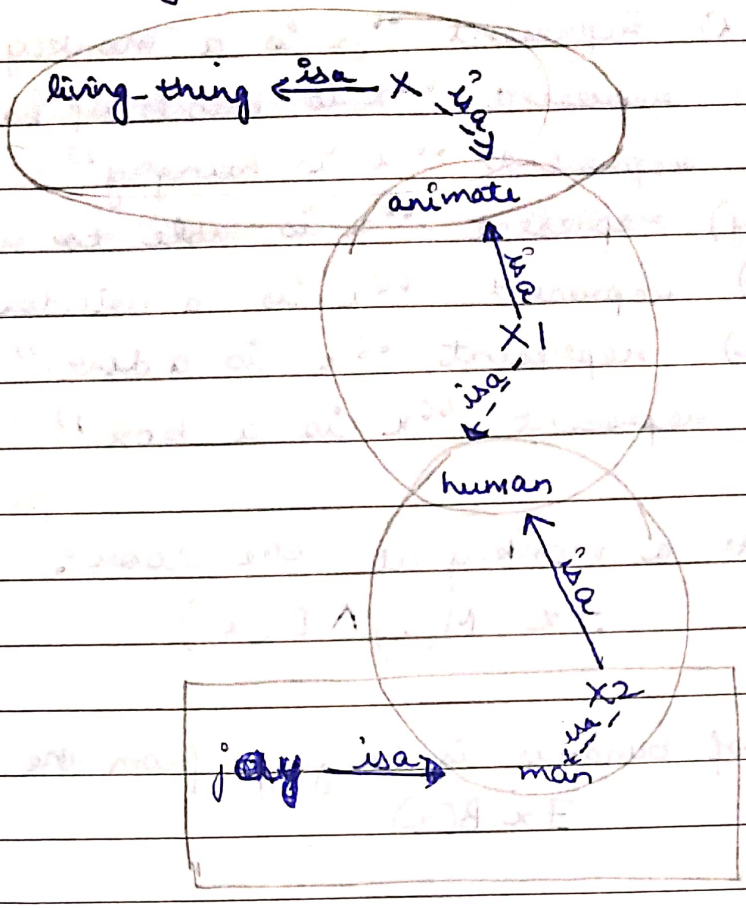
5. There is a box at the window: $\exists x (P(x) \wedge W(x))$

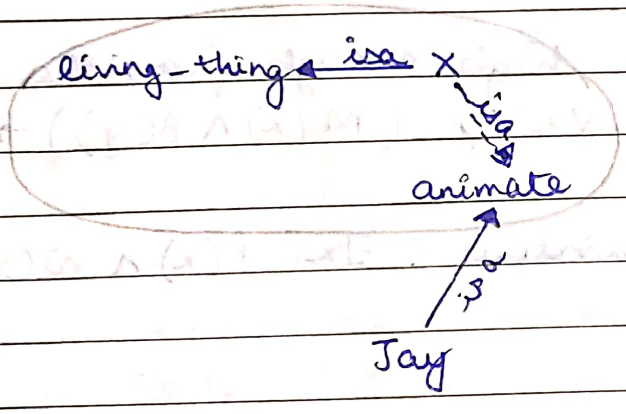
A·3 isa(X, living_thing) ← isa(X, animate)
isa(X, animate) ← isa(X, human)
isa(X, human) ← isa(X, man)
isa(Jay, man)

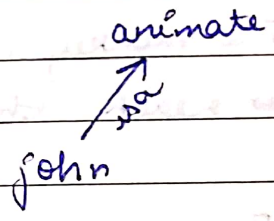
The steps for inferring that John is animate. The assertion johs is a living_thing that can be inferred similarly.

Date / /

Forward Reasoning Interference







Jay is animate: isa
(Jay, animate is inferred)

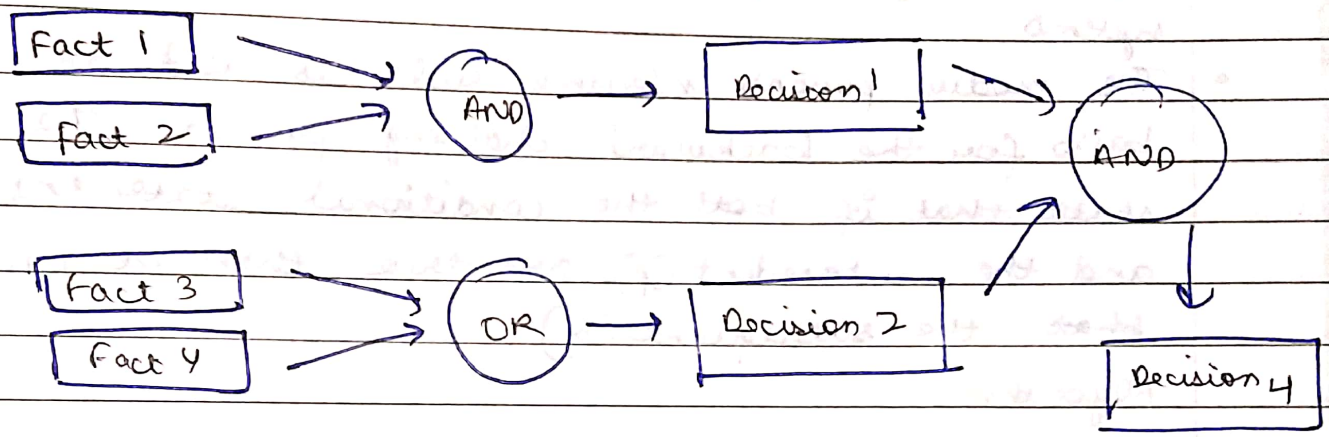
A·4 Forward Chaining: Forward chaining is the Inference Engine that goes through all the facts, conditions and derivations before deducing the outcome i.e. when based on available data a decision is taken then the process is called as Forward Chaining. It works from an initial state and reaches to the goal (final decision

DOMS | Page No.
Date / /

Properties of forward chaining

- The process uses a down up approach (bottom to top)
- It starts from initial state and uses facts to make a conclusion
- This approach is data-driven
- It's employed in expert systems & production rule system

Algorithm:



Example:- A

A → B

B

A is the starting point, A → B represents a fact. This fact is used to achieve a decision B

Tom is running (A)

If a person is running, he will sweat (A → B)

Therefore Tom is sweating (B)

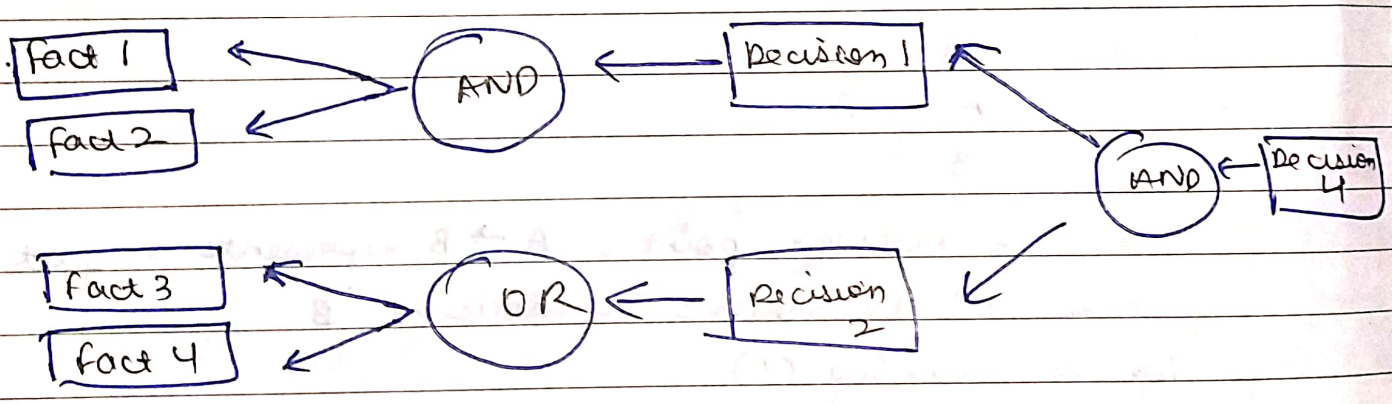
Backward chaining: In this the inference system knows the final decision or goal, this system starts from the goal and works backwards to determine what facts must be asserted so that the goal can be achieved i.e. it works from goal (final decision) and reaches the initial state

Page No.
Date / /

Properties of backward chaining

- The process uses an up-down approach (top to bottom)
- It is a goal driven method
- The endpoint (goal) is subdivided into sub-goals to prove the truth of facts
- A backward chaining algorithm is employed in inference engines, game theories & complex database systems.
- The modus ponens inference rule is used as the basis for the backward chaining process. The rule states that if both the conditional statement $(p \rightarrow q)$ and the antecedent (p) are true then we can infer the subsequent (q)

Algorithm:



Example: B

$A \rightarrow B$

A

B is the goal or endpoint that is used as starting point for backward tracking. A is the initial state. $A \rightarrow B$ is a fact that must be asserted to arrive at endpoint B

Tom is sweating (B)

If a person is running he will sweat $(A \rightarrow B)$

Tom is running (A)

A·5 We will conduct a simulation of final descent and landing approach of an aircraft. The desired profile is shown in the figure. The desired downward velocity is proportional to the square of the height. Thus, at higher altitudes, a large downward velocity is desired.

As the height diminishes the desired downward velocity gets smaller & smaller. In the limit the height becomes vanishingly small, the downward velocity also goes to zero. In this way the aircraft will descend from altitude promptly but will touch down very gently to avoid damage.

The two state variables for this simulation will be the height above ground, h, and the vertical velocity of the aircraft, v. The control output will be a force that when applied to the aircraft will alter its height h and velocity v. The differential control eqns are loosely derived as follows. $\Delta v = f \Delta t / m$

We get $v_{i+1} = v_i + f_i$

$h_{i+1} = h_i + v_i \Delta t$

where v_{i+1} is the new velocity, v_i is the old velocity h_{i+1} is the new height and h_i is the old height

Step-1 → Define membership functions for state variables as shown in the tables & figures

Step-2 → Define a membership function for the control output as shown in the Table & figure

Step-3 → Define the rules & summarize them in an FAM table. The values in the FAM table, of course, are the control outputs.

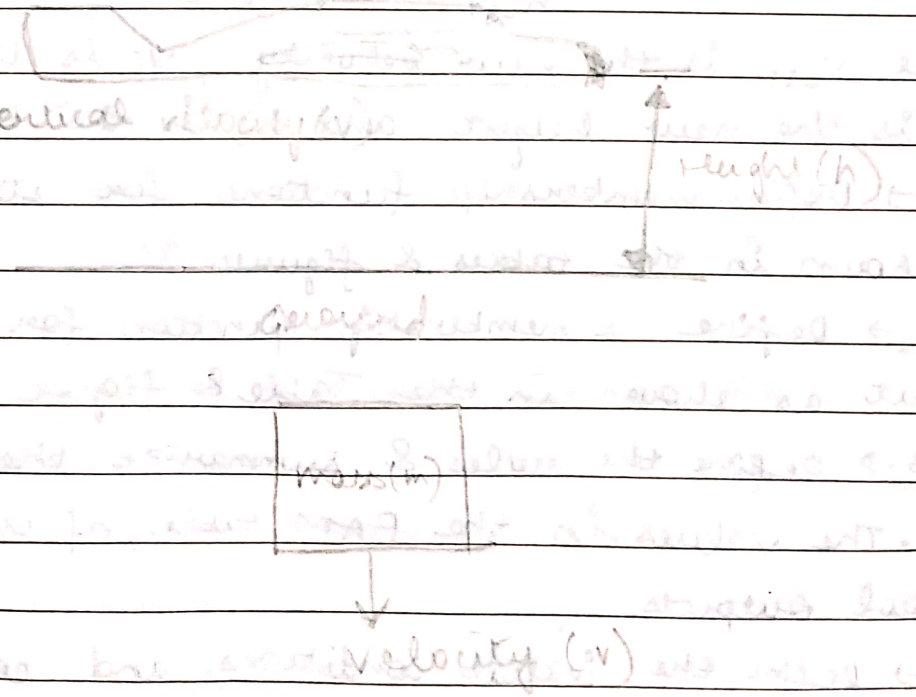
Step-4 → Define the initial conditions, and conduct a simulation for 4 cycles. Because the task at hand is to control the aircraft's vertical descent during approach & landing.

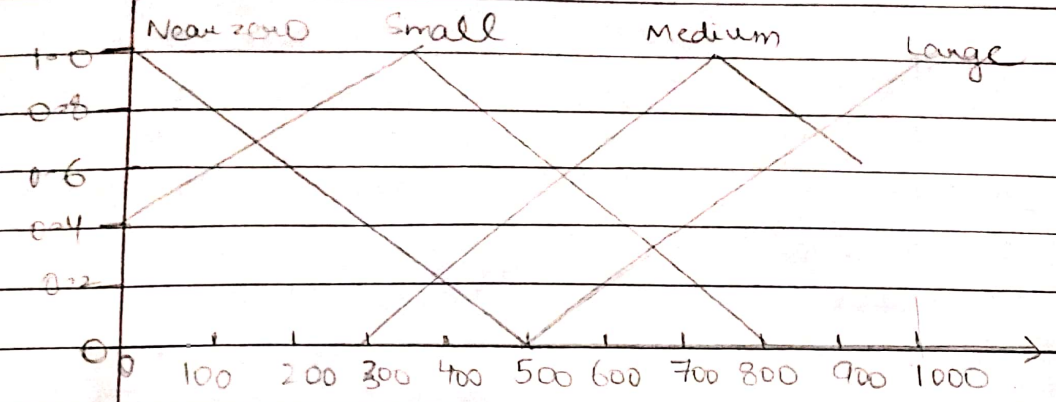
Membership values for Height

	Height (ft)										
	0	100	200	300	400	500	600	700	800	900	1000
Large (L)	0	0	0	0	0	0	0.2	0.4	0.6	0.8	1
Medium (M)	0	0	0	0	0.2	0.4	0.6	0.8	1	0.8	0.6
Small (S)	0.4	0.6	0.8	1	0.8	0.6	0.4	0.2	0	0	0
Nearzero (Z)	1	0.8	0.6	0.4	0.2	0	0	0	0	0	0

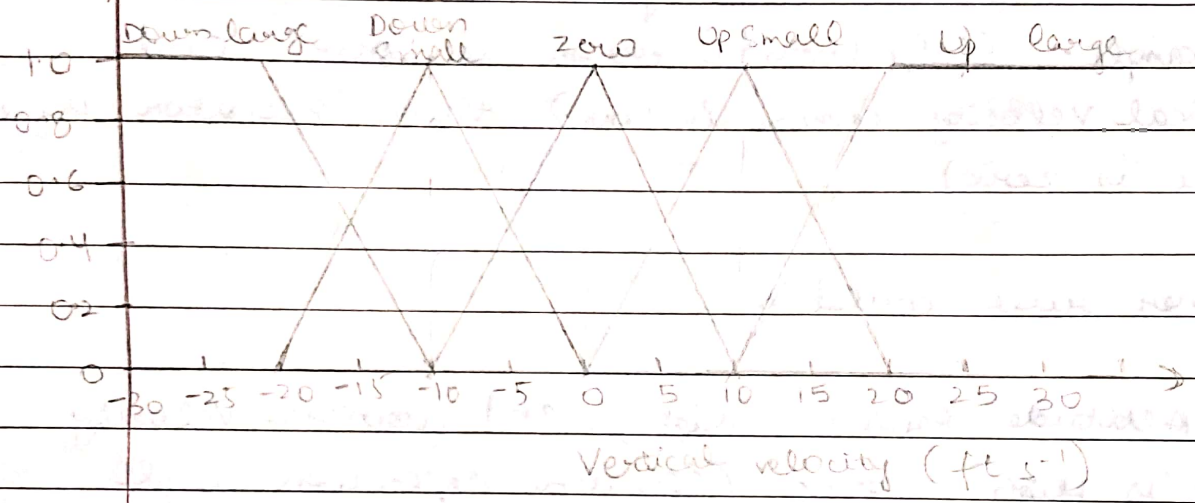
Membership values for velocity

	Vertical velocity (ft/s)												
	−30	−25	−20	−15	−10	−5	0	5	10	15	20	25	30
Uplarge (UL)	0	0	0	0	0	0	0	0	0	0.5	1	1	1
Upsmall (US)	0	0	0	0	0	0	0	0.5	1	0.5	0	0	0
Zero (Z)	0	0	0	0	0	0.5	1	0.5	0	0	0	0	0
Downsmall (DS)	0	0	0	0.5	1	0.5	0	0	0	0	0	0	0
Downlarge (DL)	1	1	1	0.5	0	0	0	0	0	0	0	0	0





Height, h, partitioned

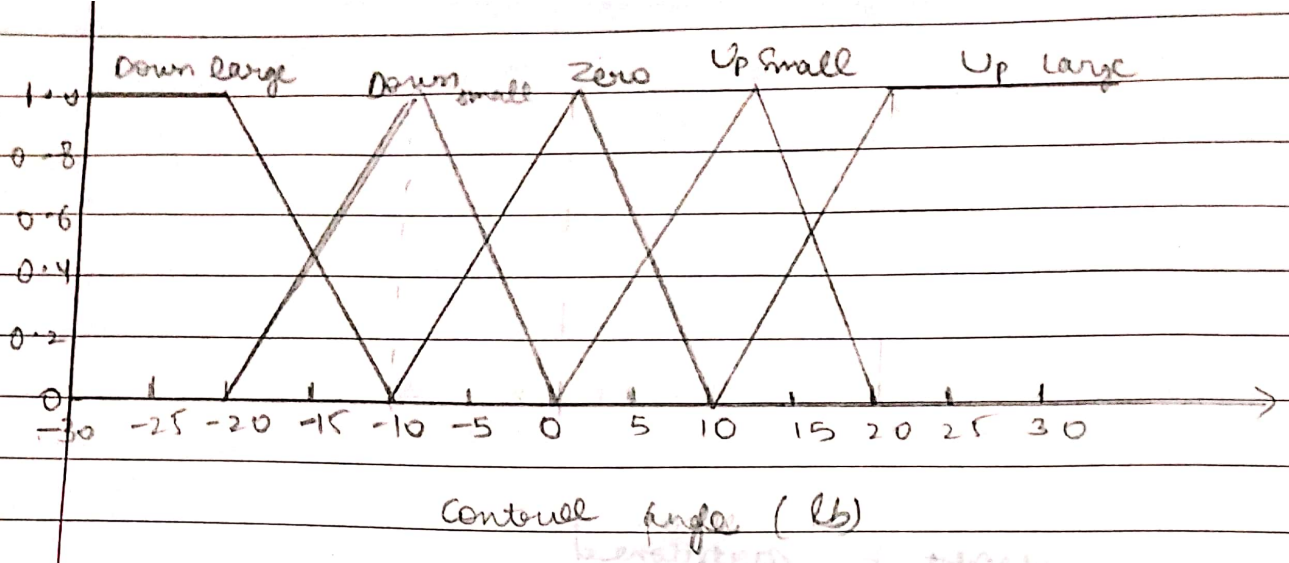


Velocity, v, partitioned

Membership values for ~~control force~~ elevator deflection angle

Output angle (b)

	-30	-25	-20	-15	-10	-5	0	5	10	15	20	25	30
Upscale (US)	0	0	0	0	0	0	0	0	0.5	1	1	1	
Upsmall (US)	0	0	0	0	0	0	0	0.5	0.5	0	0	0	
Zero (Z)	0	0	0	0	0	0.5	1	0.5	0	0	0	0	
Downsmall (DS)	0	0	0	0.5	1	0.5	0	0	0	0	0	0	
Downlarge (DL)	1	1	1	0.5	0	0	0	0	0	0	0	0	



3. Now we can define the fuzzy rules.
for example - If (Altitude Error is Low) and (Vertical Velocity Error is Low) then (Elevator Deflection Angle is zero)

Another rule could be:

If (Altitude Error is High) and (Vertical Velocity Error is High) then (Elevator deflection angle is negative)

4. Now we design the fuzzification Interface
The fuzzification interface converts the crisp input variables into fuzzy sets. This is done by assigning membership values to each fuzzy set for each input variable.
For example if altitude error is .3000 feet, then the membership value for the "Low" fuzzy set should be 0.5, the membership value for the "Medium" fuzzy set should be 1.0 and the membership value for the "High" fuzzy set would be 0.5

5. Design the Inference Engine

The inference engine evaluates the fuzzy rules & produces a fuzzy output. This is done by applying the fuzzy rules to the fuzzy input variables & then aggregating the results

6. Design the Defuzzification Interface

The defuzzification interface converts the fuzzy output into a crisp output. This is done by selecting a representative value from the fuzzy output set.

7. Implement the Fuzzy Controller

The fuzzy controller can be implemented using a variety of software & hardware platforms. Once the fuzzy controller is implemented, it can be tested in a simulation environment to ensure it meets desired performance requirements