Zadeh

February 24, 2023

1 Zadeh operators by using operator overloading

[1]:
```
import math
```

1.1 Fuzzy value

[2]:
```
class Zadeh:
    """
    Represents a fuzzy value
    """

    def __init__(self, value):
        self.value = value

    @property
    def value(self):
        return self._value

    @value.setter
    def value(self, value):
        if 0 <= value <= 1:
            self._value = value
        else:
            raise ValueError(f'The fuzzy value {value} is invalid!')

    def __and__(self, other):
        """
        T-norm
        """
        if self.value < other.value:
            return Zadeh(self.value)
        else:
            return Zadeh(other.value)

    def __or__(self, other):
        """
        S-norm
```

```
        """
        if self.value > other.value:
            return Zadeh(self.value)
        else:
            return Zadeh(other.value)

    def __str__(self):
        return f'{self.value}'

    def __repr__(self):
        return f'Zadeh({self.value})'

def not_(z):
    """
    Negation
    """
    return Zadeh(1 - z.value)
```

Example

[3]:
```
a = Zadeh(0.37)
b = Zadeh(0.23)
c = Zadeh(0.81)
```

[4]:
```
(a or b) and not_(c)
```

[4]:
```
Zadeh(0.18999999999999995)
```

1.2 Membership function

[5]:
```
class MembershipFunction:

    def __init__(self, table, value=None):
        self._table = table
        self.value = value

    def __eq__(self, other):
        if other not in self._table:
            raise ValueError(f'Undefined language variable! "{other}"')
        full_table = self._table | {'-inf': -math.inf, '+inf': math.inf}
        intervals = sorted(list(full_table.items()), key=lambda item: item[1])
        index = None
        i = 1
        while index is None:
            if intervals[i][0] == other:
                index = i
            i += 1
```

```
        m = 0
        if index == 1 and self.value <= intervals[1][1]:
            m = 1
        elif index == len(intervals) - 2 and self.value >= intervals[index][1]:
            m = 1
        elif intervals[index - 1][1] <= self.value < intervals[index][1]:
            m = (intervals[index][1] - self.value) / (intervals[index][1] -␣
↪intervals[index - 1][1])
        elif intervals[index][1] <= self.value < intervals[index + 1][1]:
            m = (self.value - intervals[index][1]) / (intervals[index + 1][1] -␣
↪intervals[index][1])
        return Zadeh(m)

    @property
    def value(self):
        return self._value

    @value.setter
    def value(self, value):
        self._value = value
```

Example

[6]:
```
battery = MembershipFunction({
    'empty': 0,
    'full': 100
})
```

[7]:
```
distance = MembershipFunction({
    'close': 0,
    'near': 5,
    'far': 20
})
```

[8]:
```
battery.value = 45
distance.value = 12
speed = (battery == 'empty') or (distance == 'close')
print(speed)
```

```
0.45
```

[9]:
```
battery.value = 70
distance.value = 3
speed = (battery == 'full') and not_(distance == 'close')
print(speed)
```

```
0.4
```