Proyecto de Simulación de Lógica Difusa

Pablo Antonio de Armas Suárez – C411

Contenido

Introducción

La Lógica Difusa se basa en la extensión del concepto de pertenencia a un conjunto. En la lógica bivalente (booleana) a la que estamos habituados, la pertencia a un conjunto se basa en una función que mapea elementos del universo de discurso a los valores 0 y 1, o sea si es 1, el elemento pertenece al conjunto y si es 0 no pertenece.

En los conjuntos difusos, vamos a tener una función que mapea los elemento del universo de discurso a los números reales en el intervalo [0, 1]. A partir de ahí, se construye un sistema matemático, que permite crear sistemas de inferencia difusa que tratan de manejar problemas de la vida real en los que los humanos acostumbramos a manejar con terminos poco precisos como:

- La temperatura del agua está caliente, tibia, o incluso fresca
- La comida está rica
- La atención del camarero fue pésima
- Y muchas más.

Como parte de este trabajo hemos implementado un Sistema de Inferencia basado en Lógica Difusa, extensible y versatil que permite:

- Utilizar la mayor parte de las funciones de pertenencia a conjuntos difusos utilizadas (Triangulares, Trapezoidales, Gaussianas, Bell, LeftRight y Sigmoid) y adicionar otras.
- Crear variables linguísticas y utilizarlas para definir reglas en una base de conocimiento.
- Utilizar los métodos de agregación de Mamdani y Larsen para evaluar e inferir.También es posibles crear otros métodos y utilizarlos sin hacer modificaciones.
- Utilizar diferentes métodos de desdifusificación (Centroid, Bisector, LOM y SOM) o agregar otros

Implementación

El sistema fue desarrollado en el lenguaje C# sobre la plataforma .NET. Se aprovecharon las facilidades del lenguaje C# para crear interfazas fluídas para permitir crear las reglas de una forma sencilla, directa y clara. De igual forma se usaron la Genericidad, los delegados (y expresiones lamdas) y las interfaces para lograr que el sistema sea extensible fácilmente.

Funciones de Pertenencia

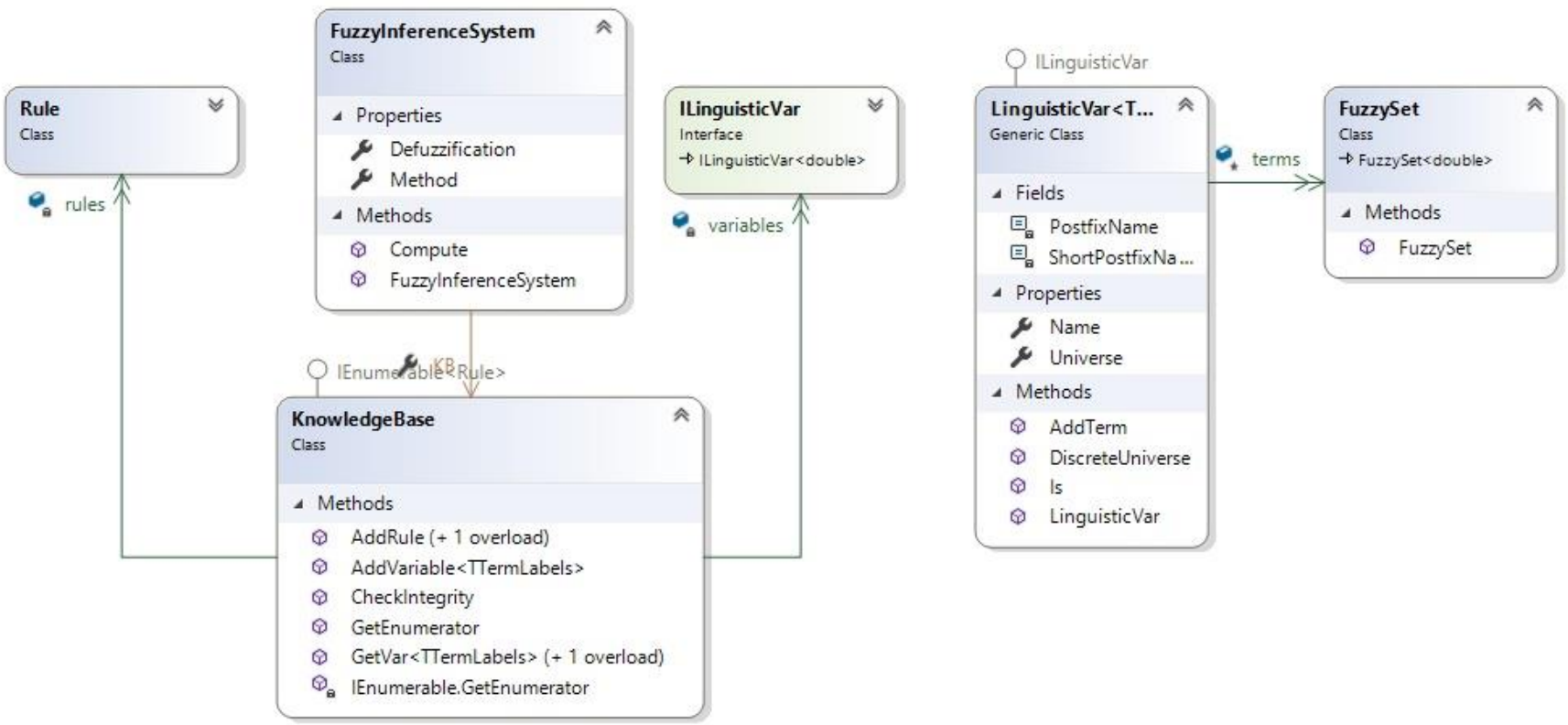
Uno de los elementos que le da versatilidad a la Lógica Difusa para modelar sistemas reales es la posibilidad de utilizar diferentes funciones de pertenencia a los conjuntos difusos. En este proyecto se implementaron las siguientes:

- Triangulares
- Trapezoidales
- Gausianas
- Campanas (Bell)
- LeftRight
- Sigmoid

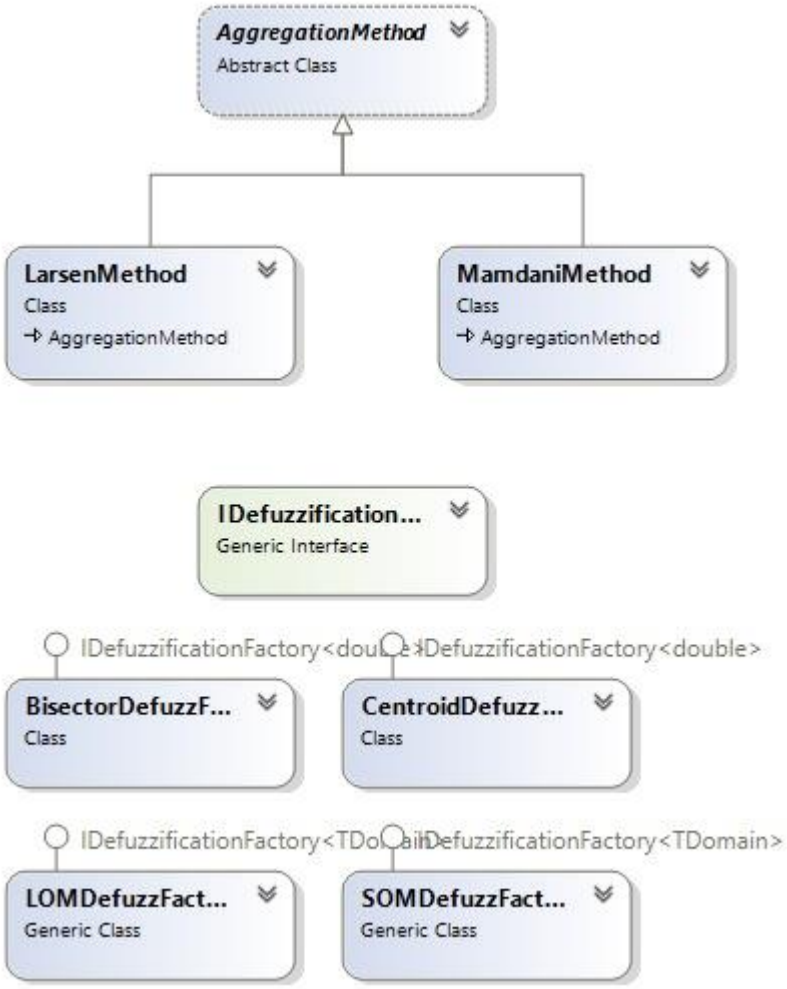
Las mismas pueden ser creadas fácilmente a través de factorías

```
var mfFactory = new TriangularFuncFactory();

var food = new LinguisticVar<FoodQuality>(
    LinguisticVar<FoodQuality>.DiscreteUniverse(0, 10, 1), "food")
    .AddTerm(FoodQuality.Bad, new FuzzySet(mfFactory.Create(0, 0, 5)))
    .AddTerm(FoodQuality.Decent, new FuzzySet(mfFactory.Create(0, 5, 10)))
    .AddTerm(FoodQuality.Great, new FuzzySet(mfFactory.Create(5, 10, 10)));
```



1- Clases principales del Sistema de Inferencia Difuso



2 Métodos de Agregación y de Desdifusificación implementados

Métodos de Desdifusificación

Para trabajar con los métodos de disfusificación se hace de manera parecida. Como ya se mencionó antes, se implementaron los siguientes métodos:

- Centroid
- Bisector
- Y de los métodos de Máximos LOM y SOM

Aplicación de Ejemplo

Para probar las facilidades del Sistema de Inferencia Fuzzy implementado, se modeló el conocido problema de “Dar Propina”. En el mismo a partir de la evaluación de la calidad del Servicio y la

comida, se desea conocer que por cierto dar de propina. Vamos entonces a tener 3 variables linguisticas:

- **Service** y **Food** que van a moverse en el rango de 1 a 10 y van a categorizarse con tres términos cada una.
- Por otra parte, vamos a tener la variable **Tip** para la propina que se moverá en el rango de 0 a 25% y que también se categorizará con tres términos.

```
7 references
public enum ServiceQuality { Poor, Acceptable, Amazing }

9 references
public enum FoodQuality { Bad, Decent, Great }

7 references
public enum Tip { Low, Medium, High }


var mfFactory = new TriangularFuncFactory();

var food = new LinguisticVar<FoodQuality>(
    LinguisticVar<FoodQuality>.DiscreteUniverse(0, 10, 1), "food")
    .AddTerm(FoodQuality.Bad, new FuzzySet(mfFactory.Create(0, 0, 5)))
    .AddTerm(FoodQuality.Decent, new FuzzySet(mfFactory.Create(0, 5, 10)))
    .AddTerm(FoodQuality.Great, new FuzzySet(mfFactory.Create(5, 10, 10)));

var service = new LinguisticVar<ServiceQuality>(
    LinguisticVar<FoodQuality>.DiscreteUniverse(0, 10, 1), "service")
    .AddTerm(ServiceQuality.Poor, new FuzzySet(mfFactory.Create(0, 0, 5)))
    .AddTerm(ServiceQuality.Acceptable, new FuzzySet(mfFactory.Create(0, 5, 10)))
    .AddTerm(ServiceQuality.Amazing, new FuzzySet(mfFactory.Create(5, 10, 10)));

var tip = new LinguisticVar<Tip>(
    LinguisticVar<FoodQuality>.DiscreteUniverse(0, 25, 0.5), "tip")
    .AddTerm(Tip.Low, new FuzzySet(mfFactory.Create(0, 0, 13)))
    .AddTerm(Tip.Medium, new FuzzySet(mfFactory.Create(0, 13, 25)))
    .AddTerm(Tip.High, new FuzzySet(mfFactory.Create(13, 25, 25)));
```

Con estás definiciones podemos crear entonces una base de conocimiento con tres reglas para evaluar la propina que se dará

```
kb = new KnowledgeBase()
         .AddVariable(food)
         .AddVariable(service)
         .AddVariable(tip);

kb.AddRule()
  .If(food.Is(FoodQuality.Bad) | service.Is(ServiceQuality.Poor))
  .Then(tip.Is(Tip.Low));
kb.AddRule()
  .If(service.Is(ServiceQuality.Acceptable))
  .Then(tip.Is(Tip.Medium));
kb.AddRule()
  .If(food.Is(FoodQuality.Great) | service.Is(ServiceQuality.Amazing))
  .Then(tip.Is(Tip.High));
```

Y finalmente crearemos nuestro Sistema de Inferencia utilizando Mamdani como método de agregación y para la Desdifusificación el método del centroide

```
var fis = new FuzzyInferenceSystem(
            kb, new CentroidDefuzzFactory().Create(), new MamdaniMethod());

var inputs =
        new InputValues()
            .AddValue("service", 9.8)
            .AddValue("food", 6.5);

var tip = fis.Compute(inputs);
```

Resultados

Para el ejemplo mostrado arriba, con una evaluación de 9.8 para el servicio y de 6.5 para la calidad de la comida, se obtuvo un resultado del 19.67%%.

A modo de validación se probó con el método de agregación de Larsen y los otros desdifusificadores y se obtuvieron resultados parecidos para los mismos valores de las variables de entrada.

Método de Agregación	Centroide	Bisección	Máximos	
			LOM	SOM
Mamdani	19.67%	24.5%	24.5%	24.5%
Larsen	20.27%	24.5%	24.5%	24.5%