Membres du groupe : 2018-2019

**OBAME OBAME Melchior** ENSISA 2A IR

**Saad BENDAOUD**

**Prachi SHARMA**

**COMPTE RENDU TP LOGIQUE FLOUE**

**Objectif** : Implémenter en C++ un Framework générique pour des systèmes d’inférence flous type **Mamdani** et **Sugeno**. Ce Framework devant être extensible et avoir des opérateurs polymorphes. On l’appliquera dans ce TP au *calcul de pourboire.*

**Introduction :**

Le projet n’est défini qu’à partir de son model et de son Métamodèle UML.

**Hypothèses de départ :**

- Template Method : les parties déléguées aux sous-classes.

- structure d’expressions hiérarchique *Composit & Interpret*.

- Utilisation des procurations et du comportement polymorphe(méta-modélisation).

**Démarche et Progression :**

Après la création des namespaces *Core* et *fuzzy*, nous avons progressé tel quel :

Dans *Core* :

**-**  Création des interfaces d’expréssion Expression, *UnaryExpression* , *BinaryExpression* ( et complété par la suite par *NaryExpression*) avec leurs méthodes *evaluate*() définies en virtuelle pur.

**-** Création du model avec les classes *UnaryExpressionModel*, *ValueModel*, *BinaryExpressionModel* implémentant en public les Interfaces d’Expression précédemment définies.

Dans *fuzzy* :

**-** Ajout des classes *AND*, *Or*, *Then*, *Agg*, *Not*, Is ainsi que leurs sous-classes *AndMin*, *AndMult*, *OrMax*, …(complétées par *MamdaniDefuzz* , *CogDefuz*, *Sugenodefuzz*, *SugenoConclusion*)

**-** Phase des test des sous-classes *AndMin*, *AndMult*, *OrMax*, *NotMinus* …

**-** Ajout des Expressions shadow *UnaryShadowExpression*, *BinaryShadowExpression*.

**-** création des Factory *ExpressionFactory* et *fuzzyFactory*.

**-** Implémentation de l’exemple simplifié et test.

**Mise en pratique :**

-Les évaluations des expressions binaires *AndMin*, *AndMax*, *OrMax* … Sont faites en utilisant les opérations auxquelles elles font référence(ex : AndMin ::evaluate(l,r)🡪 min(l->evaluate(), r->evaluate() …)

-*IsTriangle* renvoie l’image de l’évaluation de l’expression donnée par une fonction triangle, selon la pente sur laquelle elle appartient … même logique pour *IsSigmoïd*, *IsGaussian*, *IsTrapeze*…

Les classe factories, quant à elles, nous permettent de créer des expressions, et des systèmes *Mamdani* ou *Sugeno* plus simplement en utilisant des *ShadowExpression*. Ces *shadow* Expression font le lien entre les modèles d’expression et les expressions.

**Défuzzification :**

Une fois que l’on a créé notre modèle à l’aide de la *FuzzyFactory*, on crée un *defuzz* utilisant ce pour pouvoir recréer la courbe pour *CogDefuzz*, en évaluant l’expression finale pour chaque valeur de cette variable. A partir de cette courbe on peut alors calculer son *centre de gravité* et ainsi avoir le résultat.

Dans le cas de *Sugeno* cette défuzzification et obtenu par la formule barycentrique donnée.

Problèmes rencontrés :

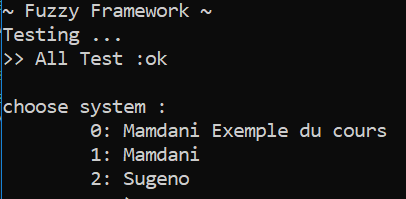
* Défuzzification type *Sugeno* et lien entre *SugenoConclusion* et *SugenoDefuzz*
* Génération de la forme Shape pour faire la défuzzification type *Mamdani* et gérer les effets de bord.
* Rôle de la *premiseValue* dans *SugenoThen*
* Rôle de memory dans *ExpressionFactory*

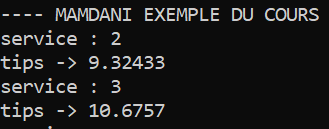
**Conclusion :**

Notre fonction main() comprend  : - tous les test d’évaluation sur les expressions binaires et unaires.

-Lance notre Framework nous demandant de choisir en un système Mamdani ou Sugeno

***Acceuil***  ***System Mamdani (sur l’exemple de cours) :***





Exemple complété avec Service et Food :

***System Mamdani :*** ***System Sugeno***

