Rapport projet LO21

I) Introduction

Ce projet a été réalisé dans le cadre de l'UV LO21 enseignée à l'Université de Technologie de Belfort Montbéliard, au semestre d'automne 2020.
L'objectif du projet est d'implémenter un système expert en langage C. Le système expert est constitué d'une base de connaissance, une base de fait et d'un moteur d'inférence. La base de connaissance est définie par ses propositions et ses règles qui mettent en relation les propositions.

II) Implémentation du projet

Pour réaliser ce projet, j'ai décidé d'implémenter les structures C suivantes :

Une liste chainée d'entiers sans répétitions de valeur

Nom Structure : ListEntier		**Nom dans le code :** i_list
Nom valeurs	**Type**	**Description**
Valeur	Entier	La valeur de l'élément de la liste
Suivant	Pointeur sur ListEntier	Désigne l'élément suivant de la liste

Une liste chainée de propositions

Nom Structure : Proposition		**Nom dans le code :** s_list
Nom valeurs	**Type**	**Description**
Id	Entier	Identifiant de la proposition
Description	String	Description de la proposition
Suivant	Pointeur sur Proposition	Désigne l'élément suivant de la liste

Une liste chainée de règles

Nom Structure : Règle		**Nom dans le code :** r_list
Nom valeurs	**Type**	**Description**
Id	Entier	Identifiant de la règle
Prémisse	ListEntier	Liste d'id des propositions de la prémisse
Conclusion	Entier	Identifiant de la proposition conclusion
Suivant	Pointeur sur Règle	Désigne l'élément suivant de la liste

Structure de la Base de connaissance

Nom Structure : BDC		**Nom dans le code :** Database
Nom valeurs	**Type**	**Description**
Nom	String	Nom de la BDC
Propositions	ListProposition	Toutes les propositions de la BDC
Règles	ListRègle	Toutes les règles de la BDC

III) Algorithmes

III.1) Algorithmes des règles

-R.Premisse est de type ListEntier, ce type de liste est considéré vide lorsqu'il est indéfini

Algorithme : Créer une règle vide

Nom Fonction : RègleVide
Données
Résultat R : Règle
Lexique :

Début
    R ← Allouer Mémoire pour Règle
    R.id ← Indéfini
    R.Prémisse ← Indéfini
    R.Conclusion ← Indéfini
    R.Suivant ← Indéfini
Fin

-P est la liste de toutes les propositions de la BDC
-EstDansListProp(Proposition, Entier) est une fonction qui retourne vrai si il existe une proposition avec l'identifiant donné dans la liste de propositions donnée
-IsérerQueue(ListEntier, Entier) est une fonction qui insère l'entier donné en queue de la liste d'entier donnée, seulement si il n'est pas déjà dans la liste

Algorithme : Ajouter une proposition à la prémisse d'une règle, l'ajout se fait en queue

Nom Fonction : InsérerPremisse
Données R : Règle, P : Proposition, IdProp : Entier
Résultat R : Règle
Lexique :

Début
    Si EstDansListProp(P, IdProp) Alors
        R.Prémisse ← InsérerQueue(R.Prémisse, IdProp)
    Sinon
        Annoncer Erreure
    Fin si
Fin

-P est la liste de toutes les propositions de la BDC
-EstDansListProp(Proposition, Entier) est une fonction qui retourne vrai si il existe une proposition avec l'identifiant donné dans la liste de propositions donnée

Algorithme : Définir la conclusion d'une règle

Nom Fonction : DéfinirConclusion
Données R : Règle, P : ListProposition, IdProp : Entier
Résultat R : Règle
Lexique :

Début
    Si EstDansListProp(P, IdProp) Alors
        R.Conclusion ← IdProp
    Sinon
        Annoncer Erreur
    Fin si
Fin

-P est la prémisse de la règle à tester
-ValeurTête(ListEntier) est une fonction qui retourne la valeur du premier élément de la liste d'entiers donnée
-Reste(ListEntier) est une fonction qui retourne la liste d'entiers donnée privée de son élément de tête

Algorithme : Teste si une proposition appartient à la prémisse d'une règle, de manière récursive

Nom Fonction : EstDansListEntier
Données P : ListEntier, IdProp : Entier
Résultat B : Booléen
Lexique :

Début
    Si P = Indéfinit Alors
        B ← Faux
    Sinon Si ValeurTête(P) = IdProp Alors
        B ← Vrai
    Sinon
        B ← EstDansListEntier(Reste(P), IdProp)
    Fin Si
Fin

- RemoveFromListEntier(ListEntier, Entier) est un fonction qui supprime un entier d'une liste d'entiers

Algorithme : Supprimer une proposition de la prémisse d'une règle

Nom Fonction : SuppDePremisse
Données R : Règle, IdProp : Entier
Résultat R : Règle
Lexique :

Début
| R.Prémisse = RemoveFromListEntier(R.Prémisse, IdProp)
Fin

-R.Premisse est de type ListEntier, ce type de liste est considéré vide lorsqu'il est indéfini

Algorithme : Tester si la prémisse d'une règle est vide

Nom Fonction : PremisseEstVide
Données R : Règle
Résultat B : Booléen
Lexique :

Début
| B ← R.Premisse = Indéfinit
Fin

-P est la liste de toutes les propositions de la BDC

Algorithme : Accéder à la proposition se trouvant en tête d'une prémisse

Nom Fonction :
Données R : Règle, P : ListProposition
Résultat out : Proposition
Lexique :
Id : Entier

Début
| Id ← HeadValue(R.Premisse)
| Tant Que P ≠ Indéfinit Et Identifiant(P) ≠ Id Faire
| | P ← Rest(P)
| Fin Tant Que
| Si P = Indéfinit Alors
| | Annoncer Erreur (id d'une proposition qui n'existe pas dans la prémisse)
| Sinon
| | Out ← P
| Fin Si
Fin

-P est la liste de toutes les propositions de la BDC

Algorithme : Accéder à la conclusion d'une règle

Nom Fonction :
Données R : Règle, P : ListProposition
Résultat P : Proposition
Lexique :
Id : Entier

Début
    Id ← R.Conclusion
    Tant Que P ≠ Indéfinit Et Identifiant(P) ≠ Id Faire
        P ← Rest(P)
    Fin Tant Que
    Si P = Indéfinit Alors
        Annoncer Erreur (id d'une proposition qui n'existe pas dans la prémisse)
    Sinon
        Out ← P
    Fin Si
Fin

III.2) Algorithmes de la base de connaissance

Algorithme : Créer une base vide

Nom Fonction : BaseVide
Données Nom : String
Résultat B : BDC

Début
B ← Allouer mémoire pour BDC
B.Propositions ← Indéfini
B.Règles ← Indéfini
B.Nom ← Nom
Fin

Algorithme : Ajouter une règle à une base

Nom Fonction :
Données B : BDC, R : Règle
Résultat B : BDC

Début
R.Suivant ← B.Règles
B.Règles ← R
Fin

Algorithme : Accéder à la règle se trouvant en tête de base

Nom Fonction :
Données B : BDC
Résultat R : Règle

Début
R ← B.Règles
Fin

III.3) Algorithme et programme C du moteur d'inférence

-Contient(ListEntier, Entier) est une fonction qui retourne vrai si l'entier donné est présent dans la liste d'entier donnée
-Test_règle(Règle, ListeEntier) est la fonction qui retourne vrai si tous les éléments de la prémisse de la règle donnée sont vrai à partir de la ListeEntier donnée qui représente les propositions qui sont vraies
-Insérer_tete(ListEntier, Entier) est la fonction qui insère en position de tête l'entier donné dans la liste d'entier donnée
-Reste(ListRègle) est une fonction qui retourne la liste de règle donnée privée de son élément de tête

Algorithme : Moteur d'inférence

```
Nom Fonction :
Données  B : BDC, faits : ListEntier
Résultat   sortie : ListEntier
Lexique :
nb_vrai : Entier
règle_vrai : ListEntier
LR : ListRègle

Début
    Nb_vrai ← 0
    Sortie ← Indéfini
    Tant que Nb_vrai > 0
        Nb_vrai ← 0
        LR ← B.Règles
        Tant Que LR ≠ Indéfini
            Si non Contient(règle_vrai, Id_tete(LR)) Alors
                Si (Test_règle(Tete(LR), faits) Alors
                    Insérer_tete(règles_vrai, Conclusion_tete(LR))
                    Insérer_tete(faits, Conclusion_tete(LR))
                    Insérer_tete(sortie, Conclusion_tete(LR))
                    nb_vrai ← nb_vrai + 1
                Fin Si
            Fin Si
            LR ← Reste(LR)
    Fin Tant Que
Fin Tant Que
Fin
```

Code : r_test

```
//Function to test if a rule has all true statments in it's premise
//Parameters :
//    -r, a pointer to the studied rule.
//    -facts, a list of integer that represent the ids of the statments that are tknown true.
//Returns :
//    -out, a booleen that is true if all the statments in the studied rule's premise are true.

bool r_test(Rule * r, i_list facts)
{
  bool out = true;
  int_list_elem * current_int = r->premise;
  while (current_int != NULL && i_contains(facts, current_int->value)){
    current_int = current_int->next;
  }

  if (current_int == NULL){
    return true;
  }
  else{
    return false;
  }
}
```

Code : Moteur d'inférence

```
//Inference engine function:
//Parameters:
//    -d, the database to be tested.
//    -facts, a list of integers that represent the ids of the statements that are true before the
//            engine runs.
//Returns:
//    -new_facts, a list of integers that represent the ids of the statements that were false before
//            the engine runs and that are true after.

i_list d_inference_engine(Database d, i_list facts)
{
  Rule * current_rule;        //pointer to the currently studied rule
  i_list true_rules = NULL;  //Rules that are found to have all true statements
  i_list new_facts = NULL;  //return value of the function
  int nb_true;                //number of new rule found each loop

  while (nb_true > 0){
    nb_true = 0;
    current_rule = d.rules;

    while (current_rule != NULL){
      if (!i_contains(true_rules, current_rule->id)){
        if (r_test(current_rule, facts)){
          true_rules = i_insert(true_rules, current_rule->id);
          facts = i_insert(facts, current_rule->conclusion);
          new_facts = i_insert(new_facts, current_rule->conclusion);
          nb_true += 1;
        }
      }
      current_rule = current_rule->next;
    }

  }

  return new_facts;
}
```

IV) Tester le programme

Vous pouvez récupérer le code source du projet sur ma page github personnelle.
Le projet peut être compilé sous Windows et Linux depuis un terminal avec la commande suivante :

```
gcc main.c src/*.c -o out.exe -I include -I lib
```

Et lancer le programme avec les commandes :

Windows	Linux
out.exe	./out.exe

Inclut dans le projet existe une base de connaissance déjà remplie qui se nomme Recette_de_cuisine, il est possible de la charger dans le programme depuis son premier menu.
Elle permet à l'utilisateur de donner au moteur d'inférence les ingrédients qui sont en sa possession et le résultat contiendra toutes les recettes réalisables avec ces ingrédients.

V) Conclusion

Voici quelques pistes d'amélioration pour ce projet :

- Interface utilisateur graphique
- Pouvoir sélectionner une proposition par son identifiant ou sa description (longueur de Levenshtein)
- Possibilité de sauvegarder une liste de fait
- Possibilité d'éditer une liste de fait sauvegardée
- Plusieurs langages disponibles