LANGAGES DE PROGRAMMATION
IFT 3000 NRC 51158
ÉTÉ 2014

Travail pratique 2 (en équipe de 2)

À remettre, par Intranet (pixel) avant 17h00 le mercredi 16 juillet 2014

1 Énoncé

Nous vous demandons dans ce travail pratique d'implanter **le système expert** du TP1 en utilisant le paradigme orienté objet. En effet, il s'agit de reprendre le travail pratique numéro 1, mais en ajoutant d'autres fonctionnalités.

2 Travail à faire

Le TP consiste en l'implantation de la structure (*module*) Tp2e14 qui a la signature (*module type*) TP2E14 (voir fichier : "TP2-SIG-E2014.mli"). **Il est à noter qu'il ne faut pas modifier la signature TP2E14.** Dans cette signature, il existe trois classes (regle, sexpert, et sexpert_taxonomique), mais votre travail consiste essentiellement à implanter les 14 méthodes de la classe **sexpert_taxonomique** qui hérite de la classe **sexpert** :

1. method regle_existe : regle → bool **(2 points)**
2. method fait_existe : fait → bool **(2 points)**
3. method ajouter_regle : regle → unit **(5 points)**
4. method supprimer_regle : regle → unit **(6 points)**
5. method ajouter_fait : fait → unit **(5 points)**
6. method supprimer_fait : fait → unit **(6 points)**
7. method vider_base_regles : unit **(2 points)**
8. method vider_base_faits : unit **(2 points)**
9. method initialiser_utilisee_regles : unit **(4 points)**
10. method ajouter_liste_regles : (fait list * fait list) list → unit **(6 points)**
11. method ajouter_liste_faits : fait list → unit **(5 points)**
12. method afficher_sexpert : unit **(10 points)**

13. method chainage_avant : int **(15 points)**
14. method chainage_arriere : fait → bool **(20 points)**

Vous connaissez normalement ce que fait la majorité de ces méthodes (consulter l'énoncé du TP1). Cependant, voici des explications supplémentaires surtout pour les nouvelles méthodes à implanter :

- **vider_base_regles : unit** est une méthode qui permet de réinitialiser la base de règles du système expert en une liste vide.
- **vider_base_faits : unit** est une méthode qui permet de réinitialiser la base de faits du système expert en une liste vide.
- **intialiser_utilisee_regles : unit** est une méthode qui permet de réinitialiser le booléen de chacune des règles du système expert à false.
- **afficher_sexpert : unit** est la même méthode du TP1. Cependant, elle permet d'afficher également le booléen de chacune des règles. Cet affichage doit se faire de la même façon que dans les résultats attendus du fichier je.ml.
- **chainage_avant : int** fait le même travail que la méthode saturer_sexpert du TP1. Par contre, elle doit retourner un entier représentant le nombre de faits ajoutés au système expert après sa saturation.
- **chainage_arriere : fait → bool** est une méthode qui permet de vérifier la véracité d'un fait. Pour mieux comprendre cette méthode, rappelons la liste des règles utilisées dans le TP1 :

- R1. Si Animal « a des poils », alors Animal « est un mammifère »
- R2. Si Animal « nourrit ses petits au lait », alors Animal « est un mammifère ».
- R3. Si Animal « a des plumes », alors Animal « est un oiseau »
- R4. Si Animal « vole » et « pond des œufs », alors Animal « est un oiseau »
- R5. Si Animal « est un mammifère » et « mange de la viande », alors Animal « est un carnivore »
- R6. Si Animal « est un mammifère » et « a les yeux dirigés en avant » et « a des dents pointues » et « a des griffes », alors Animal « est un carnivore »
- R7. Si Animal « est un mammifère » et « a des ongles », alors Animal « est un ongulé »
- R8. Si Animal « est un carnivore » et « est brun » et « a des taches noires », alors Animal « est un guépard »
- R9. Si Animal « est un carnivore » et « est brun » et « a des raies noires », alors Animal « est un tigre »
- R10. Si Animal « est un ongulé » et « a un long cou» et « est brun » et « a des taches noires », alors Animal « est une girafe »
- R11. Si Animal « est un ongulé » et « est blanc » et « a des raies noires », alors Animal « est un zèbre »
- R12. Si Animal « est un oiseau » et « ne vole pas », alors Animal « est un oiseau non volant »
- R13. Si Animal « est un oiseau » et « nage », alors Animal « est un oiseau non volant »
- R14. Si Animal « est un oiseau non volant » et « a des pattes longues » et « a un long cou » et « est noir et blanc », alors Animal « est une autruche »

- R15. Si Animal « est un oiseau non volant » et « a des pattes palmées » et « est noir et blanc », alors Animal « est un pingouin »
- R16. Si Animal « est un oiseau » et « vole remarquablement », alors Animal « est un albatros »

Voici maintenant un exemple d'une base de faits (F2) pouvant appartenir à notre système expert :

- F2-1. Prince « a des poils »
- F2-2. Prince « a des dents pointues »
- F2-3. Prince « a des griffes »
- F2-4. Prince « a les yeux dirigés en avant »
- F2-5. Prince « est brun »
- F2-6. Prince « a des taches noires »

La méthode de « chaînage arrière » permet d'implanter un moteur d'inférence qui permet de vérifier la véracité d'un fait que nous nommerons hypothèse. Si cette dernière n'appartient pas à la base de fait (normalement c'est le cas, sinon inutile de le demander à notre système), nous devons partir des règles ayant cette hypothèse comme conclusion. Nous prenons alors les prémisses et vérifions si chaque élément est un fait vérifiable. Si c'est effectivement le cas, nous avons alors démonté que l'hypothèse recherchée est vraie. Lorsqu'un des éléments est un fait inconnu, nous tentons de démontrer sa véracité par le même principe en le rendant lui-même hypothèse.

Pour éclaircir les idées, prenons un exemple utilisant les mêmes règles et la base de faits F2. Nous avons un animal, que nous nommons Prince qui est couvert de poils, qui a des dents pointues, qui a des griffes, que la direction des yeux est en avant, qui a une couleur brune et qui a des taches noires. Nous désirons vérifier si Prince est un guépard. Posons alors l'hypothèse que Prince est un guépard et tentons de démontrer que c'est un fait. Dans l'ensemble de règles à notre disposition, il n'y a qu'une seule règle ayant pour conclusion « est un guépard ».

H : Prince est un « guépard » ?
{
R8 : **Si** Prince «est un carnivore» **et** «est brun» **et** «a des taches noires», **alors** Prince «est un guépard».

- Prince « est brun » ? Oui dans la base de faits!
- Prince « a des taches noires » ? Oui dans la base de faits!
- Prince « est un carnivore » ? On ne le sait pas, ça devient une hypothèse.

H : Prince «est un carnivore» ? Deux règles (R5 et R6) ont comme conclusion «est un carnivore»:
{
R5 : **Si** Prince « est un mammifère » **et** « mange de la viande », **alors** Prince « est un carnivore ».

- Prince « est un mammifère »? On ne le sait pas, ça devient une hypothèse !
- Prince « mange de la viande »? On ne le sait pas, ça devient une hypothèse!

H : Prince « est un mammifère » ? Deux règles (R1 et R2) ont comme conclusion « est un mammifère » :
{
R1 : **Si** Prince « a des poils », **alors** Prince « est un mammifère ».

- Prince « a des poils »? Oui dans la base de faits.

Nous ajoutons alors à notre base de faits Prince « est un mammifère ».
}R1 a marché, donc pas besoin de R2!

H : Prince «mange de la viande»? Pas dans la base de fait et aucune règle ne comporte une conclusion: **alors**, Animal «mange de la viande». On ne sait pas si l'hypothèse est vraie.
}R5 n'a pas marché;

H : Prince « est un carnivore » ? Vérifions avec R6
{
R6 : **Si** Prince « est un mammifère » **et** « a les yeux dirigés en avant » **et** « a des dents pointues » **et** « a des griffes », **alors** Prince « est un carnivore ».

- Prince « est un mammifère » ? Oui ajouter dans la base de faits par R1.
- Prince « a les yeux dirigés en avant » ? Oui dans la base de faits.
- Prince « a des dents pointues »? Oui dans la base de faits.
- Prince « a des griffes » ? Oui dans la base de faits.

Donc Prince est un « carnivore » et nous l'ajoutons à la base de faits.
}R6 a marché;

Prince est donc un carnivore par la règle R6.
}R8 a marché.

Donc par chaînage arrière, nous avons réussi à démontrer que Prince est un guépard.

Voici alors l'algorithme permettant de faire ce type de validation.

Pour une hypothèse à démontrer :
{
Pour chaque règle dont la conclusion s'assortit à l'hypothèse courante :
{
Essayer de vérifier chacune des prémisses de la règle en essayant de l'assortir aux assertions de la mémoire de travail ou en « chaînant vers l'arrière » au travers d'une autre règle, créant ainsi de nouvelles hypothèses.

Si toutes les prémisses de la règle sont vérifiées, retourner succès et conclure que l'hypothèse est vraie.
}
}

Votre mandat est donc surtout d'implanter ou réimplanter les deux modes de résolutions : déduire de nouveaux faits en appliquant les règles aux faits observés (chaînage avant) et la validation d'un fait précis (chaînage arrière) en utilisant le paradigme orienté objet.

3 Démarche à suivre

Puisque la structure Tp2e14 contient des fonctions qui ne sont pas encore implantées (c'est à vous de le faire), il ne vous sera pas possible de tester cette structure ou de charger le fichier "TP2-E2014.ml", avant de compléter la définition de ces fonctions. La démarche à suivre est la suivante:

- Utiliser un éditeur comme Emacs, pour compléter l'implantation des fonctions.
- Une fois la structure Tp2e14 complétée, il vous sera alors possible de la tester et de charger le fichier «TP2-E2014.ml» pour y tester les fonctions qui y sont définies.
- Le test peut se faire en utilisant le fichier fourni "je.ml". Dans ce même fichier, il y a les résultats attendus. Vous devez donc vérifier que vous obtiendrez exactement les mêmes résultats. De plus, il ne faut pas oublier de faire la gestion des erreurs.
- Il faut donc mettre ces deux fichiers ainsi que "TP2-SIG-E2014.sml" dans un même répertoire et d'exécuter dans Ocaml la commande suivante : #use "je.ml";; (il ne faut pas oublier le # avec la commande use).

4 À remettre

À l'aide de l'intranet (pixel), Il faut seulement remettre un fichier zip contenant le fichier "TP2-E2014.ml" complété. Les fonctions doivent être bien indentées. N'oubliez pas d'indiquer votre nom et matricule dans l'entête de ce fichier. Il est à noter que **10 points** sont donnés pour le respect et la qualité des biens livrables ainsi que pour la structure générale du code (commentaires, indentation, compilation sans warnings, etc.).

Bon travail !