Projet
Un « Qui est-ce ? » à base de règles

Contenu de votre fichier de programme : votre fichier doit contenir en première ligne vos nom, prénom et numéro d'étudiant. Les commentaires doivent au minimum concerner les types des arguments et des résultats de vos fonctions, ainsi que les tests réalisés, avec leurs résultats. Cela compte pour une part importante de l'évaluation.

Procédure de rendu du projet : lors de la seconde séance, mardi 5 mai, vous prendrez un RDV pour présenter votre travail à votre enseignant lors de la séance du mardi 12 mai. Vous devez à la fin de chaque séance envoyer par mail le fichier de votre TP à votre enseignant avec comme sujet du mail « LIF3 projet». Le corps du mail doit comporter vos nom, prénom et numéro d'étudiant, ainsi que votre fichier attaché nommé « ProjetLIF3VotreNom.scm ».

I. Présentation du projet

L'objet de ce projet est de créer un petit « système expert » qui essaie de deviner le nom d'un animal choisi par l'utilisateur parmi un ensemble de noms d'animaux. Ce système raisonne à partir de règles et de questions posées à l'utilisateur.

Le programme que vous allez réaliser devra pouvoir s'appliquer à n'importe quelle base de règles, ce que vous montrerez dans la partie VI.

On considère la base de règles suivantes :

SI mammifere ET raies-noires ET carnivore ET couleur-fauve ALORS tigre
SI donne-lait ALORS mammifere
SI avec-poils ALORS mammifere
SI mammifere ET avec-sabots ALORS ongule
SI mammifere ET ruminant ALORS ongule
SI predateur ALORS mange-viande
SI mange-viande ALORS carnivore
SI mammifere ET carnivore ET taches-noires ET couleur-fauve ALORS guepard
SI ongule ET raies-noires ALORS zebre

Chaque règle a une ou plusieurs prémisses, et une seule conclusion.

On distingue dans cette base de règles :

- des faits terminaux, qui ne figurent jamais dans la partie prémisse des règles (ici les animaux)
- des faits observables, qui ne figurent jamais dans la partie conclusion des règles. Ce sont sur ces faits observables que l'on peut poser une question à l'utilisateur
- des faits intermédiaires, qui figurent à la fois comme prémisse et comme conclusion (comme mammifere).

Supposons que l'utilisateur pense à un zèbre, voici un exemple d'exécution du programme que vous allez réaliser (les réponses de l'utilisateur sont en italique) :

```
Choisissez dans la liste suivante : (tigre guepard zebre)
Est ce que donne-lait? (oui-non) non
Est ce que avec-poils? (oui-non) oui
Est ce que raies-noires? (oui-non) oui
Est ce que predateur? (oui-non) non
Est ce que avec-sabots? (oui-non) oui
zebre
```

II Définition des règles

On définit une *règle* comme une liste d'éléments dont le premier (qui correspond à la conclusion) est appelé *titre* et les suivants (qui correspondent aux prémisses) des *caractéristiques*.

```
R1 = (tigre mammifere raies-noires carnivore couleur-fauve)
R2 = (mammifere donne-lait)
R3 = (mammifere avec-poils)
```

On définit un *ensemble de règles* par une liste de règles. Un ensemble de règles est donc une liste de listes.

• Définir l'ensemble de règles *EDR* composé des règles *R1* à *R9*. *EDR* est une variable globale du programme.

• Définir la liste *EDA* composée de l'ensemble des animaux que peut choisir l'utilisateur.

III Fonctions sur l'ensemble des règles

• Écrire la fonction *EnleverTitre* qui, à partir d'un ensemble de règles, enlève le titre de chacune des règles.

```
(EnleverTitre '((a b c) (d e) (f e d)) -> ((b c) (e) (e d))
```

• Écrire la fonction *ReglesPossibles* qui, à partir d'un élément *T* et d'un ensemble de règles, renvoie l'ensemble des règles dont le titre est *T*.

```
(ReglesPossibles 'mammifere EDR)
     -> ((mammifere donne-lait) (mammifere avec-poils))
```

IV Questions à l'utilisateur

Dans cette partie, on traite le cas où le programme demande à l'utilisateur la valeur (oui ou non) d'une caractéristique. Le système manipule une liste de couples nommée LQR qui est une variable globale. Les couples ont la forme (caractéristique *V*) où *V* vaut `'oui` si la caractéristique est vraie et `'non` si elle est fausse. LQR sert à conserver les réponses de l'utilisateur pour ne pas lui poser deux fois la même question.

• Définir la variable globale LQR (elle est vide au départ).

• Écrire la fonction *Demande* qui prend en paramètre une caractéristique et affiche un message demandant à l'utilisateur si celle-ci est vraie. La fonction renvoie `'oui` ou `'non` selon la réponse de l'utilisateur. De plus, la liste LQR est mise à jour automatiquement par cette fonction.

• Écrire la fonction *RenvoieValeur* qui prend en paramètre une caractéristique et une liste de couples (caractéristique *V*) et renvoie sa valeur *V* si elle existe dans la liste, et `'sais-pas` si elle ne s'y trouve pas.

• Écrire la fonction V*aleurCaract* qui prend une caractéristique observable en paramètre et renvoie sa valeur *V* (prise dans la liste LQR si elle s'y trouve et demandée à l'utilisateur sinon).

V Programmation du système

Le système expert doit deviner le nom de l'animal. Pour cela, il essaie successivement les noms d'animaux possibles et essaie de savoir lequel est le bon. Le principe à appliquer est le suivant : pour un animal donné, le système cherche, dans l'ensemble des règles EDR, la règle le définissant. Par exemple, pour savoir si l'animal est un zèbre, il faut se servir de la dernière règle indiquant qu'un zèbre est un ongulé et qu'il a des raies noires. Ensuite, le système vérifie si toutes les caractéristiques associées à cette règle sont vraies (il doit savoir si l'animal est un ongulé et a des raies noires). Il faut alors travailler récursivement pour vérifier les différentes caractéristiques (et éventuellement leurs sous-caractéristiques).

Pour une caractéristique donnée, 3 cas sont alors possibles :

- Soit la valeur de la caractéristique est connue.
- Soit l'ensemble de règles EDR contient des règles permettant de montrer la caractéristique. C'est le cas pour "ongulé". Le programme cherche ensuite à démontrer les éventuelles sous-caractéristiques.
- Soit la valeur n'est pas connue et il n'y a pas de règle (par exemple "raies-noires"), le programme demande alors à l'utilisateur.

Les trois fonctions suivantes dépendent les unes des autres, il est donc important de lire leur spécification jusqu'au bout.

• Écrire les fonctions booléenne *UnEnsembleVrai*, *CaracteristiquesVraies* et *CaracteristiqueVraie*.

- La fonction *UnEnsembleVrai* prend en paramètre un ensemble de listes de caractéristiques, et renvoie `#t` si au moins l'une des listes ne contient que des caractéristiques vraies, `#f` sinon.
- La fonction *CaracteristiquesVraies* prend en paramètre une liste de caractéristiques, et renvoie `#t` si chaque caractéristique de la liste est vraie, `#f` sinon.
- La fonction *CaracteristiqueVraie* prend en paramètre une caractéristique et vérifie qu'elle est vraie, soit en vérifiant les sous-caractéristiques des règles qui permettent de montrer cette caractéristique, soit en cherchant dans la liste LQR ou en demandant à l'utilisateur.

• Écrire enfin la fonction *Trouve* qui prend une liste de noms d'animaux en paramètre, cherche et renvoie le nom de l'animal auquel pense l'utilisateur, c'est-à-dire le premier dont toutes les caractéristiques sont vraies.

• Écrire la fonction principale *QuiEstCe*, qui n'a pas de paramètre et appelle la fonction Trouve avec la liste EDA.

VI Un autre ensemble de règles

• Définissez un autre ensemble de règles sur un domaine qui vous intéresse, faites des tests et présentez les résultats.

VII Améliorations

N.B. Si vos améliorations vous amènent à modifier une fonction précédemment définie, laissez dans votre fichier une version initiale de la fonction sous forme de commentaire.

• Définir une fonction qui construit la liste EDA des faits terminaux à partir d'une analyse de la base de règles.

• La variable globale LQR permet de conserver la valeur des faits observables, afin de ne pas poser deux fois la même question à l'utilisateur. Cependant, il est probable que votre programme fasse plusieurs fois les mêmes calculs pour déterminer la valeur des faits intermédiaires (par exemple mammifere). Modifiez votre programme afin que les faits intermédiaires qui ont déjà été démontrés par l'application d'une règle soient mémorisés.

• Si vous souhaitez apporter une autre amélioration à votre programme, expliquez soigneusement votre objectif et la manière dont vous l'avez (ou l'auriez) réalisé.