S12 : Prolog

Enseignant : Stéphane LE PEUTREC
Assistant : Jonathan LAUPER

Instructions

- Deadline : jeudi suivant à 11:00

1. Prédicats d'ordre supérieur

- myMap(+Pred,+ LS,? LR) : équivalent à mapList/3. Utilisez le prédicat prédéfini call.

```
Exemple : ?- myMap(plus(3),[2,5,8],L).
          L=[5,8,11].
```

- myPartition(+Pred,+List,?Included,?Excluded) : équivalent au prédicat prédéfini partition/4.

```
Exemple : ?- myPartition(>(7),[4,9,23,2,6,11],L1,L2).
          L1=[4,2,6], L2=[9,23,11]
```

- filter(+Pred,+List,?Filtered) : équivalent au prédicat prédéfini include/3, vrai si Filtered contient les éléments de List qui satisfont le prédicat Pred.

```
Exemple : ?- filter(>(7),[4,9,23,2,6,11],L1).
          L1=[4,2,6]
```

- filterWithFindall(+Pred,+List,?Filtered) : identique à filter. Utilisez findall pour cette implémentation.
- myIntersection(+E1,+E2,?E3) : équivalent à myIntersection de la série S11. Utilisez findall pour cette implémentation.

2. Problème du zèbre

Cinq maisons toutes de couleurs différentes, sont habitées par des personnes toutes de nationalités différentes. Elles possèdent chacune un animal différent, ont chacune une boisson préférée différente et fument des cigarettes différentes. On sait que :

1. Le norvégien habite la première maison,
2. La maison à côté de celle du norvégien est bleue,
3. L'habitant de la troisième maison boit du lait,
4. L'anglais habite une maison rouge,
5. L'habitant de la maison verte boit du café,
6. L'habitant de la maison jaune fume des kool,
7. La maison blanche se trouve juste après la verte,
8. L'espagnol a un chien,
9. L'ukrainien boit du thé,
10. Le japonais fume des craven,
11. Le fumeur de old gold a un escargot,
12. Le fumeur de gitane boit du vin,
13. Un voisin du fumeur de chesterfield a un renard,
14. Un voisin du fumeur de kool a un cheval.

Qui boit de l'eau ? Qui possède un zèbre ?

Indications

Pour résoudre ce problème, on vous suggère d'écrire la relation housesComposition(C,A,B,F,N) où

- C est la liste des 5 couleurs: C=[C1,C2,C3,C4,C5] tel que Ci est la couleur de la ième maison.
- A est la liste des 5 animaux: A=[A1,A2,A3,A4,A5] tel que Ai est l'animal de la ième maison.
- B est la liste des 5 boissons: B=[B1,B2,B3,B4,B5] tel que Bi est la boisson consommée dans la ième maison.
- F est la liste des 5 cigarettes: F=[F1,F2,F3,F4,F5] tel que Fi sont les cigarettes fumées dans la ième maison.
- N est la liste des 5 nationalités: N=[N1,N2,N3,N4,N5] tel que Ni est la nationalité de l'habitant de la ième maison.

Vous pouvez développer et utiliser le prédicat sameIndex(E1,E2,L1,L2) où sameIndex(E1,E2,L1,L2) est vrai si E1 et E2 ont respectivement les mêmes indices dans les listes L1 et L2.

Développez ensuite les prédicats suivants :

- drink(N,D) : vrai si la personne de nationalité N boit la boisson D
- hasAnimal(N,A) : vrai si la personne de nationalité N possède l'animal A
- smoke(N,C) : vrai si la personne de nationalité N fume des cigarettes C
- color(I,C) : vrai si la maison de numéro I est de couleur C

Pour éviter de recalculer la composition des 5 maisons à chaque question sur les prédicats drink, hasAnimal, etc, on propose de calculer une seule fois la composition des maisons et de la mémoriser par 5 clauses de la forme :

house(Number,Color,Nationality,Animal,Drink,Cigarette).

où Color, Nationality, Animal, Drink, Cigarette sont respectivement la couleur, la nationalité de l'habitant, l'animal, la boisson, et les cigarettes de la maison numéro Number.

3. Mini-projet - Partie 2 : Générateur d'analyseurs syntaxiques

Le but de cet exercice est de générer dynamiquement les règles DCG pour une grammaire donnée.

Première étape : proposez sur papier une grammaire algébrique d'axiome de nom 'grammar' qui reconnaît des grammaires respectant les points suivants et implémentez les règles DCG correspondantes.

- la forme d'une règle est :
  - soit : <rule name> -> <rule body> | ... | <rule body>
  - soit : <rule name> -> <rule body>
- le nom d'une règle respecte le pattern suivant : r_xxx où xxx est une suite quelconque de caractères
- le corps d'une règle est de la forme : <rule body part> $^{+}$ où <rule body part> est soit un nom de règle soit un atome ne commençant pas par 'r_'.

Exemples de grammaires reconnues :

- grammaire correspondant à l'expression régulière $(ab)^{+}$
  r_1 -> a b r_1 | a b
- grammaire correspondant à l'expression régulière $(a|b)^{+}$
  r_1 -> r_2 r_1 | r_2

r_2 -> a | b

Exemples d'appel du prédicat grammar pour les deux grammaires précédentes.

- grammar([r_1, '->', a, b, r_1, '|', a,b],[]). retourne yes car la liste contient des règles reconnues par votre grammaire
- grammar([r_1, '->', r_2, r_1, '|', r_2, r_2, '->', a, '|', b],[]). retourne yes

Indications : vous pouvez utiliser les prédicats prédéfinis atom/1 et atom_chars/2. atom(X) est vrai si X est un atome; atom_chars(X,L) est vrai si L est la liste des caractères de l'atome X.
atom_chars(toto,L) est vrai si L=[t,o,t,o].

Deuxième étape : Ajoutez des extra arguments aux règles DCG pour qu'elles génèrent les règles DCG correspondant à la grammaire reconnue.

- Exemple : grammar(RL, [r_1, '->', r_2, r_1, '|', r_2, r_2, '->', a, '|', b],[]). retourne vrai avec RL= [(r_1-->r_2, r_1), (r_1-->r_2), (r_2-->[a]), (r_2-->[b])]

Indication : vous pouvez utiliser le prédicat transformListInTermWithoutFunctor/2 donné en annexe. transformListInTermWithoutFunctor (L,T) est vrai si T est de la forme (e1,e2,...) où e1, e2 etc sont les éléments de la liste L. Exemple : transformListInTermWithoutFunctor([[a],r_2,[b]], L) retourne vrai avec L = ([a], r_2, [b]).

Troisième étape : Développez le prédicat generateRules(Gramar) qui génère les clauses modélisant la grammaire Grammar (exprimée sous forme de liste) et les ajoute à la base de connaissance du moteur d'inférence Prolog.
Exemple : generateRules([r_1, '->', r_2, r_1, '|', r_2, r_2, '->', a, '|', b]). retourne vrai et crée dynamiquement les clauses modélisant la grammaire ayant pour règles r_1 -> r_2 r_1 | r_2 et r_2 -> a | b
Après cet appel, il est possible d'interroger le prédicat r_1. Exemple : r_1([a,a,b],[]). retourne vrai car la séquence a a b est reconnue par la grammaire d'axiome r_1.
Indication : vous pouvez utiliser les prédicats prédéfinis expand-term et assert vus en cours.