TD de *Prolog et programmation par contraintes* n° 3
(Correction)

# Le problème du M1

On considère les 4 *parcours* suivants:

- *Systèmes d'Information* (SI) .
- *Système, Réseaux, Internet* (SRI).
- *Logiciels Critiques* (LC).
- *Master parisien de Recherche en Informatique* (MPRI) .

Un parcours est caractérisé par deux ensembles: les UE (*unités d'enseignement*) obligatoires et les UE optionnelles du parcours.

Une UE est caractérisée par: le code de l'UE, les ects (*european credit transfer system*) de l'UE, le semestre de l'UE.

En page 3, les données du probléme sont présentées sous forme de tableaux.

Un *projet_de_parcours* est un ensemble de UE.

Un projets_de_parcours est:

- *cohérent* si:
  - il contient exactement une UE parmi AN1 et AN2.
  - il contient exactement une UE parmi SE ,PP et TR.
  - la somme des ects des UE du projet, pour chaque sémestre, est au moins égale à 30.
- *en accord avec* un parcours s'il contient toutes les UE obligatoires du parcours, et si la somme des ects des UE du projet qui sont obligatoires ou optionnelles pour le parcours est au moins égale à 51.
- *acceptable* s'il est cohérent et il est en accord avec au moins un parcours.

**Exercice 1** Ecrire un programme qui fournit comme interface les prédicats `coherent/1`, `en_accord/2` et `acceptable/1`, dont la sémantique est spécifiée ci-dessus. Pour cela, il faut en particulier choisir une representation des UE, des parcours, et des projets, et implementer en fonction de la representation chioisie la base de faits décrite par les tableaux de la page 3.

Voici une spécification détaillée d'une implantation possible de ces prédicats:

1. Définir un prédicat `ue/3` pour décrire les unités d'enseignement (les arguments seront respectivement le code, les ects et le sémestre de l'UE).
2. Définir un prédicat `parcours/3` pour décrire les parcours (les arguments seront respectivement le code, la *liste* des UE obligatoires et la *liste* des UE optionnelles du parcours).
   Un *projet de parcours* est décrit par un liste de codes d'UE.
3. Définir un prédicat `anglais/1` tel que `anglais(L)` réussit si et seulement si `L` est un projet de parcours qui contient exactement une UE parmi AN1 et AN2.
4. Définir un prédicat `fin_etudes/1` tel que `fin_etudes(L)` réussit si et seulement si `L` est un projet de parcours qui contient exactement une UE parmi SE,PP, et TR.
5. Définir un prédicat `ects_sem/3` tel que `ects_sem(1,L,N)` réussit si et seulement si `N` est le nombre d'ects du premier sémestre dans le projet de parcors L, et `ects_sem(2,L,N)` réussit si et seulement si `N` est le nombre d'ects du deuxième sémestre dans le projet de parcors L.

6. Définir le prédicat `coherent/1`.
7. Définir un prédicat `inclus/2` tel que `inclus(L1,L2)` réussit ssi tous les éléments e la liste `L1` sont aussi des éléments de la liste `L2`.
8. Définir un prédicat `inter/3` tel que `inter(L1,L2,L3)` réussit ssi la liste `L3` contient exactement les éléments communs aux listes `L1` et `L2`.
9. Définir un prédicat `compte/2` tel que `compte(L,N)` réussit ssi `N` est la somme des ects des UE de la liste `L`.
10. Définir le prédicat `en_accord/2`.
11. Définir le prédicat `acceptable/1`.
12. Appliquer ces prédicats a votre propre projet de parcours.

**Correction :**

```
anglais([an1|L]):-aux(L).
anglais([an2|L]):-aux(L).
anglais([X|L]):-X\==an1,X\==an2,anglais(L).

aux([]).
aux([X|L]):-X\==an1,X\==an2,aux(L).

fin_etudes([se|L]):-aux1(L).
fin_etudes([pp|L]):-aux1(L).
fin_etudes([tr|L]):-aux1(L).
fin_etudes([X|L]):-X\==se,X\==pp,X\==tr,fin_etudes(L).

aux1([]).
aux1([X|L]):-X\==se,X\==pp,X\==tr,aux1(L).

ects_sem(X,[],0).
ects_sem(X,[Y|G],N):-ue(Y,M,X),ects_sem(X,G,K), N is M+K.
ects_sem(X,[Y|G],N):-ue(Y,M,Z), X=\=Z, ects_sem(X,G,N).

coherent(L):- anglais(L), fin_etudes(L),ects_sem(1,L,N1),
N1>=30,ects_sem(2,L,N2),N2>=30.

inclus([],Y).
inclus([A|G],Y):-member(A,Y),inclus(G,Y).

inter([],T,[]).
inter([C|G],T,[C|P]):- member(C,T),inter(G,T,P).
inter([C|G],T,R):- not(member(C,T)),inter(G,T,R).

compte([],0).
compte([P|L],N):-ue(P,M,_),compte(L,Q), N is M+Q.

append([],L,L).
append([X|L],L1,[X|L2]):-append(L,L1,L2).

en-accord(Pr,X):-parcours(X,Ob,Opt), inclus(Ob,Pr),
append(Ob,Opt,T),
inter(Pr,T,L), compte(L,N),N>=51.

acceptable(X):-coh(X),en-accord(X,Y).
```

**Les UE:**

| code | nom | ects | sem. |
|---|---|---|---|
| IA | Intelligence Artificielle | 6 | 1 |
| BD | Base de données avancées | 6 | 1 |
| CA | Circuits et architecture | 6 | 1 |
| SY | Systèmes | 6 | 1 |
| CO | Compilation | 6 | 1 |
| PC | Prolog et prog. par contraintes | 6 | 1 |
| AL | Algorithmique | 6 | 1 |
| CC | Calculabilité et complexité | 6 | 1 |
| TI | Théorie de l'information | 6 | 1 |
| PR | Protocoles reseau | 6 | 2 |
| GL | Génie logiciel | 6 | 2 |
| DI | Droit de l'informatique | 3 | 2 |
| LL | Logiciels Libres | 3 | 2 |
| IG | Interfaces graphiques | 6 | 2 |
| AA | Automates avancés et applications | 6 | 2 |
| IN | Infographie | 6 | 2 |
| TC | Théor. et prat. de la concurrence | 6 | 2 |
| TE | Techniques d'expression | 3 | 2 |
| PA | Preuves assistées par ordinateur | 6 | 2 |
| SM | Sémantique des lang. de prog. | 6 | 2 |
| AV | Algorithmique avancée | 6 | 2 |
| AS | Analyse de perf. et simulation | 6 | 2 |
| SE | Stages en entreprise | 6 | 2 |
| PP | Projets programmation | 6 | 2 |
| TR | Travaux de recherche encadrés | 6 | 2 |
| AN1 | Anglais | 3 | 1 |
| AN2 | Anglais | 3 | 2 |

**Les parcours:**

| nom | UE obligatoires | UE optionnelles |
|---|---|---|
| SI | BD,SY,GL | IA,CA,CO,PC,AL,TE,DI,LL,IG,IN,AS,SE,PR,PP |
| SRI | BD,SY,AL,PR | CA,CO,CC,GL,TE,DI,,LL,IG,IN,TC,SE,PP |
| LC | CO,AL,GL,AA,TC | SY,PC,CC,PA,SM,AV,TE,SE,PR,TR,PP |
| MPRI | CO,AL,CC,TR | PC,AA,TC,TI,IN,PA,SM,AV |