DOSSIER PROLOG

AGBENU Yao Phanuel
AGOSSOU Carmel
BACHAR Kovan

Dans ce rapport, nous allons présenter l'évolution de notre travail.
Dans un premier temps nous effectuerons une description du sujet à traiter, puis sa modélisation sous forme de programme de satisfaction de contraintes.
Dans un second temps, nous décrirons le programme effectué et le résultat obtenu.

Description du problème

Le travail consiste à déterminer l'emploi du temps d'une semaine de cours à l'IMA. Cependant il y a des contraintes à respecter telles que la capacité des salles, le besoin de salle particulier de chaque cours et la non-simultanéité des cours d'un même professeur ou d'un même groupe d'étudiants dans une même salle.
Pour la suite nous modéliserons le problème et le programme.

Modélisation du problème et Programme

- **On construit la liste des débuts et fins des demi-journées (en nombre de quarts d'heure).**

```
make_time_limits([], []).
make_time_limits([N|Ns], [B,E|Ts]) :-
  (  N mod 2 =:= 1
  ->  B is 32+(N//2)*4*24,
      E is 50 + (N//2)*4*24
  ;   B is  54 + (N//2 -1)*4*24,
      E is 72 + (N//2 -1)*4*24
  )
     make_time_limits(Ns, Ts).
```

- On crée le domaine des débuts des cours sous forme d'union des intervalles de temps et la liste des débuts et fins des pauses midi et nuit.

```
make_doms_and_breaks([B,E],B..E,[]).
make_doms_and_breaks([B1,E1,B2,E2|Ts],B1..E1 \/ Ds, [E1,B2|Bs]) :-
   make_doms_and_breaks([B2,E2|Ts],Ds,Bs).
```

- **Décompositions de l'heure en minutes avec les prédicats suivants.**

```
atom_to_hour(A, H, M) :-
  atom_codes(A, L),
  codes_to_hour(L, H, M).

codes_to_hour(L, H, M) :-
  append(CH, [104|CM], L),
  number_codes(H, CH),
  number_codes(M, CM).

atom_to_quart(A, Q):-
  atom_to_hour(A, H, M),
  Q is (H*60+M)//15.

quart_to_hour(Q,H,M) :-
   A = Q mod 96,
   H is A//4,
   M is (A mod 4)*15.
```

- **Décomposition de la durée des cours.**

```
courseLengths(Ds) :-
   findall(Q, (cours(_,_,_,D,_,_),atom_to_quart(D,Q)), N),
   Ds=N.
```

- **Création de la liste des débuts et la liste des durées des pauses midi et nuit.**

```
break_start_length([],[],[]).
break_start_length([B,E|Breaks], [B|Bstarts], [Bl|L]) :-
    Bl is E-B,
    break_start_length(Breaks,Bstarts,L).
```

- **Contrainte : un cours ne peut pas avoir lieu pendant une pause.**

```
not_on_breaks(H,D,BreakStarts,BreakLengths) :-
    serialized([H|BreakStarts],[D|BreakLengths]).
```

- **Renvoie la liste Nums des numéros des cours dont le professeur est P.**

```
teacherCourses(P, Nums) :-
    findall(N, cours(N,P,_,_,_,_), Nums).
```

- **Renvoie la liste Nums des numéros des cours dont le groupe est G.**

```
groupcourses(Gr, Nm):-
    findall(N, cours(N,_,Gr,_,_,_), Nm).
```

- **Prédicats pour récupérer des éléments des indices donnes d'une liste.**

```
getByIndex([],_,[]).
getByIndex([I1|In],List,[X1|Xs]):-
    nth1(I1,List,X1),
    getByIndex(In,List,Xs).
```

- **Créations des pauses de 15 min entre chaque cours d'un professeur ou d'un groupe.**

```
make_breaks([], [], [], []).
make_breaks([H|Hs], [D|Ds], [B|Bs], [L|Ls]):-
    B is H + D,
    L = 1,
    make_breaks(Hs, Ds, Bs , Ls).
```

```
add_break([],[]).
add_break([D1|Dn],[X|DBs]):-
   X is D1+1,
   add_break(Dn,DBs).

disjoint_courses_and_breaks(Nums, AllHs, AllDs):-
   getByIndex(Nums,AllHs,Hs),
   getByIndex(Nums,AllDs,Ds),
   add_break(Ds,Dls),
   serialized(Hs,Dls).
```

- **Renvoie la liste Nums des numéros des salles dont le type est si (salle info).**

```
pcRooms(Nums) :-
   findall(N,salle(N,_,'si',_),Nums).
```

- **Renvoie la liste Nums des numéros des salles dont le type est sc (salle de cours).**

```
boardRooms(Nums) :-
   findall(N,salle(N,_,'sc',_),Nums).
```

- **Renvoie la liste caps de toutes les salles.**

```
roomCapacities(Caps):-
    findall(C,salle(_,C,_,_),Caps).
```

- **Créations de l’union des éléments d’une liste donnée.**

```
listToRange([X],X).
listToRange([X|Xs],X\/Range) :-
   listToRange(Xs,Range).
```

- **La salle d'un cours doit pouvoir contenir tous ses étudiants.**

```
fitNbStudents(I,S,Caps):-
    element(S,Caps,C),
    cours(I,_,G,_,_,_),
    groupe(G,_,Nb),
    C #>= Nb.
```

- **Créations du domaine pour les salles.**

```
def_dom_salles([], [], _, _, _).
def_dom_salles([Num|NumsCours], [S|Ss], BoardDom, PCDom, NbSalles) :-
    cours(Num, _, _, _, Type, _),
    (   Type = 'sc', S in BoardDom
    ;   Type = 'si', S in PCDom
    ;   Type = 'ts', S in 1..NbSalles
    ),
    def_dom_salles(NumsCours, Ss, BoardDom, PCDom, NbSalles).
```

- **Labeling.**

```
  append(Hs,Ss,Sol),
    labeling([],Sol).
```

- **Affichage.**

```
print([],[],[]).
print([N|NumsCours],[H|Hs],[S|Ss]) :-
    quart_to_hour(H,Hh,Hm),
    (
     (Jour = 'Lundi', H<96);
     (Jour = 'Mardi', H>=96, H<192);
     (Jour = 'Mercredi', H>=192, H<288);
     (Jour = 'Jeudi', H>=288, H<384);
     (Jour = 'Vendredi', H>=384)
```

```
),
salle(S,_,_,Salle),
atomics_to_string([Jour,' ',Hh,'h',Hm,' ',Salle],StrDate),
write(StrDate), nl(),

cours(N,NumP,NumG,_,_,NomC),
prof(NumP,NomP),
groupe(NumG,NomG,_),
atomics_to_string([NomG,' ',NomC,' ',NomP],StrCours),
write(StrCours), nl(), nl(),
print(NumsCours,Hs,Ss).
```

Résultat

Ci-joint une capture du résultat de data 1 dans la console.

```
?- emploi_temps().
Lundi 8h0 B313
M1 Statistiques Marion

Lundi 10h15 B313
M1 XML Rivault

Lundi 13h30 C017
M1 PPC Nguyen

Mardi 8h0 B313
M1 Optim Pinson
```