# Logique et Programmation Logique : Travaux Pratiques

S. PIECHOWIAK

*Les TP sont organisés en 3 séances de 3 heures chacune.*

## Consignes.

L'interpréteur PROLOG à utiliser est SWI-PROLOG.

Travail demandé : 2 fichiers doivent être envoyés dans le même message.
Le premier fichier est le compte rendu de TP sous la forme d'un seul document au format PDF. On y expliquera les parties délicates des programmes. On présentera des exemples de résultats.
Le nom du fichier est impérativement de la forme :
**< NOM DE L'ETUDIANT >-<TP-PROLOG-2020-2021>.pdf**
Le 2ème fichier contient tous les sources des programmes documentés. Le nom de ce fichier est impérativement de la forme :
**<NOM DE L'ETUDIANT>-<TP-PROLOG-2020-2021>.pl**
Les 2 documents doivent être envoyés ensembles à l'adresse :
sylvain.piechowiak@uphf.fr avant le 11/06/2021 23h.
**Après cette date, les documents seront automatiquement mis à la corbeille (note = 00/20).**

## Utilisation de PROLOG pour gérer une base de données déductive.

Prolog dispose de structures. Par exemple, on peut représenter une date par un triplet comprenant le jour dans le mois, le mois et l'année :

- `date(8,4,1997).`

A partir de cette structure on, peut facilement obtenir le jour, le mois ou l'année pour une date donnée :

- `jour(date(X,Y,Z),X).`
- `mois(date(X,Y,Z),Y).`
- `année(date(X,Y,Z),Z).`

On souhaite gérer les emplois du temps d'un collège à l'aide d'une base de données en PROLOG. Les séances ont toutes une durée de 2h. Les créneaux sur une journée sont 8h-10h (c1), 10h-12h (c2), 14h-16h (c3) et 16h-18h (c4). Les emplois du temps étant reproduits de semaine en semaine, chaque semaine peut être représentée par une liste de créneaux : c11, c12, c13, c14 (lundi), c21, c22, c23, c24 (mardi), c31, c22, c23, c24 (mercredi), c41, c42, c43, c44 (jeudi), c51, c52, c53, c54 (vendredi), c61, c62, c63 et c64 (samedi).

On définit les entités suivantes :

```
professeur(nom, prenom, discipline, identifiant)
eleve(nom, prenom, niveau, identifiant)
salle(nom, capacite, type , identifiant)
matiere(nom, discipline, type, niveau, identifiant)
groupe(nom, niveau, liste-eleves, identifiant)
seance(matiere, prof, groupe, salle, creneau, identifiant).
```

Les identifiants sont uniques : ce sont des entiers.

En PROLOG, il existe des prédicats qui permettent d'ajouter ou de retirer de manière dynamique des faits/prédicats dans une base : **`assert`** et **`retract`** (il en existe d'autres qui leur ressemblent et qui sont décrits dans la documentation de SWI-PROLOG).
Par exemple **`assert(salle(e215, 30, tp, 10)).`** ajoute le fait **`salle(e215, 30, tp, 10)`** dans la base. Pour le retirer, il suffit d'utiliser **`retract(salle(e215, 30, tp, 10)).`**
Tout prédicat qui donne lieu à des ajouts ou des retraits doit être déclaré comme dynamique en début de programme grâce à l'instruction :
**`:- dynamic(<nom du prédicat>/<arité du prédicat>).`**
Dans notre exemple, il faudra donc ajouter la ligne au début du programme
**`:- dynamic(salle/4).`**

1. Définir les prédicats qui permettent d'ajouter et de retirer un professeur, un élève, une salle, une matière, un groupe ou une séance. Avant d'ajouter, on s'assurera que la donnée n'est pas déjà présente dans la base et que l'identifiant n'est pas déjà utilisé.

2. Définir un prédicat qui ajoute un élève dans un groupe. On supposera qu'un élève ne peut appartenir qu'à un seul groupe.

3. Définir un prédicat qui recherche les séances qui ont un problème de capacité de salle (nombre d'élèves du groupe concerné supérieur à la capacité de la salle).

4. Définir les prédicats qui permettent d'avoir la liste des séances d'un professeur, d'un élève, d'une salle, d'un groupe.

5. Définir un prédicat qui détermine le jour associé à un créneau donné. Par exemple, ce prédicat doit renvoyer mardi pour le créneau c24.

6. Définir un prédicat qui détermine l'horaire d'un créneau donné. Par exemple, ce prédicat doit renvoyer 14h-16h pour le créneau c24

7. Définir un prédicat qui vérifie que 2 séances ne sont pas en conflit. On dira que 2 séances sont en conflit si une ressource (professeur, groupe ou salle) est présente dans les 2 séances et que les 2 séances ont lieu pendant le même créneau.

8. Définir les prédicats qui affichent l'emploi du temps d'un professeur, d'un élève, d'une salle. Pour un professeur, on affichera le nom du professeur, puis la liste des séances sous la forme (jour de la semaine, créneau horaire, matière, nom de la salle, nom du groupe). Pour un élève on affichera le nom de l'élève, puis la liste des séances sous la forme (jour de la semaine, créneau horaire, matière, nom de la salle, nom du professeur). Pour une salle on affichera le nom de la salle, puis la liste des séances sous la forme (jour de la semaine, créneau horaire, matière, nom du groupe, nom du professeur). Les prédicats **`write`** et **`writeln`** peuvent s'avérer utiles.

9. Définir les prédicats qui calculent la charge de travail d'un professeur, d'un élève, d'une salle.

On décide de représenter les classes par :

```
classe(nom, niveau, groupe, liste_professeurs, liste_matieres,
liste_seances, identifiant)
```

Ceci permet au directeur du collège de définir les activités qu'il va devoir planifier pour l'année scolaire. Par exemple, on peut supposer qu'il y a 4 classes de 6ème, 3 classes de 5ème, 3 classes de 4ème et 2 classes de 3ème.

10. Définir les prédicats qui permettent d'ajouter et de retirer une classe. Avant d'ajouter, on s'assurera que la classe n'est pas déjà présente dans la base et que l'identifiant n'est pas déjà utilisé.

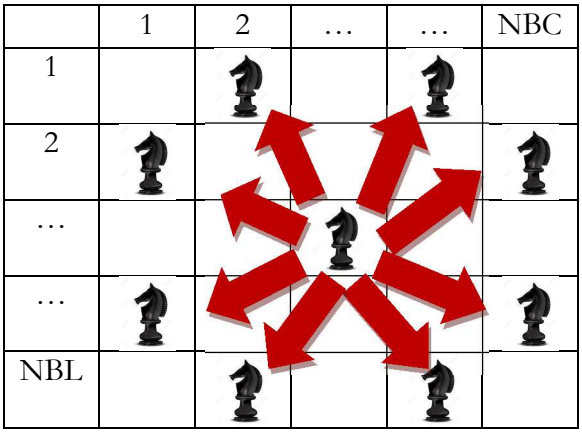
11. En supposant que chaque matière d'une classe donne lieu à 2 séances, définir un prédicat qui vérifie que toutes les séances d'une classe ont bien été planifiées.

12. Définir un prédicat qui permet au directeur du collège de s'assurer que tous ses emplois du temps sont correctement planifiés, c'est-à-dire que toutes les matières de toutes les classes sont bien planifiées et il n'y a aucun conflit. Pour cela on supposera qu'un collège est représenté par la liste de ses professeurs, de ses salles, de ses élèves, de ses groupes, de ses classes

13. Afin de conserver les données dans un fichier, prévoir les prédicats de lecture et de sauvegarde dans un fichier. Pour vous aider, une recherche sur le net peut être utile.

## *De la réflexion …*

On souhaite résoudre le problème du cavalier. Sur un échiquier on place un cavalier et on cherche à savoir s'il peut se déplacer sur toutes les cases sans jamais repasser plusieurs fois par la même. Les mouvements possibles d'un cavalier sont décrits dans le schéma suivant.




On suppose que la taille de l'échiquier est NBL × NBC.

1. Pouvez-vous estimer le nombre de déplacements d'un cavalier en fonction de la taille de l'échiquier ?
2. Proposez une solution en PROLOG pour ce problème.
3. On modifie le problème et cette fois on place 2 cavaliers sur l'échiquier. Chaque cavalier se déplace à tour de rôle et ne peut se déplacer que sur une case qui n'a pas été occupée.
   Proposez une solution en PROLOG pour ce nouveau problème.