Université Paris 13 - Licence d'informatique Année 2009-2010

Projet de *Programmation Logique* n° 1

ITINERAIRE
Calcul d'un trajet de transports publics

L'objectif du projet Itinéraire est de développer un programme en ProLog capable d'aider un usager d'un réseau de transport (par exemple le RATP) à identifier le trajet lui permettant de se rendre d'une station à une autre, en respectant certain conditions (horaire de départ, horaire d'arrivé, minimiser le nombre de correspondances ou la durée du voyage, etc.).

Les questions de l'énoncé sont très détaillées au début, et moins vers la fin pour vous encourager à faire preuve d'originalité. Vous avez le droit d'utiliser n'importe quel prédicat défini dans la bibliothèque standard de `swipl`.

Vous devrez rendre par email à votre responsable de TP d'ici le 22 mai un fichier `.tar.gz` contentant :
- un rapport en format `pdf` détaillant précisément le travail effectué et les choix d'implémentation ;
- un fichier `.pl` contenant le code correspondant aux exercices des sections 2 et 3, en suivant rigoureusement l'énoncé ;
- un fichier `README` expliquant les commandes à taper pour exécuter votre programme ProLog.

Une soutenance de projet sera organisée le jeudi 27 mai. Vous devrez faire une démonstration de votre programme et une présentation de vos choix. Ces soutenances auront lieu sur les machines des salles de TP et sur votre compte. Il est impératif que la démonstration soit prête (programme déjà compilé et sans erreurs) dès le début de la soutenance.

Évaluation. Le travail pourra être fait en binôme, mais la soutenance est individuelle. Le travail sera évalué selon les critères suivants :
- qualité de la conception ;
- fonctionnalités réalisées ;
- qualité du code écrit, lisibilité (votre code doit être bien indenté, et imprimable sans remise en page) ;
- soutenance orale ;
- qualité du rapport.

On préférera un projet simple dont les choix de conception sont bien motivés et qui marche, à un projet trop ambitieux dont on ne comprend pas vraiment la structure et qui ne marche pas trop bien.

1 Représentation de notre réseaux de transport

L'ensemble des lignes sera définie par le prédicat suivant :

```
ligne(Nom, Type, LArret, LTrajetAller, LTrajetRetour)
```

où :

- `Nom` décrit le nome de la ligne, il peut être un numéro `1`, `256`, ou une lettre majuscule `A`, `B`, etc.
- `Type` décrit le type de moyen de transport, soit `metro`, `rer` ou `bus`.
- `LArret` est une liste des pairs

  `[ [A1, T1], [A2, T2], ...,[An,Tn]]`

  ou `Ai` décrit le nom d'un arrêt desservi par la ligne et `Ti` le temps nécessaire à parcourir la distance entre `A(i-1)` et `Ai`. On suppose un temps constant dans les deux directions (de `A(i-1)` vers `Ai` et de `Ai` vers `A(i-1)`) ;
- `LTrajetAller` est le triplet ayant comme premier élément l'horaire du premier départ de la ligne de `A1` vers `An`, comme deuxième élément l'intervalle en minutes entre un départ et l'autre, et comme dernier élément l'horaire du dernier départ de la ligne de `A1`. Les horaires sont représentés sous forme d'une paire des nombres, le premier élément représentant les heures (de 0 jusqu'à 23), le deuxième élément les minutes (de 0 jusqu'à 59). Par exemple le triplet `[[5,15], 5, [1,30]]` signifie que il y aura un départ de `A1` chaque 5 minutes à partir de $05h15$ et jusqu'à $1h30$;
- `LTrajetRetour` est le triplet ayant comme premier élément l'horaire du premier départ de la ligne de `An` vers `A1`, comme deuxième élément l'intervalle en minutes entre un départ et l'autre, et comme dernier élément l'horaire du dernier départ de la ligne de `An`.

Par exemple on peut imaginer de définir la ligne 11 du métro parisienne comme suit :

```
ligne(11, metro,
               [
                [mairie_lilas, 0],
                [porte_lilas, 3],
                [telegraphe,1],
                [place_fetes,1],
                [jourdain, 1],
                [pyrenees, 1],
                [belleville, 2],
                [goncourt, 2],
                [republique, 3],
                [arts_metiers, 2],
                [rambuteau, 1],
                [hotel_de_ville, 1],
                [chatelet, 1]
                ], [[5,15],5,[1,30]], [[5,0],5,[2,0]]
 )
```

2 Recherche des itinéraires

Exercice 1 Let but de cet exercice est de développer les outils nécessaires à manipuler les horaires. On vous suggère de représenter les horaires sous forme d'une paire des nombres `[Heures, Minutes]`, le premier élément représentant les heures (de 0 jusqu'à 23), le deuxième élément les minutes (de 0 jusqu'à 59). Écrire les prédicats

`addh(X,M,R)`, qui est vrai quand `R` est l'horaire obtenu en sommant les minutes `M` à l'horaire `X`, par exemple :

$$\texttt{addh}([13, 34], 30, [14, 4]) \text{ est vrai}$$
$$\texttt{addh}([10, 14], 25, [14, 4]) \text{ est faux}$$

`affiche(H)`, qui affiche sur l'écran l'horaire dans un format lisible par l'utilisateur, comme par exemple l'évaluation du `affiche([5,37])` affichera sur l'écran $05h37$.

Exercice 2 Etant donné un ensemble de lignes, on étudie le problème de savoir si un ligne passe par deux arrêts, éventuellement en respectant un choix de l'horaire de départs ou d'arrivée. Écrire les prédicats suivants :

`lig(Arret1,Arret2,Ligne)`, qui est vrai quand `Ligne` passe de l'`Arret1` à l'`Arret2` ;

`ligtot(Arret1,Arret2,Ligne,Horaire)`, qui est vrai quand `Ligne` part le plus tôt possible après `Horaire` parmi les lignes qui vont de l'`Arret1` à l'`Arret2` ;

`ligtard(Arret1,Arret2,Ligne,Horaire)`, qui est vrai quand `Ligne` arrive le plus tard possible avant `Horaire` parmi les lignes qui vont de l'`Arret1` à l'`Arret2`

Exercice 3 Maintenant on considère le problème de savoir s'il y a un itinéraire entre deux arrêts (éventuellement avec des échanges de moyen de transport). Les itinéraires seront représentés par des listes qui doivent contenir comme informations l'arrêt et l'horaire de départ et d'arrivée, et tous les arrêts et horaires des échanges. Etant donné un ensemble de lignes, écrire les prédicats suivants :

`itinTot(Arret1,Arret2,horaire,Parcours)`, qui est vrai quand `Parcours` décrit un itinéraire de `Arret1` à `Arret2` qui part le plus tôt possible après `Horaire` ;

`itinTard(Arret1,Arret2,horaire,Parcours)`, qui est vrai quand `Parcours` décrit un itinéraire de `Arret1` à `Arret2` qui arrive le plus tard possible avant `Horaire` ;

3 Interface avec l'utilisateur

Exercice 4 Re-écrire les prédicats des exercices précédents en ajoutant des options comme le choix du réseau (ferré ou bus), la préférence par rapport à la longueur du trajet, ou le nombre de correspondances.

Exercice 5 Écrire une interface utilisateur dont le fonctionnement est le suivant :

- le programme affiche les stations desservies par les transports publics ;
- l'utilisateur choisit une station de départ et une station d'arrivée et les options éventuelles (choix du réseau, préférence par rapport à la longueur du trajet, au nombre de correspondances...) ;
- le programme affiche le ou les parcours possibles.