TP numéro 5

Intelligence artificielle, ENSIIE

D'après Michele Pagani

Semestre 4, 2023–24

L'objectif de ce TP est de développer un programme en ProLog capable d'aider un usager d'un réseau de transport (par exemple la 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'arrivée, etc.)

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 nom de la ligne, il peut être un numéro 1, 256, ou une lettre minuscule a, b, etc.
- `Type` décrit le type de moyen de transport, soit `metro`, `rer` ou `bus`.
- `LArret` est une liste des paires `[ [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]]
)
```

On trouvera dans `metro.pl` une petite base pour tester.

2 Itinéraires sans horaires

Le but de cette partie est, étant données une station de départ d et une station d'arrivée a, est de construire un itinéraire de la forme $[d, l_1, s_1, \ldots, l_n, a]$ où les g_i sont des stations de correspondance et l_i est la ligne qui mène de s_{i-1} à s_i. Par exemple on pourra avoir l'itinéraire `[saint_fargeau, bis_3, porte_lilas, 11, place_fetes, bis_7, jaures]` qui prend la ligne 3 bis à Saint-Fargeau pour aller à Porte des Lilas, puis la 11 pour aller à Place des fêtes, puis la 7 bis pour aller à Jaures.

1. Écrire un prédicat `trajet/3` tel que `trajet(L, D, A)` est vrai si on peut aller de `D` à `A` sur la ligne `L` (quel que soit le sens).
2. Écrire un prédicat `a_une_forme_d_itineraire/1` qui vérifie que son argument a une forme d'itinéraire, c'est-à-dire que c'est une liste contenant un nombre impair supérieur ou égal à 3 d'éléments. On utilisera une définition récursive, on ne calculera pas la taille.
3. Écrire un prédicat `itineraire/3` tel que si `itineraire(D, A, I)`, alors `I` est un itinéraire qui mène de `D` à `A`.

   Pour faire une recherche en largeur d'abord plutôt qu'une recherche en profondeur d'abord, on pourra utiliser le prédicat `a_une_forme_d_itineraire/1` dans le cas inductif, pour rechercher d'abord les itinéraires de taille 3, puis 5, puis 7, etc.

3 Gestion des heures

Le but de cette partie est de développer les outils nécessaires à manipuler les horaires. On représentera 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).

4. Définir le prédicat `addh/3` tel que `addh(X,M,R)`, qui est vrai quand `R` est l'horaire obtenu en sommant les minutes `M` à l'horaire `X`, par exemple :

   `addh([13, 34], 30, [14, 4])` est vrai

   `addh([10, 14], 25, [14, 4])` est faux

   `addh([23, 59], 1, [0, 0])` est vrai

   `addh([23, 59], 1, [24, 0])` est faux

5. Définir le prédicat `est_apres/2` tel que `est_apres(H1, H2)` est vrai si `H1` est plus tard dans la journée que `H2`.

4 Itinéraires avec horaires

On veut maintenant des itinéraires qui contiennent des informations horaires. On gardera la même forme que ci-dessus, mais en plus de la ligne on indiquera l'heure de départ sur cette ligne et l'heure d'arrivée. On aura par exemple :
`[goncourt, [[12, 3], 11, [12, 5]], belleville, [[12, 5], 2, [12, 7]], jaures]`

6. Définir un prédicat `revligne/2` tel que `revligne(L, R)` est vrai si `L` est une liste de paires arrêt, temps comme définie dans les lignes, et `R` est la liste obtenue en considérant les stations dans le sens inverse. Attention les temps doivent être décalés !

   Par exemple on aura

   ```
   revligne([[a, 0], [b, 2], [c, 3], [d, 1]],
            [[d, 0], [c, 1], [b, 3], [a, 2]])
   ```

7. Définir un prédicat `passages/4` tel que `passages(D, LS, PA, PD)` est vrai si `LS` est une liste de station avec temps de trajet comme définie dans les lignes, `PA` est un triplet heure de départ, fréquence, heure d'arrivée pour la station de départ d'une ligne, et `PD` est ce même triplet, mais pour la station `D`.

   Avec les valeurs de l'exemple pour la ligne 11, on aura donc
   `passages(goncourt, ..., [[5, 15], 5, [1, 30]], [[5, 26], 5, [1, 41]])`

8. Définir un prédicat `depart/2` tel que `depart(P, H)` est vrai si `P` est un triplet heure de départ, fréquence, heure d'arrivée pour une certaine station, et `H` est un horaire de passage possible à cette station.

   On aura donc

   `depart([[5, 15], 5, [1, 30]], [5, 15])` est vrai

   `depart([[5, 15], 5, [1, 30]], [12, 25])` est vrai

   `depart([[5, 15], 5, [1, 30]], [0, 15])` est vrai

   `depart([[5, 15], 5, [1, 30]], [1, 30])` est vrai

   `depart([[5, 15], 5, [1, 30]], [5, 16])` est faux

   `depart([[5, 15], 5, [1, 30]], [4, 30])` est faux

9. Définir un prédicat `duree/4` tel que `duree(D, A, LS, M)` est vrai si `LS` une liste de station avec temps de trajet comme définie dans les lignes, et `M` est le nombre de minutes pour aller de `D` à `A`, dans ce sens. (Autrement dit le prédicat sera faux si `A` est avant `D` dans la liste, ou si au moins un des deux n'est pas présent.)
10. Définir un prédicat `prochain_depart/6` tel que `prochain_depart(L, D, A, H, HD, HA)` est vrai si le prochain départ pour aller de `D` à `A` sur la ligne `L` en partant après l'horaire `H` part de `D` à `HD` et arrive à `A` à `HA`.
11. Définir un prédicat `partir_apres/4` tel que `partir_apres(D, A, H, I)` est vrai si `I` est un itinéraire avec horaire qui permet d'aller de `D` à `A` en partant après `H`.
12. Définir un prédicat `arriver_avant/4` tel que `arriver_avant(D, A, H, I)` est vrai si `I` est un itinéraire avec horaire qui permet d'aller de `D` à `A` en arrivant avant `H`.