djait
ikram

projet de prolog

Sommaire

Introduction

Le Planificateur de Transport est un programme basé sur Prolog conçu pour aider les utilisateurs à naviguer dans le réseau complexe des systèmes de transport entre différentes stations. Dans les paysages urbains animés d'aujourd'hui, un outil de planification de transport efficace et convivial devient de plus en plus essentiel pour les navetteurs. Ce programme Prolog vise à offrir une solution intuitive et personnalisable aux utilisateurs pour planifier leurs trajets, en tenant compte de facteurs tels que le choix des réseaux de transport, les préférences pour le trajet le plus court et le moins de correspondances.

Travail effectué

dans l’exercice 1 j’ai utilisé 2 prédicats :
1/addh:Le prédicat addh prend une liste [H, M] représentant une heure et des minutes, ainsi qu'un nombre de minutes supplémentaires. Il calcule le total des minutes, convertit ce total en heures et minutes, puis assure que l'heure reste dans la plage de 0 à 23.
2/affiche:Le prédicat affiche une liste d'heures et de minutes et les affiche dans un format spécifique. Ces prédicats sont utiles pour manipuler et afficher des heures dans le contexte d'une application Prolog.
pour l’exercice 2 j’ai utilisé 3 prédicats:
1/lig : Ce prédicat vérifie si deux arrêts, Arrêt 1 et Arret2, appartiennent à la même ligne de transport (Ligne). Il récupère la liste des arrêts pour la ligne donnée à partir du prédicat ligne, vérifie la présence des arrêts spécifiés, et s'assure que l'arrêt 1 (Arrêt 1) précède l'arrêt 2 (Arrêt 2) sur la ligne.
2/ligtot: Ce prédicat étend lig en ajoutant une vérification du temps. Il prend deux arrêts, Arrêt 1 et Arret2, ainsi qu'un nom de ligne (Nom) et un horaire de départ représenté par [HR, MR]. Il utilise le prédicat lig pour vérifier l'appartenance à la même ligne, puis utilise les informations de la ligne pour comparer l'horaire donné avec l'horaire de départ le plus tôt.
3/ligtard: Ce prédicat est similaire à ligtot, mais il compare l'horaire donné avec le temps de départ le plus tardif. Il prend en compte l'intervalle entre les arrêts pour s'assurer que l'horaire donné est compatible avec le temps de départ et d'arrêt sur la ligne.
exercice 03:
1/itinTot:Le prédicat itinTot en Prolog planifie un itinéraire entre deux arrêts en considérant l'horaire de départ spécifié. Il utilise le prédicat ligtot pour vérifier la faisabilité de prendre une ligne de transport entre les deux arrêts à l'horaire donné. En récupérant les détails de la ligne, les horaires de départ sont convertis en minutes pour permettre la comparaison avec l'horaire spécifié par l'utilisateur. La liste des arrêts entre le point de départ et d'arrivée est obtenue, et le parcours, composé des noms des arrêts, est calculé. La vérification finale compare l'horaire. spécifié par l'utilisateur avec l'horaire de départ effectif de l'itinéraire. Si l'horaire de l'utilisateur est antérieur, l'itinéraire est retourné dans la variable Parcours, sinon, le prédicat échoue.
2/sommeIntervalle: Le prédicat sommeIntervalle calcule la durée totale du trajet entre deux arrêts sur une ligne de transport. Il utilise les horaires de départ, les intervalles entre les arrêts, et d'autres informations spécifiques à la ligne. La fonction trouverArrets identifie la liste des arrêts entre l'arrêt de départ et celui d'arrivée. Ensuite, les intervalles entre ces arrêts sont sommés pour obtenir la durée totale du

trajet en heures et minutes, retournée sous la forme d'une liste [Heures, Minutes] dans la variable Resultat.

3/ **trouverArrets:**Le prédicat trouverArrets permet de trouver la liste des arrêts entre deux stations, représentées par les arguments Arret1 (station de départ) et Arret2 (station d'arrivée), dans une liste d'arrêts donnée ListeArrets. La fonction utilise la construction append(_, [[Arret1, _] | Arrivees], ListeArrets) pour extraire la partie de la liste qui commence avec l'arrêt de départ et se termine à l'arrêt d'arrivée. Ensuite, la fonction vérifie si l'arrêt d'arrivée est présent dans cette liste extraite avec member([Arret2, _], Arrivees).

4/ **itinTard** :

Le prédicat itinTard vérifie la faisabilité d'un itinéraire entre deux stations en tenant compte d'une contrainte horaire. Il calcule la somme des intervalles entre les stations, détermine les temps de départ et d'arrivée, puis compare le temps donné par l'utilisateur avec le temps d'arrivée. Si le temps donné est suffisant, le prédicat réussit; sinon, il échoue.

exercice 04:

Les modifications apportées aux prédicats consistent à prendre en compte des options spécifiées par l'utilisateur lors de la planification du trajet. Ces options incluent le réseau de transport, la préférence pour la longueur du trajet, et le nombre préféré de correspondances. La logique a été ajoutée pour vérifier ces options lors de la recherche de trajets et pour adapter les résultats en conséquence. Cela permet à l'utilisateur de personnaliser davantage son expérience de planification de voyage en prenant en compte des critères spécifiques.

exercice 05:

1/interface_utilisateur : Ce prédicat représente l'interface utilisateur. Il affiche la liste des stations desservies par les transports publics, puis demande à l'utilisateur de saisir la station de départ, la station d'arrivée, le type de réseau (metro, bus, train), la préférence par rapport à la longueur du trajet (court, normal, long) et le nombre maximal de correspondances. Enfin, il appelle le prédicat parcours_possibles/5 avec les paramètres choisis par l'utilisateur.

2/afficher_itineraire : Ce prédicat prend en argument une liste représentant un itinéraire avec deux étapes (Arret1 et Arret2) et leurs horaires respectifs (HoraireDebut et HoraireFin). Il affiche ces informations de manière lisible.

Tests

Au fur et à mesure que j'avançais, j'avais évidemment besoin de m'assurer que l'ensemble des prédicats implémentés fonctionnaient correctement avant de les assembler dans d'autres prédicats plus importants.

- Prédicat addh :

?- addh([13, 34], 30, [14, 4]) .
true.

- Prédicat affiche :

?- affiche([5, 30]).
0530
true.

- Prédicat lig :

?- lig(Arret1, Arret2, Ligne) .
Arret1 = mairie_lilas,
Arret2 = porte_lilas,
Ligne = 11 Arret1 = mairie_lilas,
Arret2 = porte_lilas,
Ligne = 11

- Prédicat ligtot:

?- ligtot(porte_lilas, pyrenees, 11, [5, 30]).
330
315
false.

- Prédicat ligtard:

?- ligtard(mairie_lilas, chatelet, 11, [3, 30]).
210
315
true

- Prédicat itintot:

?- itinTot(mairie_lilas, republique, [5, 00], Parcours).
300
315
Parcours = [porte_lilas, telegraphe, place_fetes, jourdain, pyrenees, belleville, goncourt, republique, arts_metiers|...]
?- itinTot(mairie_lilas, republique, [5, 30], Parcours).
330
315
false.

- Prédicat itintard:

?- ligtard(mairie_lilas, chatelet, 11, [3, 30]).
210

315
true

- Prédicat ligtot1 :
?- ligtot1(mairie_lilas, republique, 11, [5, 0], [reseau(ferroviaire), prefLongueur(true), prefCorrespondances(2)]).
300
315
true

- Prédicat interface_utilisateur :
?- interface_utilisateur .
Stations desservies par les transports publics :
[station1, station2, station3, ...]

Choisissez une station de départ : station1.
Choisissez une station d'arrivée : |: station2.
Choisissez le type de réseau (metro, bus, train) : |: metro.
Préférence par rapport à la longueur du trajet (court, normal, long) : |: normal.
Nombre de correspondances maximal : |: 1.

Conclusion

Le projet de planificateur de trajets en Prolog propose une solution personnalisable pour la navigation dans les transports publics. L'interface utilisateur simplifiée permet aux utilisateurs de spécifier leurs préférences, telles que le type de réseau, la longueur du trajet et le nombre de correspondances. Les prédicats Prolog modulaires facilitent la maintenance et la compréhension du code. Le projet démontre l'efficacité de Prolog pour modéliser des problèmes complexes, offrant des itinéraires optimaux en tenant compte des préférences des utilisateurs et des différents réseaux de transport. En conclusion, le planificateur de trajets en Prolog constitue une approche flexible pour répondre aux besoins variés des utilisateurs dans le contexte des transports publics urbains, avec un potentiel d'extension pour des développements futurs.