TP de programmation logique en prolog

Objectif : Découvrir le langage Prolog et les concepts fondamentaux de la programmation logique à travers des exercices pratiques. À la fin du TP, vous serez capables de créer une base de connaissances, de définir des règles et de formuler des requêtes pour en déduire des informations. Vous approfondirez également la manipulation des listes et des structures de données en Prolog.

Contexte

Prolog (PROgrammation en LOGique) est un langage déclaratif où l'on se concentre sur **quoi** faire plutôt que sur **comment** le faire. Utilisé largement en intelligence artificielle et en résolution de problèmes complexes, Prolog permet d'exprimer des relations logiques sous forme de faits et de règles, et de laisser l'interpréteur Prolog raisonner sur ces relations.

Prérequis

- Savoir utiliser une interface en ligne de commande.
- Compréhension des concepts de base de la programmation (variables, fonctions).
- Aucune connaissance préalable en Prolog n'est requise.

Partie 1 : Installation et Introduction à Prolog (15 minutes)

1. **Installation de Prolog** :
    - Suivez les instructions pour installer **SWI-Prolog** :

      https://www.swi-prolog.org/download/stable
    - Une fois installé, lancez SWI-Prolog en utilisant la commande swipl dans votre terminal.

Vérification de l'installation :

- Dans l'interpréteur Prolog, tapez la commande suivante :

```
?- write('Bonjour Prolog!'), nl.
```

  Si Prolog est correctement installé, vous verrez Bonjour Prolog! s'afficher.

Partie 2 : Créer une Base de Connaissances Simple (30 minutes)

1. **Définir des faits** : Créez un fichier Prolog nommé famille.pl avec les faits suivants qui décrivent une famille simple :

```
homme(pierre).
homme(marc).
homme(paul).
femme(marie).
femme(sophie).

parent(pierre, paul).
parent(marie, paul).
parent(marc, sophie).
```

Définir des règles : Ajoutez des règles pour définir les relations de père et de mère :

```
père(X, Y) :- homme(X), parent(X, Y).
mère(X, Y) :- femme(X), parent(X, Y).
```

Charger la base de connaissances :

- Ouvrez Prolog et chargez votre fichier avec la commande :

```
?- consult('famille.pl').
```

Effectuer des requêtes : Posez des questions simples pour interroger votre base de connaissances :

```
?- père(pierre, paul).  % Qui est le père de Paul ?
?- mère(marie, paul).   % Qui est la mère de Paul ?
?- parent(marc, sophie). % Marc est-il parent de Sophie ?
```

Partie 3 : Exercices Pratiques (1h)

Exercice 1 : Ajouter de nouveaux faits et requêtes

Ajoutez les faits suivants dans votre fichier famille.pl :

```
homme(jacques).
parent(jacques, marc).
```

Ensuite, répondez aux questions suivantes :

1. Qui est le père de Marc ?

Jacques

2. Marc a-t-il des enfants ?

Sophie

Exercice 2 : Règle pour les Grands-parents

Ajoutez une règle pour définir ce qu'est un grand-parent :

grand_parent(X, Y) :- parent(X, Z), parent(Z, Y).

Posez les questions suivantes à Prolog :

1. Qui est le grand-parent de Paul ?

Paul n'a pas de grand-père

2. Jacques est-il grand-parent de Sophie ?

Oui

Exercice 3 : Règle pour les Frères et Sœurs

Ajoutez une règle pour définir les frères et sœurs :

frere_ou_soeur(X, Y) :- parent(Z, X), parent(Z, Y), not (X-Y).

Ensuite, posez les questions suivantes :

1. Paul a-t-il des frères ou des sœurs ? Non

Exercice 4 : Requêtes avec Variables

Posez des requêtes avec des variables pour explorer la base de connaissances :

1. Trouvez tous les hommes dans la base de données :

? – homme(X).
X = pierre ;
X = marc ;
X = paul ;
X = jacques.

2. Recherchez tous les parents de Sophie :

Marc est le père de Sophie.

Partie 4 : Introduction aux Listes en Prolog (30 minutes)

Prolog permet de manipuler des **listes**, une structure de données essentielle. Voici comment les listes fonctionnent dans Prolog et quelques exemples pratiques.

Exercice 5 : Manipuler des Listes

1. Déclarez une liste en Prolog :

```
?- L = [pierre, marie, paul, sophie].
```

Décomposer une liste :

- En Prolog, vous pouvez diviser une liste en tête et en queue :

```
?- [Tête | Queue] = [pierre, marie, paul].
```

Cela assignera Tête = pierre et Queue = [marie, paul].

Exercice 6 : Longueur d'une Liste

1. Ajoutez une règle pour calculer la longueur d'une liste :

```
longueur([], 0).
longueur([_ | Queue], N) :- longueur(Queue, M), N is M + 1.
```

Testez la règle en posant les requêtes suivantes :

```
?- longueur([pierre, marie, paul], N).  % Réponse attendue : N = 3
```

Exercice 7 : Trouver un Élément dans une Liste

Ajoutez une règle pour vérifier si un élément est présent dans une liste :

element(X, [X|_]).

element(X, [_|T]) :- element(X, T).

Posez la question suivante pour trouver si Marie est présente dans la liste contenant pierre, marie et paul

element(marie, [pierre, marie, paul]). OUI

Partie 5 : Défis supplémentaires (30 minutes)

Exercice 8 : Règle pour les Oncles et Tantes

Créez une règle pour définir les oncles et tantes :

oncle_ou_tante(X, Y) :- frere_ou_soeur(X, Z), parent(Z, Y).

Posez les questions suivantes :

1. Marc est-il l'oncle de Paul ?

Non

2. Quels sont les oncles de Sophie ?

Elle n’a pas d’oncle.

Exercice 9 : Arbre Généalogique Étendu

Étendez la base de connaissances pour inclure d'autres relations familiales. Ajoutez de nouveaux faits pour des cousins et des tantes :

```
parent(julie, sophie).
femme(julie).

% Règle pour les cousins
cousin(X, Y) :- parent(Z, X), frere_soeur(Z, W), parent(W, Y).
```

Interrogez la base de connaissances pour trouver les cousins de Paul :

D’après SWI-Prolog , Paul n’a pas de cousin.

Conclusion

À la fin de ce TP, vous avez exploré les concepts fondamentaux de Prolog, appris à manipuler des faits, des règles et des requêtes, et découvert comment utiliser des structures de données comme les listes. Vous avez également vu comment les relations logiques peuvent être utilisées pour résoudre des problèmes complexes.

Si vous voulez aller plus loin, vous pouvez essayer de modéliser un problème réel en Prolog, comme un petit jeu de logique ou un système expert simplifié !