Thème : Projet de stratification de programme Datalog

Auteurs :
M. YE Gildas
M. MOHAMED MAHAMOUD Naguib

1 Introduction

L'objectif de ce projet est l'implementation de l'algorithme de stratification pour des programmes Datalog.

Pour ce faire, nous devons concevoir et développer une application bureau (desktop) importable dans chacun de vos ordinateurs. Ce programme vous permettra de stratifier justement vos base de données.

2 Guide d'installation et d'utilisation

2.0.1 Implémentation

Pour l'implémentation du projet, nous avons utilisé le language de programmation Java à sa version 18. Nous avons opté pour JavaFX qui est un framework et une bibliothèque d'interface utilisateur issue du projet OpenJFX, qui permet de créer une interface graphique pour des applications de bureau qui permettra à l'utilisateur de choisir le fichier d'entré contenant les EDB et les IDB.

2.0.2 Lancement de l'application

Le programme est packagé sous format jar alors vous aurez besoin d'avoir java installé sur votre machine. Pour le lancer, ouvrez votre terminal puis placez-vous dans le dossier contenant le fichier .jar. Puis tapez la commande suivante :

```
$> java -jar stratification.jar
```

Une fenetre devrait s'ouvrir.

2.0.3 Utilisation

Pour lancer le programme de stratification vous devez lui fournir un fichier d'entrée contenant les EDB et les IDB portant l'extension .dl ou .txt. Cliquez sur le bouton selectionner puis choisissez votre fichier .dl ensuite cliquer sur stratifier.
Une fenêtre devrait s'ouvrir avec le résultat de la stratification. Le résultat de la stratification est également enregistrer dans votre repertoire Document sous le nom stratified_<nom_fichier.dl>[.txt]

Quelques petites règles pour une meilleur utilisateur :

— 1. Il faut que tous les faits ainsi que les règles commencent par une lettre majuscule.
— 2. Il faut terminer toujours par un point pour pas que le programme tourne à l'infini et qu'à la fin se plante.
— 3. Il faut preceder le caractère # pour rendre tout ce qui suit en commentaire. Il sera pas lu et pas prise en compte par notre parser.

3 Conclusion

En conclusion, ce travail nous a permis d'apprendre de nouveau. Ce projet nous a permis à travers l'implémentation du programme de stratification de mettre en pratique les notions apprises en cours de base de données déductives. De plus, ce projet nous a ouvert l'esprit un peu plus au niveau de gestion de base de données. Nous vous mettons ici un lien vers notre forge GitLab (pour tout ce qui est code) [1] et le .jar exécutable [2].

1. https://forge.univ-lyon1.fr/p2111931/projet_bdd/
2. https://forge.univ-lyon1.fr/p2111931/projet_bdd/-/blob/main/stratification.jar

4 Annexes

Dans cette partie vous trouverez les tests de programmes qu'on a utilisé pour voir que notre implementation foncitonne correctement. Vous trouverez également les fichiers sur la forge si besoin pour les exécuter.

4.1 Test 1

```
# EDB
F(1,2).
F(2,3).
F(3,1).
C(1,2).
C(1, R).
C(2,B).
C(3,J).

#IDB
S(X) :- P(X,X), not R(X,X).
R(X,Y) :- P(X,Y), C(X, R).
R(X,Y) :- P(X,Y), C(Y, R).
P(X,Y) :- P(X,Y).
P(X,Z) :- P(X,Y), F(Y,Z).
A(X) :- P(X,X), not S(X).

 Rsultat  :

P1 = {
      P(X, Y) :- P(X, Y).
      P(X, Z) :- P(X, Y), F(Y, Z).
      R(X, Y) :- P(X, Y), C(X, R).
      R(X, Y) :- P(X, Y), C(Y, R).
}

P2 = {
      S(X) :- P(X, X), not R(X, X).
}

P3 = {
```

```
      A(X) :- P(X, X), not S(X).
}

(P , 1)  (P , 1)  (A , 3)  (R , 1)  (R , 1)  (S , 2)
```

4.2 Test 2

```
# EDB
Personne(id, nom, age, sex).
Femme(x).
Homme(x).
Parent(x,y).

#IDB
Pere(x,y) :- Homme(x), Parent(x,y).
Mere(x,y) :- Femme(x), Parent(x,y).
Something(z) :- not Femme(c).

 Rsultat  :

P1 = {
      Pere(x, y) :- Homme(x), Parent(x, y).
      Mere(x, y) :- Femme(x), Parent(x, y).
}

P2 = {
      Something(z) :- not Femme(c).
}

(Something , 2)  (Pere , 1)  (Mere , 1)
```

4.3 Test 3

```
# EDB
```

```
Person(pID,name,sex,married).

# IDB
Man(x) :- Person(id,x,M,m).
Singleman(x) :- Man(x), not Husband(x).
Married(x) :- Person(id,x,s,1).
Husband(x) :- Man(x),Married(x).

 Rsultat  :
P1 = {
      Husband(x) :- Man(x), Married(x).
      Married(x) :- Person(id, x, s, 1).
      Man(x) :- Person(id, x, M, m).
}

P2 = {
      Singleman(x) :- Man(x), not Husband(x).
}

(Husband , 1)  (Married , 1)  (Man , 1)  (Singleman , 2)
```

4.4 Test 4

```
# IDB
MadCity ( to ) :  Flights ( _ , "Madison" , to , _ , _ , _ ).
MadCity ( to ) :  MadCity ( from ) , Flights ( _ , from , to , _ , _ ,
    _ ).
Ans ( fino ) :  Flights ( fino , from , to , _ , _ , _ ) , not MadCity
    ( from ) , not Flights ( fino , "Madison" , to , _ ,_ , _ ).

 Rsultat  :

P1 = {
      MadCity(to) :- Flights(_, Madison, to, _, _, _).
      MadCity(to) :- MadCity(from), Flights(_, from, to, _, _, _).
}

P2 = {
```

```
      Ans(fino) :- Flights(fino, from, to, _, _, _), not MadCity(from),
         not Flights(fino, Madison, to, _, _, _).
}
(Ans , 2)  (MadCity , 1)  (MadCity , 1)
```

4.5 Test 5

```
Q(x,z) :- T(z,y,v), T(x,y,a), not R(x, y).
R(x, y) :- R(y,x), T(b,v,x).
T(x, y, u) :- R(x,u), T(x,y,c).
S(x, z) :- R(z, b), not T(b, v, x).
P(y, x) :- P(a, x), not S(x, y), Q(x, y).

 Rsultat  :

P1 = {
      R(x, y) :- R(y, x), T(b, v, x).
      T(x, y, u) :- R(x, u), T(x, y, c).
}

P2 = {
      Q(x, z) :- T(z, y, v), T(x, y, a), not R(x, y).
      S(x, z) :- R(z, b), not T(b, v, x).
}

P3 = {
      P(y, x) :- P(a, x), not S(x, y), Q(x, y).
}

(P , 3)  (Q , 2)  (R , 1)  (S , 2)  (T , 1)
```

4.6 Test 6

```
Q(x, z) :- T(z, y, v), T(x, y, a), R(x, y).
```

```
R(x, y) :- Q(x, y), T(b, v, x), not S(x, y).
T(x, y, u) :- R(x, u), T(x, y, c).
S(x, z) :- R(z, b), not T(b, v, x).

 Rsultat  :

P1 = {
      Q(x, z) :- T(z, y, v), T(x, y, a), R(x, y).
}

P2 = {
      R(x, y) :- Q(x, y), T(b, v, x), not S(x, y).
      T(x, y, u) :- R(x, u), T(x, y, c).
}

P3 = {
      S(x, z) :- R(z, b), not T(b, v, x).
}
(Q , 1)  (R , 2)  (S , 3)  (T , 2)
```

4.7 Test 7

```
R(x) :- not Q(x, a).
Q(x, y) :- S(y), T(y, b).
T(x, y) :- R(y), Q(c, x).

 Rsultat  :

P1 = {
      Q(x, y) :- S(y), T(y, b).
}

P2 = {
      R(x) :- not Q(x, a).
      T(x, y) :- R(y), Q(c, x).
}

(Q , 1)  (R , 2)  (T , 2)
```

4.8 Test 8

```
person(x) :- Knows(x, y).
Person(y) :- Knows(x, y).
Person(x) :- Owns(x, y).
Unhappy(x) :- Person(x), not Happy(x).
Happy(x) :- Owns(x, iPad).
Happy(x) :- Knows(x, y), Happy(y).

 Rsultat  :

P1 = {
      Happy(x) :- Owns(x, iPad).
      Happy(x) :- Knows(x, y), Happy(y).
      person(x) :- Knows(x, y).
      Person(y) :- Knows(x, y).
      Person(x) :- Owns(x, y).
}

P2 = {
      Unhappy(x) :- Person(x), not Happy(x).
}

(Happy , 1)  (Happy , 1)  (person , 1)  (Unhappy , 2)  (Person , 1)
   (Person , 1)
```

4.9 Test 9

```
C(1,blue). C(2,red). C(3,red). C(4,red).
E(1,2). E(2,3). E(3,4). E(4,1).

V(X) :- E(X,Y).
V(Y) :- E(X,Y).
```

```
WCP(X,Y,R) :- E(X,Y), C(X,R).
WCP(X,Y,R) :- WCP(X,Z,S), E(Z,Y), C(Z,R), R != S.
WCP(X,Y,R) :- WCP(X,Z,S), E(Z,Y), C(Z,R), C(Y,T), R != T.
ExistsWCP(X,Y) :- WCP(X,Y,R).
ExistsWCP(X,X) :- V(X).
Answer(X,Y) :- V(X), V(Y), not ExistsWCP(X,Y).
```

Rsultat :

```
P1 = {
      WCP(X, Y, R) :- E(X, Y), C(X, R).
      WCP(X, Y, R) :- WCP(X, Z, S), E(Z, Y), C(Z, R), R, =, ., (, ,, ,,
          ), -, (, ,, ,, ), E(Z, Y), C(Z, R), C(Y, T), R, =, ., (, ,, ),
          -, (, ,, ,, ).
      ExistsWCP(X, X) :- V(X).
      V(X) :- E(X, Y).
      V(Y) :- E(X, Y).
}

P2 = {
      Answer(X, Y) :- V(X), V(Y), not ExistsWCP(X, Y).
}

(WCP , 1)  (WCP , 1)  (ExistsWCP , 1)  (Answer , 2)  (V , 1)  (V , 1)
```

4.10 Test 10

```
Souspiece (x,Y) :- Piece (x,y).
Souspiece (x,Y) :- Souspiece (x,Z), Souspiece (x,T).
Compose (x,Y) :- Souspiece (x,Y), not Souspiece (x,Z).
```

Rsultat :

```
P1 = {
      Souspiece(x, Y) :- Piece(x, y).
      Souspiece(x, Y) :- Souspiece(x, Z), Souspiece(x, T).
}
```

```
P2 = {
      Compose(x, Y) :- Souspiece(x, Y), not Souspiece(x, Z).
}

(Souspiece , 1)  (Souspiece , 1)  (Compose , 2)
```