Module Algorithmes et Raisonnement
S8 - 2A - 2023-2024
BE3

Mars - 2024

1 La base de données

- Soit une base de données avec 3 relations :
  - ***servir(Bar, Biere)***
  - ***frequenter(Buveur, Bar)***
  - ***aimer(Buveur, Biere)***

On travaillera avec la BD suivante (fichier fourni) :

```
servir('Les Guinguettes', 'Heinekin').          servir('Les Guinguettes', 'Leffe').
servir('Les Guinguettes', 'Export 33').         servir('Les Guinguettes', 'Leffe').
servir('Les Guinguettes', 'Pilsner').           servir('Les Guinguettes', 'Kronenbourg').
servir('Les Guinguettes', 'Quintor').

servir('Lucifer Inn', 'Grimbergen').            servir('Lucifer Inn', 'Castagnor').
servir('Lucifer Inn', 'Quintor').               servir('Lucifer Inn', 'Carlsberg').
servir('Lucifer Inn', 'Kanterbrau').            servir('Lucifer Inn', 'Pils').

servir('Cervisia Caupona', 'Force 4').          % 'Cervisia Caupona' = Taverne de Bière
servir('Cervisia Caupona', '1664').             servir('Cervisia Caupona', 'Chtis').
servir('Cervisia Caupona', 'Kronenbourg').      servir('Cervisia Caupona', 'Guinness').

servir('Auberge du Village', 'Fischer').        servir('Auberge du Village', 'Castagnor').
servir('Auberge du Village', 'Desperado').      servir('Auberge du Village', 'Grimbergen').

servir('Opus Satanae', 'Desperado').            servir('Opus Satanae', 'Castagnor').
%-------------------------------------------------------------------
frequenter('Jean', 'Les Guinguettes').          frequenter('Jean', 'Opus Satanae').
frequenter('Jean', 'Lucifer Inn').              frequenter('Jean', 'Cervisia Caupona').
frequenter('Jean', 'Auberge du Village').

frequenter('Marie', 'Lucifer Inn').             frequenter('Marie', 'Cervisia Caupona').

frequenter('Pierre', 'Auberge du Village').     frequenter('Pierre', 'Opus Satanae').

frequenter('Sara', 'Auberge du Village').       frequenter('Sara', 'Lucifer Inn').
frequenter('Sara', 'Cervisia Caupona').
%-------------------------------------------------------------------
aimer('Jean', 'Heinekin').                      aimer('Jean', 'Grimbergen').
aimer('Jean', 'Castagnor').                     aimer('Jean', 'Desperado').

aimer('Marie', 'Force 4').          aimer('Marie', 'Fischer').          aimer('Marie', 'Quintor').

aimer('Pierre', 'Quintor').         aimer('Pierre', '1664').            aimer('Pierre', 'Pilsner').

aimer('Sara', 'Kronenbourg').       aimer('Sara', 'Leffe').             aimer('Sara', 'Guinness').
```

☞ **Important** : on remarque que toutes les bières servies ne sont pas forcément appréciées de quelqu'un. ➙ Cette information sera utile dans la suite.

2 Avant de commencer

Les prédicats toutes-solutions

Pour obtenir l'ensemble (au sens algébrique, c'est à dire une séquence ordonnée et sans doublon) des réponses à une question, on utilise **setof** (voir l'exemple ci-après).
bagof est comme *setof* mais produit une liste de réponses (une séquence pas forcément ordonnée, et avec d'éventuels doublons).
☞ *setof* et *bagof* échouent si la question posée n'a aucune réponse.

En matière de "toutes-réponses", *findall* produit aussi une liste de réponses mais n'échoue pas et renvoie une liste vide si la question posée n'a pas de solution.

Exemples :

$setof(Bar, \_Bir \hat{} servir(Bar, \_Bir), L1)$ se lit : $\exists \_Bir, servir(Bar, \_Bir)$.

☞ **setof** a une contrainte d'utilisation particulière : les variables anonymes (comme *_Bir*) doivent y être quantifiées existentielle-ment.

Si on devait ne pas quantifier la variable anonyme *_Bir* et écrire simplement $servir(Bar, \_)$, on n'obtiendrait qu'une seule réponse.

Tester $setof(Bar, servir(Bar, \_Bir), L1)$

➛ Vous aurez les réponses une par une, pas toutes ensemble.

Comme indiqué ci-dessus, "bagof" peut être utilisé mais ne donne pas un ensemble (c-à-d. une séquence sans doublon et ordonné) mais une liste.

bagof et setof échouent si pas de réponse,
findall n'échoue pas si pas de réponse et construit une liste vide si aucun réponse n'est possible.

Par exemple sachant que **aimer(X, titi)** *échoue :*

```
?- bagof(X, aimer(X, titi),L).
   false (ou non selon le 'Prolog')
?- setof(X, aimer(X, titi),L).
   false
?- findall(X, aimer(X, titi),L).
   L = []
```

➛ On constate que *bagof* (comme *setof*) échoue s'il y a aucune réponse alors que *findall* renvoie une liste vide.
Pour cette raison, on propose *mon_setof/3* qui nous dispense de devoir quantifier nos variables et permet d'obtenir un **ensemble** de réponses.

2.1 "mon_setof" remplace setof

Pour éviter un échec dans l'utilisation de setof (en plus de devoir quantifier les variables anonymes), utiliser le prédicat suivant qui fait l'équivalent de *setof* en utilisant *findall*.

```
mon_setof(Terme, Buts, Set) :-   % vrai si 'Set' contient l'ensemble des 'Terme's où 'Buts' = vrai
    findall(Terme, Buts, Liste),    % lors de l'utilisation, 'Buts' ne doit pas être une variaable
    setof(Terme, Terme ^ member(Terme, Liste), Set),
    !.
% si Liste=[] dans findall, member échoue et mon_setof également. On ajoute donc le "!" et :
mon_setof(_Terme, _Buts, []).
```

Exemples : observez bien les requêtes et les résultats !

- Avec **setof** :

```
% | ?- setof(X, frequenter(X,_),L)
% L = ['Jean','Pierre','Sara'] ?;
% L = ['Jean','Marie','Sara'] ? ; .....
```

➙ Les réponses ne sont pas collectées dans L (il faut quantifier la variable anonyme).

- Avec **bagof** :

```
% | ?- bagof(X, frequenter(X,_),L)
% L = ['Jean','Pierre','Sara'] ?;
% L = ['Jean','Sara','Marie'] ?;
% L = ['Jean'] ?;
% L = ['Jean','Marie','Sara'] ?
```

➙ Les réponses ne sont pas collectées dans L.

- Avec **findall** (on obtient une liste avec des doublons) :

```
% | ?- findall(X, frequenter(X,_),L)
% L = ['Jean','Jean','Jean','Jean','Sara','Jean','Marie','Marie','Pierre','Pierre','Sara','Sara']
```

➙ Bien mais on veut un "ensemble" !

- Avec **mon_setof** :

```
% | ?- mon_setof(X, frequenter(X,_),L)
% L = ['Jean','Marie','Pierre','Sara']
```

➙ OK.

Un dernier mot : SWI-Prolog a un mécanisme particulier (et propre à lui) qui permet d'obtenir les réponses sans devoir quantifier les variables libres.
A l'occasion, testez ceci et comparez les résultats :

```
?- bagof(X, frequenter(X,_),L).
L = ['Jean', 'Pierre', 'Sara'] ;
L = ['Jean', 'Marie', 'Sara'] .


% notez "{X} /" : on met entre "{}" les variables dont la valeur  NOUS INTERESSE.
?- bagof(X, {X} / frequenter(X,_),L).
L = ['Jean', 'Jean', 'Jean', 'Jean', 'Jean', 'Marie', 'Marie', 'Pierre', 'Pierre'|...].


?- setof(X, {X} / frequenter(X,_),L).
L = ['Jean', 'Marie', 'Pierre', 'Sara'].
```

➙ Le dernier "setof" équivaut "mon_setof".

Passons maintenant aux questions sur notre base de données.

3 Questions

3.1 Q1

• Pour démarrer, poser quelques questions simples :
☞ On traitera le problème de réponse unique plus loin (à l'aide de *mon_setof*).

Q1_1 Toutes les bières servies par le bar 'Lucifer Inn' ?
Q1_2 Toutes les bières appréciées par 'Marie' ?
Q1_3 Tous les bars qui servent 'Grimbergen'.
Q1_4 Tous les buveurs : ceux qui fréquentent (au moins) un bar ? Utiliser frequnter/2.
Q1_5 Tous les bars ?

3.2 Q2- Vérifications des domaines

Qu' est-ce un **Domaine** (du discours) ? : l'ensemble de valeurs qu'une variable peut prendre. Par exemple, le domaine des buveurs est {jean, marie, ...} : tous les prénoms cités dans les clauses (les faits et règles).

On constate que dans les 3 relations de notre BD., chaque attribut est répété 2 fois. On veut vérifier si le domaine de chaque attribut varie ou pas d'une relation à l'autre.

Q2_1 Définir le domaine de l'attribut **bar** en vérifiant que les bars fréquentés par quelqu'un correspondent aux bars qui servent une bière.
➙ Cela revient à vérifier l'égalité d'es 'ensembles de solutions : l'ensemble des Bars issu de servir est-il le même que celui issu de la relation frequenter ?
➛ Est-ce que les résultats des deux relations suivantes sont identiques ?
$\{\exists Bar | servir(Bar, \_)\} \quad = \quad \{\exists Bar | frequenter(\_, Bar)\}$?

Q2_2 Vérifier si les bières servies (par un bar quelconque) correspondent aux bières appréciées (par quiconque).

Q2_3 Vérifier que les buveurs (qui apprécient une bière quelconque) correspondent à ceux qui fréquentent un bar (quelconque).
Exprimer les requêtes correspondantes.

3.3 Q3- dans quels bars 'Pierre' peut-il trouver une des ses bières favorites ?

• Les bars qui servent une bière appréciée par 'Pierre'
  ◦ Expliquer les réponses doubles.
  ◦ Si on veut les réponses uniques, on utilisera "*mon_setof*" (voir plus loin).
  ◦ D'une manière générale, on utilisera *mon_setof* à la place de *setof*. (voir 2.1 page 2)

3.4 Q4- les copains-de-Zinc de 'Pierre'

• Les buveurs qui vont dans les mêmes bars que 'Pierre'
  ◦ Expliquer les réponses doubles.

3.5 Q5- qui biberonne partout !

• Les buveurs qui fréquentent tous les bars (pour boire leur bière favorite ou pas).
☞ Vérifier les réponses que vous obtenez en écrivant le prédicat verifier_q7.????

3.6 Q6- buveurs étroits d'esprit

• Les buveurs qui fréquentent tous les bars ou on sert au moins une bière qu'ils aiment.
Ou de manière similaire :
les buveurs qui ne fréquentent QUE les bars ou on sert au moins une bière qu'ils aiment.

☞ Remarque : on n'est pas intéressé par les buveurs qui peuvent également fréquenter d'autres bars, pas exclusivement les bars ou on sert une bière qu'ils aiment) puisqu'on veut les buveurs qui vont dans ces bars pour boire une de leur bières favorites.

3.7 Q7- les buveurs qui ne veulent pas mourir de soif

• Les buveurs qui fréquentent au moins un bar où on sert une bière qu'ils aiment.

3.8 Q8- les buveurs qui ne vont pas forcément boire partout

• Les buveurs qui ne fréquentent **pas forcément** tous les bars où on sert une bière qu'ils aiment.

3.9 Q9- Banzaï : les buveurs qui veulent mourir de soif !

• Les buveurs qui ne fréquentent aucun bar où l'on sert une bière qu'ils aiment.
☞ Vérifier votre solution en la comparant avec les réponses à la questions précédente.

3.10 Q10

• pour tout buveur donné, calculer le nombre de bars qui servent une bière qu'il aime
On peut s'inspirer de *q3(Buv)* mais en plus, on compte les bars.

3.11 Q11

• les buveurs qui fréquentent tous les bars ou on sert au moins DEUX bière qu'ils aiment.

3.12 Q12

• Y a-t-il a un buveur qui va dans les mêmes bars que 'Marie' ?

3.13 Q13

• Il y a un Bar fréquenté par seul Pierre (aucun autre Buveur ne va dans ce Bar)
→ Donner ce buveur et le Bar en question.