

## TP2 :

## 1 Unification

## 1.1 Prédicat d'unification

Le prédicat prédéfini `=/2` est satisfait lorsque les deux termes sont unifiables, et dans ce cas l'unificateur le plus général est donné.

```
?- nourriture(pain,X) = nourriture(Y,saucisse).
X = saucisse,
Y = pain .

?- nourriture(pain,X) = nourriture(saucisse,Y).
false.
```

**Question 1** Prévoyez les réponses fournies par l'interprète PROLOG pour les couples de termes qui suivent :

1. `p(f(Y),W,g(Z))` et `p(U,U,V)` ;
2. `p(f(Y),W,g(Z))` et `p(V,U,V)` ;
3. `p(a,X,f(g(Y)))` et `p(Z,h(Z,W),f(W))` ;
4. `p(a,g(X,Y),X)` et `p(W,Z,f(Z))`.

**Question 2** Sans utiliser `=`, réalisez un prédicat `unifie/2` qui est satisfait lorsque les deux termes sont unifiables<sup>1</sup>.

## 1.2 Occur-check

Par défaut, SWI-PROLOG utilise un algorithme d'unification sans test d'occurrence (occur-check).

```
?- f(X) = X.
X = f(**) .
```

L'indication `f(**)` signifie un terme infini dans lequel la double étoile doit être remplacée par la variable substituée, autrement dit ici le terme `f(f(f(f(...))))` avec une infinité de `f`.

**Question 3** Quelle réponse va donner l'interprète au sujet de l'unification de `W` et `h(W,g(W))` ?

On peut demander à SWI-PROLOG d'appliquer l'occur-check dans son algorithme d'unification. Pour cela, on applique le prédicat prédéfini `set_prolog_flag/2` avec pour premier terme `occurs_check` et pour second terme `true` ou `error`.

```
?- set_prolog_flag(occurs_check,true).
true.
```

ou

```
?- set_prolog_flag(occurs_check,error).
true.
```

**Question 4** Testez le but `X=f(X)` avec l'option `true` puis avec l'option `error`.

## 1.3 Non unification

Le prédicat `\=/2` est satisfait lorsque les deux termes sont non unifiables. C'est le prédicat opposé à `=`.

**Question 5** Redéfinissez le prédicat `jaloux/2` de sorte qu'un individu ne puisse pas être jaloux de lui-même.

<sup>1</sup>Ne cherchez pas midi à quatorze heures, la réponse est `Simple=tres(Simple)`, sans occur-check.

## 2 Résolution

Considérons le programme PROLOG suivant

```
f(a).  
f(b).  
  
g(a).  
g(b).  
  
h(b).  
  
k(X) :- f(X), g(X), h(X).
```

Dans une session avec l'interprète SWI-PROLOG si on l'interroge sur le but **k(X)**, on obtient la seule réponse **X|b**.

```
?- [preuve].  
% preuve compiled 0.00 sec, 1,364 bytes  
true.  
  
?- k(X).  
X = b ;  
false.
```

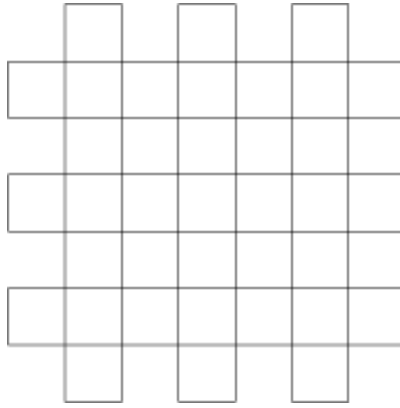
Comment l'interprète trouve-t-il cette unique solution? Pour le savoir, nous pouvons invoquer le prédicat prédéfini **trace/0**.

```
?- trace.  
Unknown message: query(yes)  
[trace] ?- k(X).  
  Call: (7) k(_G312) ? creep  
  Call: (8) f(_G312) ? creep  
  Exit: (8) f(a) ? creep  
  Call: (8) g(a) ? creep  
  Exit: (8) g(a) ? creep  
  Call: (8) h(a) ? creep  
  Fail: (8) h(a) ? creep  
  Redo: (8) f(_G312) ? creep  
  Exit: (8) f(b) ? creep  
  Call: (8) g(b) ? creep  
  Exit: (8) g(b) ? creep  
  Call: (8) h(b) ? creep  
  Exit: (8) h(b) ? creep  
  Exit: (7) k(b) ? creep  
X = b .
```

**Question 6** Dessinez l'arbre de résolution à partir de cette trace.

## 3 Mots croisés

Voici six mots italiens : *astante*, *astoria*, *baratto*, *cobalto*, *pistola* et *statale*.  
On doit les ranger à la façon des mots croisés dans la grille suivante :



La base de connaissances suivante représente un lexique contenant ces mots :

```

mot(astante,a,s,t,a,n,t,e).
mot(astoria,a,s,t,o,r,i,a).
mot(baratto,b,a,r,a,t,t,o).
mot(cobalto,c,o,b,a,l,t,o).
mot(pistola,p,i,s,t,o,l,a).
mot(statale,s,t,a,t,a,l,e).

```

Le but est d'écrire un prédicat `motscroises/6` qui nous dise comment remplir la grille. Les trois premiers arguments devront être les mots verticaux de gauche à droite et les trois derniers les mots horizontaux du haut en bas.

**Question 7** Donnez une réalisation de ce prédicat qui donne des solutions dans lesquelles un même mot peut être répété dans la grille.

**Question 8** Testez votre prédicat. Combien y a-t-il de solutions ?

## 4 Générateur de phrases

Nous considérons une langue dans laquelle toutes les phrases ont la même structure :

**determinant nom verbe determinant nom.**

Par exemple, **le chat mange la souris** ou bien **l' étudiant écoute le professeur** sont de telles phrases.

**Question 9** Définissez un petit lexique à l'aide d'un prédicat `mot/2`. Ce prédicat permet de définir un certain nombre de mots (ceux que vous voulez) accompagné de leur attribut (**determinant**, **nom** ou **verbe**).

**Question 10** Définissez un prédicat `phrase/5` qui est satisfait lorsque ses cinq arguments dans l'ordre forment bien une phrase.

**Question 11** Prévoyez en fonction du lexique que vous avez choisi le nombre de phrases et l'ordre dans lequel elles apparaissent lorsque vous interrogez l'interprète PROLOG sur le prédicat `phrase`.