UE ELFE - Programmation Logique

Interrogation 1

durée 1h – documents autorisés

Exercice 1 : Un peu d'unification

Q 1 . Pour chacun des ensembles de termes suivants, dire si les termes qu'il contient s'unifient ? Justifier votre réponse, si c'est le cas on donnera l'unificateur le plus général.

- $E_1 = \{f(Y, g(V), g(a) + V); f(Z, g(g(b)), X + g(b))\}$
- $E_2 = \{f(a, g(V), X + V); f(a, g(g(V)), g(a) + g(b))\}$
- $E_3 = \{f([X|[X|R]]); f([a, a, b])\}$
- $E_4 = \{couleur(X); couleur([Z|[a|T]])\}$

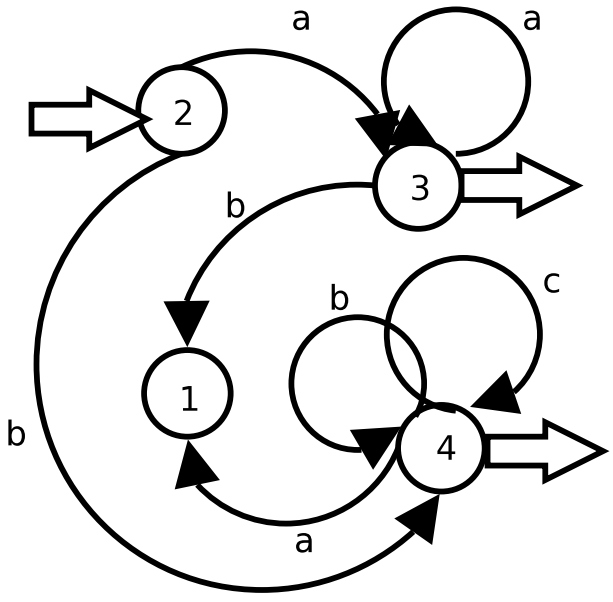
Q 2 . Soit la substitution $\sigma = \{V|b, Y|3, Z|3, X|g(a)\}$, calculer σE_1, que peut on dire de σ ?

Q 3 . Donner un terme qui s'unifie avec toute liste de longueur au moins 2.

Exercice 2 : Automates

(Aucune connaissance spécifique sur la notion d'automate n'est nécessaire pour traiter cet exercice, les définitions utiles sont données dans le sujet).

On décide de représenter un automate par un ensemble de faits portant sur les prédicats **alphabet(***L***)** (*L* est une lettre possible sur une transition), **etat(***E***)** (*E* est un etat de l'automate), **initial(***E***)** (*E* est un etat initial), **final(***E***)** (*E* est un etat final) et **transition(***D,L,A***)** (il existe une transition de l'état *D* vers l'état *A* avec la lettre *L*).
Voici un exemple d'automate et les faits qui le définissent :




```
alphabet(a).
alphabet(b).
alphabet(c).

etat(1).
etat(2).
etat(3).
etat(4).

initial(2).
final(3).
final(4).

transition(2,a,3).
transition(2,b,4).
transition(4,b,4).
transition(4,a,1).
transition(4,c,4).
transition(3,a,3).
transition(3,b,1).
```

Les prédicats demandés par la suite doivent "fonctionner" pour n'importe quel automate défini par l'ensemble de ces faits (et pas uniquement pour celui donné en exemple !).

Q 1 . Que donne l'évaluation des buts suivants :

Q 1.1. `?- final(X).`

Q 1.2. `?- transition(1,_,_).`

Q 1.3. `?- transition(X,Y,X).`

Q 1.4. `?- transition(2,A,_).`

Q 2 . Un automate est non déterministe,

- s'il y a plusieurs états initiaux (entrées).
- ou bien s'il existe un état d'où partent des transitions étiquetées par la même lettre de l'alphabet et aboutissant à des états différents.

Réaliser un prédicat sans paramètre **non_deterministe** qui est satisfait si l'automate est non déterministe.

Q 3 . Un automate n'est pas complet s'il existe un état E, et une lettre L de l'alphabet, telle qu'il n'existe pas de transitions partant de l'état E et étiquetée par L.

Réaliser un prédicat sans paramètre **non_complet** qui est satisfait si l'automate est non complet.

Exercice 3 : Listes

Q 1 . Le prédicat **prefixe(***?Liste,?Prefixe***)** est satisfait si la liste *Liste* a pour préfixe la liste `Prefixe`.

Ainsi :

```
?- prefixe([a,b,c,d],[a,b]).
Yes
?- prefixe([a,b,c,d],[c,d]).
No
```

Q 1.1. Codez ce prédicat sans utiliser le prédicat **append**.

Q 1.2. Codez **prefixe** en utilisant le prédicat **append**.

Q 2 . Le prédicat **dissoutDans(***?L1,?L2***)** est satisfait si les éléments de la liste *L1* se retrouvent tous, dans le même ordre dans *L2* mais éventuellement séparés par d'autres éléments de *L2*.

Ainsi :

```
?- dissoutDans([],[1,a,2,3,b,4,c,5]).
Yes
?- dissoutDans([a,b,c],[1,a,2,3,b,4,c,5]).
Yes
?- dissoutDans([a,b,c],[b,1,a,2,3,b,4,c,5]).
Yes
?- dissoutDans([a,b,c],[1,a,2,3,c,b,5]).
No
```

Codez ce prédicat.

Q 3 . Le prédicat **separePairsImpairs(**+*LEntiers*,-*LPairs*,-*LImpairs***)** est satisfait si la liste *LPairs*, respectivement *LImpairs*, est unifiée avec la liste des entiers pairs, respectivement impairs, de la liste d'entiers *LEntiers*. L'ordre des entiers pairs et impairs de chacune des listes respecte l'ordre initial dans *LEntiers*.

Ainsi :

```
?- separePairsImpairs([1,2,3,4,5],LP,LI).
LP = [2,4]  LI = [1,3,5]
?- separePairsImpairs([1,3,5],LP,LI).
LP = []  LI = [1,3,5]
```

Q 3.1. Codez **separePairsImpairs**.

Q 3.2. Proposez une solution utilisant le *cut* (!) et évitant des calculs inutiles.

Q 3.3. Donnez l'arbre de résolution du but `separePairsImpairs([1,2,3],LP,LI).` pour votre version "avec *cut*".