# TD : unification et résolution

## Unification

Rappelons la définition du type `terme` :

```
type ('a,'b) term =
    Var of 'b
  | Operation of 'a * ('a,'b) term list ;;
```

**Exercice 1.** Écrire l'algorithme d'unification en CAML.

**Exercice 2.** Appliquer l'algorithme précédent pour dire si les couples suivantes sont unifiables, et dans ce cas produire une substitution principale.
- $(\, x + s(y) * s(z)\ ,\ 7 + s(5) * s(s(x)))$
- $(\, x + f(y, 5)\ ,\ g(x) + f(g(x), 5)\,)$
- $(\, f(z, f(y, c))\ ,\ f(g(w), x)\,)$.

**Exercice 3.** Appliquer l'algorithme d'unification pour calculer le types des expressions CAML suivantes :
- `fun x -> fun f -> f(x + 3) ; ;`
- `fun f a l -> f a : :l ; ;`
- `fun f a l -> f (a : :l) ; ;`

## Résolution

Sur l'île des cavaliers, larrons , et loups, chaque habitant est un cavaliers ou un larrons. Il peut être ou un loup garou, dans ce cas il mange le hommes dans les nuits de pleine lune. Les cavaliers disent la vérité, les ladrons mentent. Un loup garou peut être cavalier ou larron.
Un explorateur arrive sur l'île et rencontre `a`, `b` et `c`. Entre eux il y a un loup garou.
- `a` dit que `b` est un loup.
- `b` dit qu'il n'est pas un loup.
- `c` dit qu'au moins deux entre eux sont des ladrons.

Qui doit choisir l'explorateur comme compagnon de voyage ?

**Exercice 4.** Formaliser ce problème en logique du premier ordre.

**Exercice 5.** Comment utiliser la résolution pour obtenir une réponse ?

# Un démonstrateur automatique

On veut implémenter en OCaml un démonstrateur automatique. Le code de haut niveau pour ce démonstrateur est comme il suit :

**saturer.code**

```
 1 : Procédure saturer(T)
 2 : T : ensemble de clauses
 3 : {
 4 :
 5 :    WO (worked off) (* : ensemble de clauses déjà elaborées *)
 6 :    Us (usable)     (* : enseble de clause utilisables*)
 7 :
 8 :    tant-que( vrai )
 9 :    {
10 :       Us := T
11 :       WO := ensemble vide
12 :
13 :       si Us contient la clause vide
14 :         retourner "théorie contradictoire''
15 :
16 :       si Us est vide,
17 :         retourner "ensemble saturé"
18 :
19 :       choisir C in Us
20 :       ajouter C à WO
21 :       enlever C de Us
22 :       appliquer toutes les règles entre C et WO
23 :         et obtenir un nouvel ensemble de clauses New
24 :       simplifier l'ensemble New (par rapport à New, Us et WO)
25 :       simplifier WO et US (par rapport à new)
26 :       ajouter les clause New à Us
27 :    }
28 : }
```

**Exercice 6.** Organiser le travail d'implémentation :
- Définir de modules, de sous modules, et des relations de dépendance entre les modules.
- Pour chaque module, définir son interface (fichier `.mli`).
- Partager le travail d'implémentation en groupes.