# TD 5

## 1 Unification

Finir le TP 4, et implanter l'algorithme d'unification.

## 2 Réécriture

À l'aide du type `term` défini dans le TP précédent, implanter les fonctions suivantes :
- `pos`, qui calcule le sous-terme d'un terme $t$ à une position $p$ donnée.
- `remplacement`, qui calcule $t[s]_p$.

## 3 Induction sur les listes

Définir les fonctions suivantes sur les listes[1] :
- `sum`, qui prend la somme de tous les éléments d'une liste d'entiers.
- `length`, qui calcule la longueur d'une liste.
- `append`, qui concatène deux listes.
- `concat`, qui prend une liste de listes et concatène tous ses éléments. Par exemple :
```
# concat [[1;2]; []; [3;4;5]] ;;
- : int list = [1; 2; 3; 4; 5]
```
- `modulo`, qui prend une liste et renvoie la liste des éléments congrus à 0 modulo 2
- `filter`, qui prend un prédicat et une liste, et renvoie tous les éléments de la liste qui vérifient le prédicat. Exemple :
```
# filter (fun x -> x mod 2 = 0) [1; 2; 3; 4; 5] ;;
- : int list = [2; 4]
```

1. À l'exception de `sum`, ces fonctions sont prédéfinies dans le module `List` de Caml et sont préfixées par le nom du module, par exemple `List.length`. La fonction `append` s'écrit habituellement avec infixe `@`.