— INFORMATIQUE — MIAS/SM 1 — TP 6 —

Listes et calcul des prédicats

Exercice 1 : Calcul des prédicats – quantificateurs

Un *prédicat* est une fonction dont le codomaine (c'est-à-dire l'ensemble des résultats) est le type booléen. Autrement dit, un prédicat est tout simplement une fonction qui retourne un booléen. Ainsi, par exemple, les deux fonctions suivantes (que nous avons déjà vues) sont des prédicats :

```
let pair  = function n -> (n mod 2) = 0 ;;
let ou    = function a -> function b -> a or b ;;
```

Parmi ces prédicats, on appelle *quantificateurs* les fonctions dont les arguments sont un ensemble de valeurs et un prédicat sur ces valeurs. Les quantificateurs les plus connus sont notés $\forall$ (appelé « quel que soit » ou « pour tout ») et $\exists$ (« il existe »). Leur définitions mathématiques sont respectivement :

$$\begin{aligned} \forall\, x \in \{x_1, \ldots, x_n\},\, P(x) &\iff P(x_1) \text{ et } \ldots \text{ et } P(x_n) \text{ et vrai} \\ \exists\, x \in \{x_1, \ldots, x_n\} / P(x) &\iff P(x_1) \text{ ou } \ldots \text{ ou } P(x_n) \text{ ou faux} \end{aligned}$$

En admettant qu'on représente un ensemble de valeurs par la liste de ces valeurs, on souhaite écrire en Caml ces deux quantificateurs (à partir de leur définitions mathématiques). Ces quantificateurs seront appelés respectivement `for_all` et `exists`, et seront tous deux de la forme `('a -> bool) -> 'a list -> bool`. Écrivez le code Caml correspondant à chacun (il existe deux possibilités, une forme directe et une utilisant `map` et `it_list` ; choisissez celle que vous préférez).

Exercice 2 : Calcul des prédicats – relations binaires

On représente encore une fois un ensemble de valeurs par la liste de ses valeurs. On s'intéresse maintenant au relations binaires : on souhaite déterminer si une relation $\mathcal{R}$ donnée est une relation d'équivalence.

Plutôt que de définir une fonction (ce qui est un peu lourd), on représente une relation binaire par un ensemble de couples vérifiant cette relation. Autrement dit, la relation est définie par son graphe sur son ensemble de définition. Ainsi, par exemple, la liste `[(1, 2) ; (1, 3) ; (2, 3)]` représente la relation $<$ sur l'ensemble {1,2,3}.

Pour vérifier si une telle relation est une relation d'équivalence, on doit vérifier qu'elle est réflexive, symétrique et transitive.

.../...

1. Le plus simple est de vérifier la symétrie d'une relation. Il suffit pour cela d'utiliser le quantificateur « quel que soit » défini dans l'exercice précédent, ainsi que le prédicat `mem` qui vérifie l'appartenance d'un élément à une liste :

$$\mathcal{R} \text{ symétrique} \iff \forall\ (x,y),\ (x\ \mathcal{R}\ y) \Longrightarrow (y\ \mathcal{R}\ x)$$

2. Pour vérifier la réflexivité, on utilise encore une fois sa définition mathématique :

$$\mathcal{R} \text{ réflexive} \iff \forall\ x \in \mathcal{D}\text{ef}(\mathcal{R}),\ (x\ \mathcal{R}\ x)$$

   On utilisera à nouveau les prédicats `for_all` et `mem`, auxquels il faut rajouter une fonction `ens_def` calculant l'ensemble de définition d'une relation binaire. Pour définir cette dernière fonction, on pourra commencer par en écrire une première version appelée `ens_def_simple` proposant un ensemble contenant éventuellement des éléments en double, avant d'éliminer ces éléments grâce à une fonction nommée `unique` (à définir).

3. Pour finir, on souhaite vérifier la transitivité. Rappelons que la définition mathématique associée s'écrit :

$$\mathcal{R} \text{ transitive} \iff \forall\ (x,y,z),\ (x\ \mathcal{R}\ y) \text{ et } (y\ \mathcal{R}\ z) \Longrightarrow (x\ \mathcal{R}\ z)$$

   Utilisez cette définition pour écrire un prédicat `transitive` vérifiant la transitivité d'une relation binaire donnée en argument.
   En déduire un prédicat `equivalence` vérifiant si une relation binaire est une relation d'équivalence.

Si le cœur vous en dit, vous pouvez prolonger cet exercice en définissant un prédicat vérifiant si une relation binaire est une relation d'ordre (c'est-à-dire réflexive, antisymétrique et transitive).