

## Preuves de programmes – Projet

On considère un damier de dimensions  $3 \times 3$ . Sur chaque case se trouve un jeton bicolore : blanc sur une face et noir sur l'autre (une seule des deux faces est visible). À chaque étape il est possible de retourner une rangée ou une colonne du damier. On cherche ici à examiner les configurations qu'il est possible d'atteindre à partir d'une configuration donnée.

### Exercice 1 – Modélisation du damier

- Définir un type `color` à deux valeurs `White` et `Black` ainsi qu'une fonction `inv_color : color → color` qui échange les deux couleurs.
- Ouvrir une section avec la commande `Section Triple`, et déclarer une variable `X : Type` à l'intérieur de cette section à l'aide de la commande `Variable X : Type`.
  - Définir un type `triple : Type` des triplets  $(x, y, z)$  d'éléments de `X`.
  - Définir une fonction `triple_x : X → triple` qui à tout  $x : X$  associe le triplet  $(x, x, x)$ .
  - Définir une fonction `triple_map : (X → X) → triple → triple` qui à une fonction `f` et à un triplet  $(x, y, z)$  associe le triplet  $(f(x), f(y), f(z))$ .
- On définit un type des positions à l'aide de la commande
 

```
Inductive pos : Type := A : pos | B : pos | C : pos.
```

  - Définir une fonction `triple_proj : pos → triple → X` qui extrait une composante d'un triplet donnée par sa position.
  - Définir une fonction `triple_map_select : (X → X) → pos → triple → triple` qui applique une fonction à une composante d'un triplet donnée par sa position.

Fermer la section avec la commande `End Triple`, et vérifier le type des objets introduits.

- Définir le type `board` des configurations comme le type des triplets de triplets de couleurs, ainsi qu'un objet `white_board` qui modélise la configuration blanche partout. Définir en Coq les deux configurations suivantes :

$$\text{start} = \begin{array}{|c|c|c|} \hline \square & \square & \blacksquare \\ \hline \blacksquare & \square & \square \\ \hline \blacksquare & \blacksquare & \blacksquare \\ \hline \end{array} \qquad \text{target} = \begin{array}{|c|c|c|} \hline \blacksquare & \blacksquare & \blacksquare \\ \hline \square & \blacksquare & \square \\ \hline \blacksquare & \blacksquare & \square \\ \hline \end{array}$$

### Exercice 2 – Manipulation des configurations

- Définir la fonction `board_proj : board → pos → pos → color` qui extrait le contenu d'une case d'une configuration donnée.
- Définir les fonctions `inv_row, inv_col : board → pos → board` qui inversent respectivement une rangée et une colonne d'une configuration donnée.
- Définir une relation `move : board → board → Prop` telle que `move b1 b2` exprime que `b2` s'obtient à partir de `b1` en inversant une rangée ou une colonne. Montrer que cette relation est symétrique.

4. Définir inductivement la relation `moves` : `board`  $\rightarrow$  `board`  $\rightarrow$  `Prop` à partir des deux règles :

$$\frac{}{\text{moves } b \ b} \qquad \frac{\text{moves } b1 \ b2 \quad \text{move } b2 \ b3}{\text{moves } b1 \ b3}$$

Montrer que cette relation est réflexive, symétrique et transitive.

5. Vérifier que `moves start target`. Comment montrer que  $\neg(\text{moves white\_board start})$  ?

### Exercice 3 – L’invariant de normalisation

1. Définir une fonction `force_white` : `board`  $\rightarrow$  `board` qui inverse certaines rangées et/ou certaines colonnes d’une configuration donnée de manière à ce que la première rangée et la première colonne de la configuration retournée par cette fonction soient entièrement blanches. Vérifier que pour toute configuration `b` on a `moves b (force_white b)`.
2. Montrer que `move b1 b2`  $\rightarrow$  `force_white b1 = force_white b2`.
3. Montrer que `moves b1 b2`  $\leftrightarrow$  `force_white b1 = force_white b2`.
4. En déduire que  $\neg(\text{moves white\_board start})$ .

### Exercice 4 – Décidabilité de la relation moves

En Coq, on exprime qu’une relation `R` (définie sur un type de données `X`) est décidable par la proposition

$$\text{forall } x \ y : X, \{R \ x \ y\} + \{\sim R \ x \ y\}$$

où  $\{A\} + \{B\}$  désigne la disjonction calculatoire (définie dans `Type`)<sup>1</sup>.

1. Montrer que l’égalité `x = y` est décidable sur le type `color`.
2. Montrer que si l’égalité est décidable sur un type `X`, alors elle est décidable sur `triple X`.  
En déduire qu’elle est décidable sur le type `board`.
3. À l’aide de ce qui précède (exercice 2), montrer que la relation `moves` est décidable.
4. Extraire le programme correspondant et tester.

---

<sup>1</sup>Cette forme de tiers-exclu calculatoire ne peut pas être déduite du tiers-exclu sur les propositions.