TD9 — Observateurs et vérification dynamique

1. Motivation

On souhaite maintenant vérifier qu'à l'exécution d'un programme, certaines propriétés *dynamiques* soient respectées : par exemple, pour l'interprète, que la fonction «avancer» soit toujours précédée d'une «mesure» par le robot de la case devant lui.

Pour cela, on va utiliser un *observateur*, c'est-à-dire une *instrumentation* du programme de test dédiée à la vérification à l'exécution de cette propriété.

2. Définition de l'observateur

Exercice 1. Donner sous forme d'automate d'états fini la propriété «avant d'avancer, une mesure est toujours effectuée».

Exercice 2.

1. Écrire un exemple de programme-robot *correct*, *accepté* par l'observateur.
2. Écrire un exemple de programme-robot *incorrect*, *rejeté* par l'observateur.
3. Écrire un exemple de programme-robot *correct*, *rejeté* par l'observateur.
4. Écrire un exemple de programme-robot *incorrect*, *accepté* par l'observateur.

3. Implémentation de l'observateur

Exercice 3. Spécifier un paquetage `observateur`, fournissant les types de données et fonctions permettant d'instrumenter un programme. Écrire l'implémentation de ce paquetage pour l'automate défini à l'exercice 1.

Exercice 4. Modifier le paquetage `environnement` (dont la spécification est rappelée en annexe), en utilisant le paquetage `observateur`, pour permettre la vérification à l'exécution de la propriété associée à l'observateur.

A. Spécification du paquetage environnement

```
#ifndef _ENVIRONNEMENT_H_
#define _ENVIRONNEMENT_H_

#include "robot.h"
#include "terrain.h"

/* Environnement : terrain + robot */

typedef struct {
  Robot r;
  Terrain t;
} Environnement;

/* Initialise l'environnement envt :
   - lit le terrain dans le fichier fichier_terrain
   - initialise le robot : coordonnées initiales lues dans le fichier
   terrain, orientation initiale vers l'est
*/
erreur_terrain initialise_environnement(Environnement *envt,
                                        char *fichier_terrain);

/* Résultat d'un déplacement de robot */
typedef enum {
  OK_DEPL, /* Déplacement sur case libre */
  PLOUF, /* Déplacement dans l'eau */
  CRASH, /* Déplacement dans un rocher */
  SORTIE, /* Sortie du terrain */
} resultat_deplacement;

/* Avancer le robot sur le terrain : */
resultat_deplacement avancer_envt(Environnement *envt);

/* Tourner le robot àgauche */
void gauche_envt(Environnement *envt);

/* Tourner le robot àdroite */
void droite_envt(Environnement *envt);

/* Effectuer une mesure
   Paramè tre d : la direction de la mesure
     0 sur place
     1 devant
     2 devant droite
     3 droite
     4 derriè re droite
     5 derriè re
     6 derriè re gauche
     7 gauche
     8 devant gauche
   Renvoie le résultat de la mesure :
     0 rien (case libre ou en-dehors du terrain)
     1 eau
     2 rocher
 */
int mesure_envt(Environnement *envt, int d);

/* Afficher le terrain avec la position et l'orientation du robot */
void afficher_envt(Environnement *envt);

#endif
```