

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 packaging environnement

```
1  #ifndef _ENVIRONNEMENT_H_
2  #define _ENVIRONNEMENT_H_
3
4  #include "robot.h"
5  #include "terrain.h"
6
7  /* Environnement : terrain + robot */
8
9  typedef struct {
10     Robot r;
11     Terrain t;
12 } Environnement;
13
14 /* Initialise l'environnement envt :
15  - lit le terrain dans le fichier fichier_terrain
16  - initialise le robot : coordonnées initiales lues dans le fichier
17  terrain, orientation initiale vers l'est
18 */
19 erreur_terrain initialise_environnement(Environnement *envt,
20                                         char *fichier_terrain);
21
22 /* Résultat d'un déplacement de robot */
23 typedef enum {
24     OK_DEPL, /* Déplacement sur case libre */
25     PLOUF, /* Déplacement dans l'eau */
26     CRASH, /* Déplacement dans un rocher */
27     SORTIE, /* Sortie du terrain */
28 } resultat_deplacement;
29
30 /* Avancer le robot sur le terrain : */
31 resultat_deplacement avancer_envt(Environnement *envt);
32
33 /* Tourner le robot à gauche */
34 void gauche_envt(Environnement *envt);
35
36 /* Tourner le robot à droite */
37 void droite_envt(Environnement *envt);
38
39 /* Effectuer une mesure
40  Paramètre d : la direction de la mesure
41  0 sur place
42  1 devant
43  2 devant droite
44  3 droite
45  4 derrière droite
46  5 derrière
47  6 derrière gauche
48  7 gauche
49  8 devant gauche
50  Renvoie le résultat de la mesure :
51  0 rien (case libre ou en-dehors du terrain)
52  1 eau
53  2 rocher
54 */
55 int mesure_envt(Environnement *envt, int d);
56
57 /* Afficher le terrain avec la position et l'orientation du robot */
58 void afficher_envt(Environnement *envt);
59
60 #endif
```