Compte Rendu Projet Curiosity

Nguyen Phuoc Loc Luu
Diallo Mamadou

TP7 Explication de la démarche suivie pour écrire les tests
On a choisi de présenter le test qui couvre plus fontions possibles pour le test fonctionnel

Description du programme-robot SIMPLE9.PRG

3 M { A C } {A} 7 M { D A X} { A } ??

3 M: Effectue une mesure à droite du robot.
{ A C } {A} ?: Si la mesure n'est pas égale à 0, avance (A) et clone (C), sinon avance (A).
7 M: Effectue une mesure à gauche du robot.
{ D A X} { A } ?: Si la mesure n'est pas égale à 0, avance (A) puis tourne à droite (D), sinon avance (A).
?: Exécute la condition précédente.

Finalemet, on a fait des exemple de programmes ayant une erreur à la lecture de programme (test de robutesse) : robutesse1 « { A A » ici on a pas ferme l'accolade « } » pour indique la fin de boucle et robutesse2 « Z » ici la lettre Z n'est pas dans le programme.h

TP8 Description des deux programmes-robots
INFINI_2.PRG
{ 1 M { G } { A } ? C ! } C !

1 M: Effectue une mesure devant le robot.
{ G } { A } ?: Si la mesure n'est pas égale à 0, tourne à gauche (G), sinon avance (A).
C !: Clone l'élément en haut de la pile.
C !: Clone l'élément en haut de la pile.

INFINI_5.PRG
{1 M {1 M {D A} {G} ?} {A} ? C !} C !

1 M: Effectue une mesure devant le robot.
-{1 M {D A} {G} ?}: Effectue une mesure devant le robot. Si la mesure n'est pas égale à 0, tourne à droite (D) et avance (A), sinon tourne à gauche (G).
-{A} ?: Si la mesure n'est pas égale à 0, avance (A).
→ Si 1M au debut est n'est pas 0, effectuer 2e tiret, sion 3e.
C !: Clone l'élément en haut de la pile.
C !: Clone l'élément en haut de la pile.

Ces programmes utilisent des mesures et des actions conditionnelles pour contrôler les mouvements du robot, en utilisant des commandes pour avancer, tourner et cloner des éléments sur la pile (programmes infinies).

Usage: ./curiosity-perf <fichier_programme> <N> <L> <H> <d> <graine> <nb_step_max> <fichier_res>

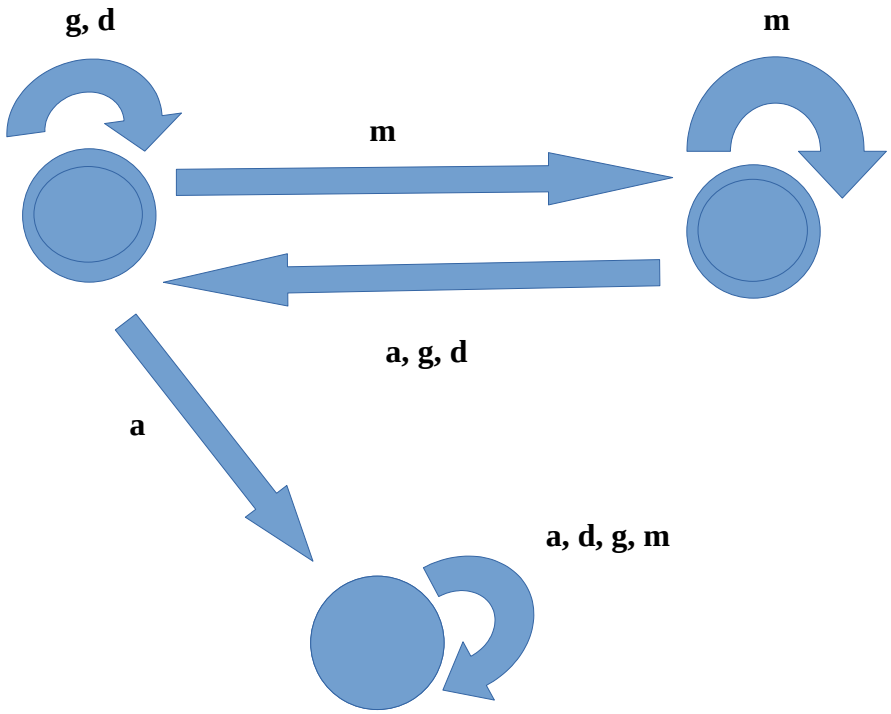
luunpl@im2ag-turing-01:[~/304_INF/TP/Projet_Curiosity]: ./curiosity-perf Tests_TP8/infini_2.prg 1000 11 9 0.3 123 1000 fichier_res.txt
Nombre de terrains testés: 1000
Nombre de sorties: 715 (71.50%)
Nombre de terrains bloqués: 285 (28.50%)
Nombre de collisions avec obstacles: 0 (0.00%)
Nombre moyen de pas pour les sorties: 100.20
Observateur : Propriete valide

luunpl@im2ag-turing-01:[~/304_INF/TP/Projet_Curiosity]: ./curiosity-perf Tests_TP8/infini_5.prg 1000 11 9 0.3 123 1000 fichier_res.txt
Nombre de terrains testés: 1000
Nombre de sorties: 445 (44.50%)
Nombre de terrains bloqués: 59 (5.90%)
Nombre de collisions avec obstacles: 496 (49.60%)
Nombre moyen de pas pour les sorties: 74.80
Observateur : Propriete valide

TP9 Description de l'observateur défini, sa traduction en automate d'états fini

On observe que dans le test correct_accepte, incorrect_accepte, correct_rejecte et incorect_rejecte, on obtient un résultat d'observation qui est valide après l'exécution.

Traduction sous forme d´automate avec une fonction de transition totale :



Conclusion :

- Test fontionnel et test robutesse valident les demandes de l'énoncé.
- L'algorithme pour generer aleatoirement des terrains reste quelque point faible.
- Programme infini_2 semble plus efficace que infini_5(pas de collision).
- Observateur joue le role observer le démarche des fonctions.