Projet d'introduction à la vérification

Adrien Mollet
Romain Soumard

19 mars 2020

1 Introduction

Le présent document est un rapport écrit à l'issue du projet de fin d'année d'introduction à la vérification. Il a été rédigé par M.Adrien Mollet et M.Romain Soumard à l'université d'Aix-Marseille durant l'année scolaire 2019-2020. Le but de ce projet consistait en la résolution de deux exercices :

- Exercice 1 : Trouver la plus courte solution au problème du berger.
- Exercice 2 : Modélisation du jeu du solitaire à l'aide du langage Promela.

Les sections suivantes présenteront brièvement l'organisation du travail et la résolution de chaque exercice.

2 Organisation du travail

Le travail a été séparé en deux groupes. M.Soumard s'est chargé de l'exercice 1 tandis que M.Mollet s'est chargé de l'exercice 2. Le travail s'est fait via github. Une fois les exercices terminés, une dernière sessions s'est faite afin d'harmoniser le rendu, nettoyer le code, et rajouter des script bash faisant office de makefile afin de faciliter la correction.

3 Exercice 1 : Problème du berger

3.1 Résolution du problème

La résolution du problème du berger s'est faite en deux temps. Il a tout d'abord fallu trouver les caractéristiques de l'exécution que l'on désirait obtenir. Une fois ceci fait, il a fallut l'exprimer à l'aide d'une formule LTL.
Bien que dans un premier temps, nous exprimions la propriété à vérifier à l'aide d'un processus never, il nous est apparu en parcourant la documentation de spin qu'il était bien plus aisé d'exprimer directement la propriété recherchée dans le code. Il s'agit en effet de la méthode recommandée depuis spin 6, celle-ci étant plus propre et équivalente (la formule reste la même, elle n'est juste pas automatiquement nier quand on génère un processus never).
Après avoir trouver la formule appropriée, une longue phase de débugguage s'en est suivi pour comprendre pourquoi nous n'arrivions pas à générer le modèle approprié. Il s'est avéré que cela était dû à l'ordre dans lequel nous modélisions les traversées des différents protagonistes.
En effet, pour vérifier la propriété voulu, il était essentiel que le berger arrive avant ses amis sur la rive opposée dans notre modèle.
Une fois terminée, nous nous sommes aperçus que l'exécution engendrée par notre propriété produisait déjà une solution optimale (facile à vérifier à l'aide d'un graphe.). Nous avons donc généré un graphe de séquence de message (MSC) afin de compléter l'exercice.

3.2 MSC du berger

Vous trouverez ci-joint dans le fichier MSC_berger.txt le MSC du problème du berger.

4 Exercice 2 : Modélisation du solitaire