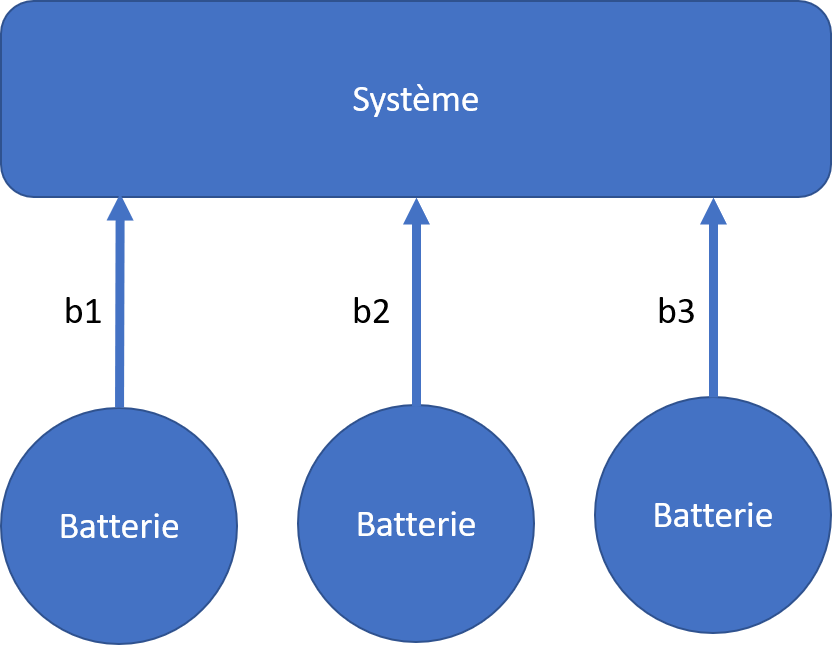
Question 3.1

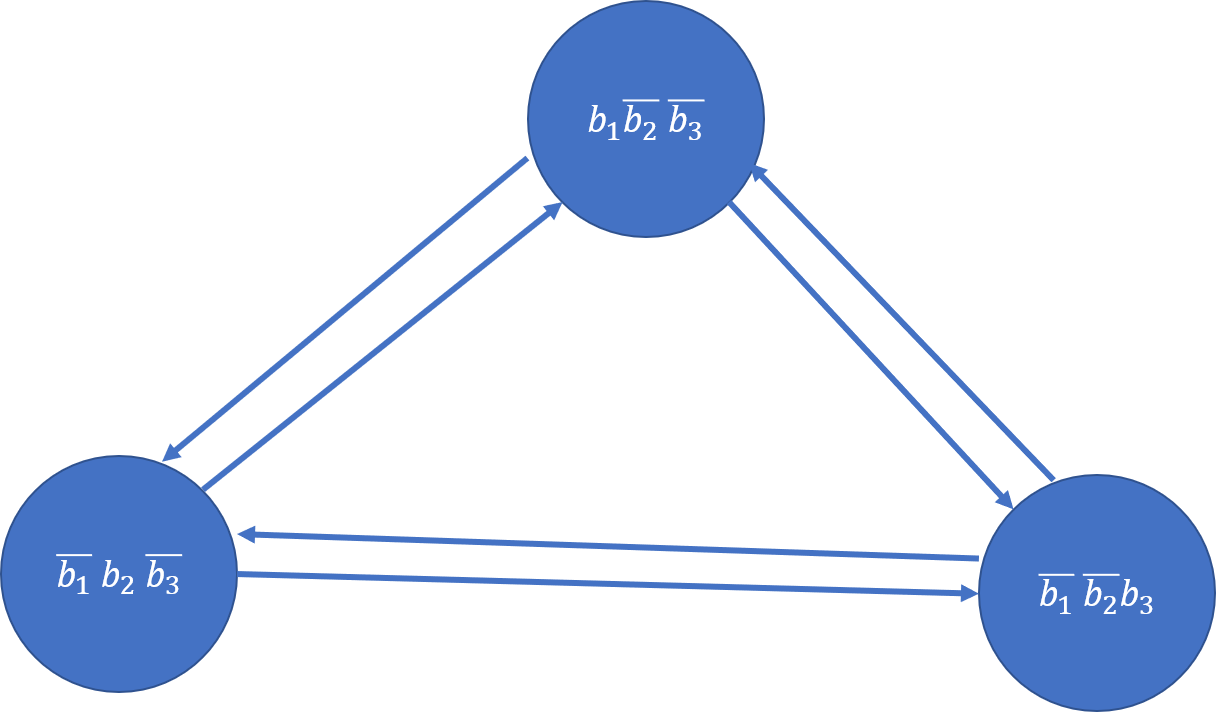


Question 3.2

- Propriété "pas de court-circuit" : ­­

- Propriété "continuité de l'alimentation" :

- Propriété "Changement de batterie d'un état à l'état suivant" :

Question 3.3 

Question 7.1

Les N processus commencent avec chacun une valeur de drapeau quelconque entre 0 et K. Puis à chaque transition, les processus mettent leur drapeau à jour, selon les règles énoncées : le processus 0 incrémente modulo K son drapeau si le processus N-1 a un drapeau différent du sien. Les autres processus, prennent la valeur de drapeau de leur voisin de droite respectif si cette valeur est différente de la valeur actuelle de leur propre drapeau.

Mathématiquement, on note l’état où les processus 0 à N-1 ont respectivement les valeurs entre 0 et K-1. Dans l’état suivant, on aura alors les valeurs avec :

* Si alors , sinon
* Pour tout , si , alors , sinon

Par exemple, pour , voici deux marches partielles possibles du système avec deux états initiaux différents :

On entrevoit ici le caractère auto-stabilisateur de l’algorithme puisque dans les deux cas, au bout d’un moment, seul un processus change de valeur à chaque transition.

Question 7.2

Voici quelques propriétés attendues du système :

* Le processus 1 prendra la valeur 0 infiniment souvent 🡪 GF(p1=0). Aussi vrai pour tous les autres processus et tous les entiers inférieurs strictement à K.
* Au bout d’un certain temps, le processus 0 prendra la valeur 0 et la gardera jusqu’à ce que tous les autres processus aient aussi la valeur 0 🡪 F((p0=0)U(p1=p2=…=pN-1)). Vrai avec toutes les autres valeurs que 0.
* Au bout d’un certain moment, on aura toujours, pour tous , 🡪 AFG(p0 = p1 v p0 = p1 + 1 mod K) (exemple avec i = 1).
* Au bout d’un cer

Question 7.4

L’algorithme impose que car sinon la propriété d’auto-stabilisation est trivialement violée. Par exemple, avec et l’état initial on a :

(boucle infinie)

Question 9.1

Approche naïve : On peut modéliser le rubik’s cube en utilisant 24 variables, ce qui correspond au nombre de petits carrés (4 sur chacune des 6 faces). Les couleurs sont modélisées par des entiers entre 0 et 5 inclus. Les transitions sont les actions possibles sur le rubik’s cube, c’est-à-dire le fait de pivoter une face. Afin de restreindre le nombre de transitions possibles, on n’autorisera le pivot que dans un sens. On ne perd pas de généralité car par exemple, le fait de tourner une face dans le sens antihoraire revient à tourner trois fois de suite cette même face dans le sens horaire. Cela donne 6 mouvements possibles, car il y a 6 faces. Afin d’obtenir une résolution du rubik’s cube, on demandera à NuSMV de vérifier la propriété qui dit que tous les carrés d’une même face doivent être différents. Cette propriété doit être fausse, et le contre-exemple fourni indique comment résoudre le problème.

Approche optimisée : L’approche précédente prend beaucoup trop de temps de calcul, car il y a trop d’états (car les symétries ne sont pas exploitées). Une approche plus optimisée est de modéliser le problème par les sommets et non les petits carrés. En effet, chaque sommet a un triplet de couleurs qui lui est propre, et dans toutes les solutions, modulo les symétries, les chaque sommet est toujours à la même place. On a donc 8 variables correspondant aux 8 sommets. Les transitions sont les mêmes que précédemment. La propriété de logique temporelle exprime le fait que chaque sommet est à sa place.

Afin de coordonner les mouvements, on ajoute une variable move, qui permet de ne pas tout mélanger. Cette variable prend les valeurs de 1 à 6 inclus. 1 signifie qu’on tourne la partie gauche du rubik’s cube vers le haut, 2 signifie qu’on tourne la partie haute vers la droite, 3 signifie qu’on tourne la partie avant dans le sens horaire, 4 signifie qu’on tourne la partie gauche vers le haut, 5 signifie qu’on tourne la partie basse vers la droite et 6 la partie arrière dans le sens horaire.

Question 9.2

La version optimisée donne une solution dans un temps raisonnable. Dans la section ASSIGN, on peut donner la configuration initiale du rubik’s cube. L’une d’entre elles est donnée par défaut, mais on peut modifier pour tester plusieurs configurations.