Introduction a la vérification formelle 2019-2020
TP de Model-checking

Consignes générales :

Le TP est constitué de plusieurs exercices indépendants (plus ou moins, en ordre de difficulté croissante). Il est attendu que le TP comporte :

- un document texte (type word, ou latex) incorporant les réponses ou commentaires aux questions ;
- un répertoire compressé des modèles SMV réalisés (avec des noms de fichiers explicites).

  Comme pour tout travail d'implémentation, il est fortement conseillé d'insérer des commentaires dans les modèles (les commentaires sont introduits par un double tiret -).

L'essentiel du travail demandé est un travail de modélisation : il y a donc en général plusieurs options possibles. Lorsque des choix de modélisation (avec leurs avantages et limitations éventuels), veuillez les argumenter. N'hésitez pas à opter pour des modèles simples.

Date limite de rendu de projet : 8 mars. Le TP peut se faire seul ou en binôme.

Les exercices à rendre sont les exercices 3 à 7 (l'exercice 7 étant optionnel) : les exercices 1 et 2 ne sont que des exercices de prise en main.

Exercice 1 : Prise en main

Considérons l'exemple SMV suivant (disponible dans le fichier exemple.smv) :

```
MODULE main
VAR
 request : boolean ;
 state : {ready, busy};
ASSIGN
 init(state) := ready;
 next(state) := case
  state = ready & request : busy;
  TRUE : {ready,busy};
 esac;

SPEC AG(request -> AF state = busy)
```

1. Dessiner l'automate décrit par le modèle SMV et traduire en langage naturel la propriété de logique temporelle donnée dans le modèle SMV.

2. Ouvrir le fichier `exemple.smv` en mode interactif (commande `NuSMV -int exemple.smv` en ligne de commande dans le répertoire contenant le fichier) et se référer au chapitre 3 du tutoriel Nusmv pour une première prise en main en mode interactif.

```
tintin-4:CodeSMV pascalelegall$ nusmv -int exemple.smv
*** This is NuSMV 2.6.0 (compiled on Wed Oct 14 15:31:33 2015)
*** Enabled addons
...
WARNING *** Please contact Sharad Malik (malik@ee.princeton.edu)      ***
WARNING *** for details.                                              ***

NuSMV >
```

3. Utiliser NuSMV pour vérifier la formule CTL présente dans le fichier (commande `NuSMV exemple.smv` en ligne de commande dans le répertoire contenant le fichier)

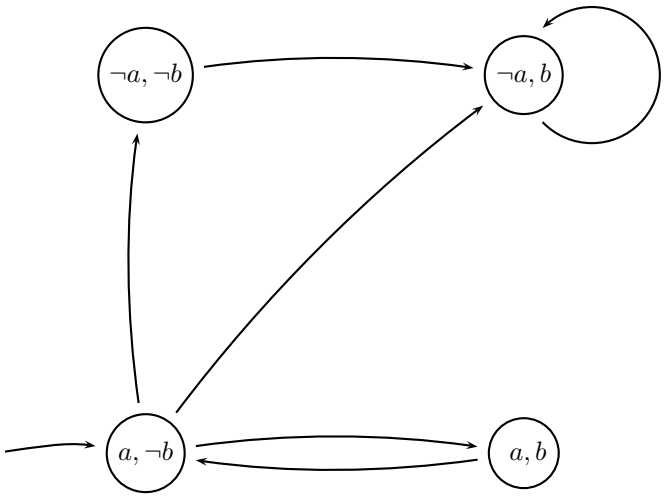
```
tintin-4:CodeSMV pascalelegall$ nusmv exemple.smv
*** This is NuSMV 2.6.0 (compiled on Wed Oct 14 15:31:33 2015)
*** Enabled addons are: compass
*** For more information on NuSMV see <http://nusmv.fbk.eu>
...
WARNING *** Please contact Sharad Malik (malik@ee.princeton.edu)      ***
WARNING *** for details.                                              ***

-- specification AG (request -> AF state = busy)  is true
tintin-4:CodeSMV pascalelegall$
```

   Proposer d'autres formules CTL et LTL vraies et fausses pour le modèle exemple.smv

---

Exercice 2 : Un premier modèle simple

Pour le modèle ci-dessous avec a et b variables booléennes (une flèche entrante signalant un état initial) :



1. Ecrire trois modèles NuSMV le modélisant
   (a) le premier à l'aide de deux variables booléennes, via la construction `ASSIGN` ;
   (b) le second à l'aide d'une variable définie à l'aide d'un type énuméré de 4 valeurs et d'un champ DEFINE pour la définition des variables booléennes a et b, via la construction `ASSIGN` ;
   (c) le troisième en définissant la relation de transitions entre états à l'aide de formules logiques, via les constructions `INIT` et `TRANS`.
2. Pour chacun des modèles proposés et pour chacune des formules ci-dessous :
   (a) $\mathbf{G}\, a$
   (b) $a\, \mathbf{U}\, b$
   (c) $a\, \mathbf{U}\ \mathbf{X}\, (a \wedge \neg b)$
   (d) $\mathbf{X}\, \neg b \wedge\ \mathbf{G}\, (\neg a \vee \neg b)$
   (e) $\mathbf{X}\, (a \wedge b) \wedge\ \mathbf{F}\, (\neg a \wedge \neg b)$
   (f) $\mathbf{EF}\ \mathbf{EG}\, a$
   (g) $\mathbf{EF}\ \mathbf{EG}\, b$
   (h) $\mathbf{F}\, ((\ \mathbf{G}\, a) \vee (\ \mathbf{G}\, b))$

   Utiliser le model checker NuSMV pour indiquer si la formule est valide pour le modèle, en donnant pour chaque formules non valide, une trace qui l'invalide.

   Prenez soin de vous assurer que les réponses fournies sont conformes à vos attentes (étape de validation de conception de modèles et de formules)

Exercice 3 : Expression de propriétés

Dans cet exercice, on considère des propriétés exprimées dans la logique LTL où les formules élémentaires seront de simples variables propositionnelles choisies dans l'ensemble $\{p, q, p_1, p_2, q_1, q_2\}$.

Exprimer en LTL les propriétés suivantes :

- Il y a toujours p.
- Il y a p une infinité de fois.
- Il n'y a jamais p et q simultanément.
- Après chaque occurrence de p il y a au moins une occurrence de q.
- S'il y a une infinité de p_1 et une infinité de p_2 alors toute occurrence de q_1 est suivie d'une occurrence de q_2.
- Avant chaque occurrence de p il y a au moins une occurrence de q.
- Entre chaque paire d'occurrence de p il y a au moins une occurrence de q.

Pour chaque formule LTL ci-dessous, donner un modèle NuSMV qui la satisfait.

Exercice 4 : Traversée de rivière

Modéliser en NuSMV le système suivant :

Un lombric de 50g, un millepatte de 30g et une sauterelle de 20g sont ensemble sur une même berge d'une rivière, et souhaitent traverser la rivière. Pour cela, ils disposent d'une feuille d'arbre qui peut porter au maximun 60g et qui ne va d'un bord à l'autre de la rivière qu'en portant un insecte sur son dos.

Utiliser NuSMV pour savoir s'il est possible aux trois insectes de traverser la rivière.

Exercice 5 : Algorithme auto-stabilisateur de Dijkstra (74)

Considérons un problème d'exclusion mutuelle pour N processus, P_0, P_1, $\ldots P_{N-1}$ disposés sur un anneau. Chacun des processus

- possède une variable `drapeau`, qu'il peut lire et écrire,
- et peut lire la variable `drapeau` de son voision de droite (P_{N-1} pour le processus P_0, P_{i-1} pour le processus P_i pour $i \in 1..N$)

avec `drapeau` $\in 0..(K-1)$ avec $K > N$.

Le protocole distingue le processus P_0 des autres processus :

- le processus P_0 peut entrer en section critique si son drapeau est égal au drapeau de P_{N-1}. Dans ce cas, le drapeau de P_O est mis à `(drapeau + 1) mod K` ;
- le processus P_i (avec $i \neq 0$) peut entrer en section critique si son drapeau est différent du drapeau de P_{i-1}. Dans ce cas, le drapeau de P_i est mis au drapeau de P_{i-1}.

Cet algorithme est dit *auto-stabilisant* car au bout d'un certain temps, quelles que soient les conditions initiales, le système des N processus vérifie que chacun des processus entre en section critique (i.e. est le seul en capacité de modifier son drapeau) infiniment souvent.

1. Expliquer le fonctionemment du protocole (en langage naturel, à l'aide de raisonnement mathématique, à base d'un exemple) ;
2. Donner des propriétés attendues du protocole en des phrases susceptibles d'être traduites en logique temporelle ;
3. Proposer un modèle SMV de ce protocole avec les propriétés en logique temporelle que vous jugerez appropriées. Commenter.

   (il conviendra de choisir des valeurs de N et K petites mais représentatives du problème)
4. L'algorithme de Dijkstra requiert que $K > N$. Pourquoi ?

Exercice 6 : Election distribuée d'un leader

L'algorithme distribué de Dolev, Klawe et Rodeh (1982), noté DKR, met en œuvre l'élection d'un leader dans un anneau unidirectionnel. Chaque processus (ou nœud) de l'anneau est repéré par son identifiant, sous la forme d'un entier (tous les identifiants sont distincts deux à deux).

L'objectif est d'élire un unique processus comme leader à l'aide d'un algorithme distribué, i.e. sous l'hypothèse que chaque processus exécute le même algorithme. L'algorithme DKR se décrit à l'aide des étapes suivantes :

- Chaque processus (ou nœud) fonctionne selon 2 modes différents : tant qu'il espère encore être élu leader, un processus est en mode actif.

  Dès qu'il considère qu'il ne sera pas élu, il devient passif. Dans ce mode, il se contente de retransmettre au nœud suivant les messages reçus sans les modifier. S'il reçoit l'information d'une élection (choix d'un processus), il en prend aussi note.
- Tous les processus sont initialement actifs.
- Chaque processus contient une variable, appelée leader, qui est initialisée avec son propre identifiant.

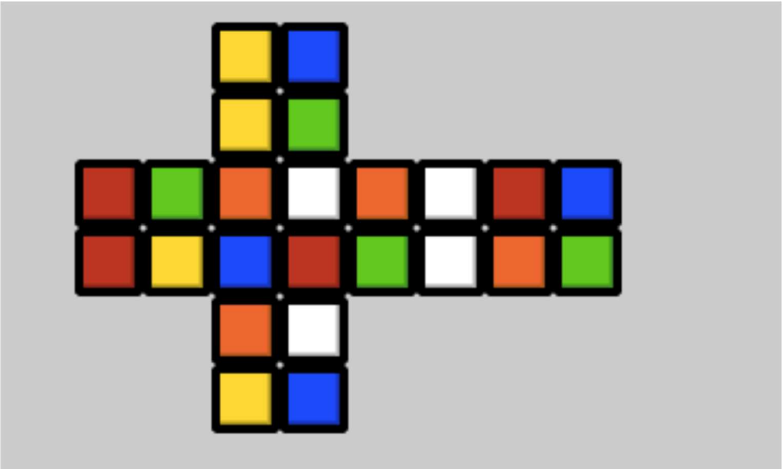


Figure 1: Exemple (extrait de https://ruwix.com/) du rubik'cube mélangé, vu en format déployé

- (Etape 0) Un processus commence par transmettre la valeur `val` stockée dans sa variable leader à son voisin direct dans l'anneau, et obtient une valeur (`val1`) transmise par son précédent voisin.
  - Si ces valeurs sont égales, alors il s'agit de l'identifiant leader (la valeur a fait le tour de l'anneau en étant préservée), le processus envoie alors cette valeur au nœud suivant en indiquant qu'il s'agit de l'identifiant leader.
  - Sinon, il transmet la valeur `val1` (qu'il vient de recevoir) au nœud suivant en indiquant qu'il s'agit de la valeur du nœud précédent. Il reçoit alors aussi de la part de son nœu prédécesseur la valeur du nœud deux fois prédécesseurs (`val2`) qui a été relayée par le nœud directement prédécesseur.

    Si `val1` est minimal parmi `val`, `val1`, `val2`, la valeur `val1` devient la valeur de sa variable leader, et le processus reste actif (et reprend donc à l'étape 0).

    Dans le cas contraire (`val1` n'est pas minimal parmi `val`, `val1`, `val2`), le processus devient inactif.

Modélisez ce protocole avec NuSMV (à titre indicatif, l'anneau contient 6 processus) et vérifiez que deux processus ne peuvent être élus conjointement leaders et qu'il existe au moins une exécution permettant l'élection d'un leader.

Le fichier leader.smv est un fichier à compléter pour un anneau unidirectionnel à 6 processus

---

Exercice 7 : Rubik's cube

Dans cet exercice, il s'agit de résoudre les rubik's cube de dimension 2x2 à l'aide de NUSMV. Autrement, à partir d'un rubik's cube mélangé (tel que celui décrit par la figure 1), il faut fournir la liste des mouvements qui permette de réarranger le rubik's cube (tel que sur la figure 2) en assurant que toutes les faces sont monocolores.

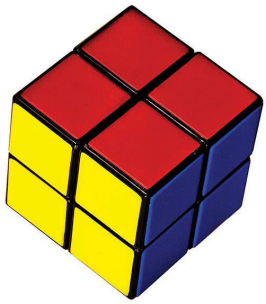


Figure 2: Exemple du rubik'cube rangé, vu en format 3D

1. Proposez une façon de modéliser le rubik's et ses mouvements sous forme d'un modèle SMV.
2. Utilisez NuSMV pour résoudre un rubik's cube[1].

---

[1]La résolution du problème nécessite d'avoir un modèle nusmv assez optimal, *i.e* minimisant la taille du modèle en nombre d'états et de transitions.