Introduction a la vérification formelle 2017-2018
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 vous rendiez le TP par email à pascale.legall@centralesupelec.fr avec deux pièces jointes :

- 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 : 30 janvier. Le TP peut se faire seul ou en binôme.

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)

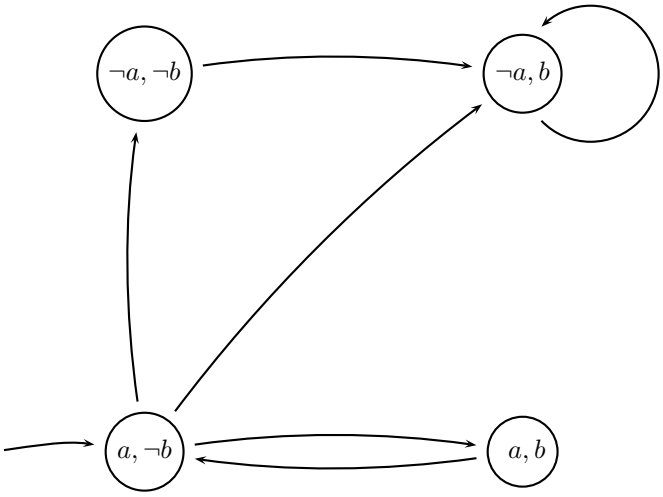
```
tintin-4:CodeSMV pascalelegall$ nusmv exemple.smv
*** This is NuSMV 2.6.0 (compiled on Wed Oct 14 15:31:33 2015)
*** Enabled addons are: compass
*** For more information on NuSMV see <http://nusmv.fbk.eu>
...
WARNING *** Please contact Sharad Malik (malik@ee.princeton.edu)      ***
WARNING *** for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

Il s'agit de donner un modèle NuSMV (système de transitions muni de propriétés temporelles) qui rende compte de l'exercice.

Soit un système alimenté par 3 batteries. Entre chaque batterie et le système se trouve un interrupteur. Un programme *commutateur* permet de jouer sur ces 3 interrupteurs à intervalles réguliers pour commuter ou non les batteries et éviter qu'une même batterie ne débite trop longtemps mais aussi pour éviter les surcharges si plusieurs batteries débitaient en même temps (court-circuit).

Question 1. Faire un dessin du système.

Question 2. en vous basant sur 3 propriétés booléennes (une par interrupteur), exprimez les propriétés suivantes : pas de court-circuit, continuité de l'alimentation, changement de batterie d'un état à l'état suivant.

Question 3. En étiquetant les états avec les propriétés (i.e. un état est défini par la valeur booléenne de chacun des interrupteurs), donner un système de transition du commutateur qui en respecte les propriétés.

Exercice 4 : Expression de propriétés (extrait du td Automate et logique temporelle)

Il s'agit de donner pour chaque propriété ci-dessous un modèle NuSMV qui la satisfait.

Dans cet exercice, on va exprimer des propriétés dans la logique LTL où les formules élémentaires seront de simples variables propositionnelles (i.e. des variables dont les valeurs sont vrai ou faux) choisies dans l'ensemble $\{p, q, p_1, p_2, q_1, q_2\}$. Ainsi pour exprimer la propriété en LTL "Un jour il y aura p", on écrit la formule Fp. 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.

Exercice 5 : 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 6 : Formules équivalentes

Parmi les couples de formules CTL qui suivent, lesquelles sont équivalentes (i.e. sont validées exactement par les mêmes modèles) ?

1. $\mathbf{E}\,\mathbf{F}\,\varphi$ et $\mathbf{E}\,\mathbf{G}\,\varphi$
2. $\mathbf{E}\,\mathbf{F}\,\varphi \vee \mathbf{E}\,\mathbf{F}\,\psi$ et $\mathbf{E}\,\mathbf{G}\,(\varphi \vee \psi)$
3. $\mathbf{A}\,\mathbf{F}\,\varphi \vee \mathbf{A}\,\mathbf{F}\,\psi$ et $\mathbf{A}\,\mathbf{F}\,(\varphi \vee \psi)$
4. $\mathbf{A}\,\mathbf{F}\,\neg\psi$ et $\neg\,\mathbf{E}\,\mathbf{G}\,\psi$
5. $\mathbf{E}\,\mathbf{F}\,\neg\psi$ et $\neg\,\mathbf{A}\,\mathbf{F}\,\psi$
6. $\mathbf{A}[\psi\,\mathbf{U}\,\mathbf{A}[\varphi\,\mathbf{U}\,\rho]$ et $\mathbf{A}[\,\mathbf{A}[\psi\,\mathbf{U}\,\varphi]\,\mathbf{U}\,\rho]$
7. $\top$ et $\mathbf{A}\,\mathbf{G}\,\varphi \Rightarrow \mathbf{E}\,\mathbf{G}\,\varphi$
8. $\top$ et $\mathbf{E}\,\mathbf{G}\,\varphi \Rightarrow \mathbf{A}\,\mathbf{G}\,\varphi$

Selon le cas, utiliser NuSMV pour attester de l'équivalence ou pour exhiber une trace qui invalide l'équivalence.

Exercice 7 : 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 8 : Ascenseur à 3 étages

Dans cet exercice, il s'agit de modéliser un ascenceur à 3 étages :

Appel de l'ascenseur (à chaque étage) : à chaque étage, il y a un (ou plusieurs) bouton(s) d'appel pour demander l'ascenseur ;

Demande d'étage (dans l'ascenseur) : dans l'ascenseur, il y a un (ou plusieurs) bouton(s) pour indiquer l'étage demandé.

1. Proposer
   (a) un premier modèle `ascenseur_mod1.smv` qui donne priorité à la demande de la personne présente dans l'ascenseur. Autrement dit, la demande d'étage à l'intérieur de l'ascenseur, si elle existe, est prioritaire à toute appel de l'ascenceur ;

(b) un second modèle `ascenseur_mod2.smv` qui privilégie la direction courante. Autrement dit, lorsque l'ascenseur commence à monter (resp. descendre), il continue à monter (resp. descendre) s'il y a un appel à servir.

Pour chacun des deux modèles, vérifier si tout appel de l'ascenseur est nécessairement satisfait (propriété de logique temporelle que l'on notera par suite φ).

Nota bene : la description est volontairement elliptique, ce qui laisse de nombreuses options pour la modélisation.

2. Les fournisseurs de systèmes peuvent offrir des options (ou *services*) comme :

Etage de direction (ED)

Ce service permet de servir en priorité l'étage de direction sur tous les autres étages ;

Parcage par défaut (PD)

Lorsque l'ascenseur est libre (i.e. sans appel en cours), il va par défaut se positionner à l'étage le plus bas ;

Indiquer quelles sont les propriétés attendues des options ED et PD, en langage naturel et en logique temporelle.

Modifier votre modèle `ascenceur_mod2.smv` pour intégrer séparément le service ED (modèle `ascenceur_ED.smv`) et le service PD (modèle `ascenceur_PD.smv`).

Vos nouveaux modèles satisfont-ils la propriété φ (éventuellement légèrement adaptée en fonction des modifications apportées par vos modèles) ? Cela est-il conforme avec votre intuition ?

Exercice 9 : 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.

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.

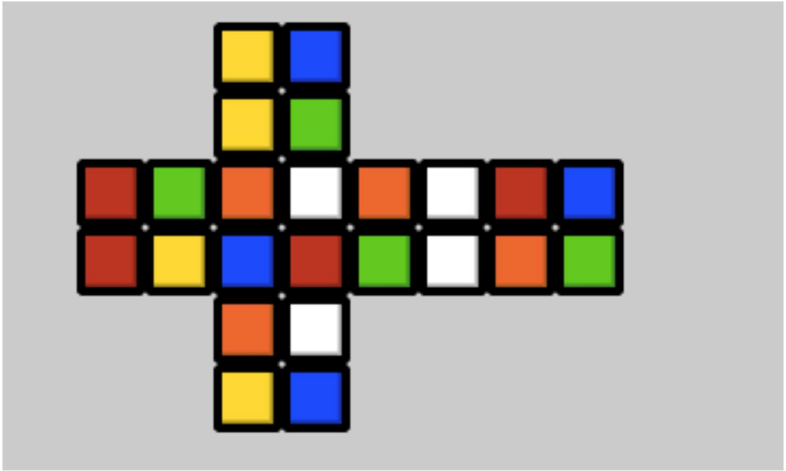


Figure 1: Exemple (extrait de https://ruwix.com/) du rubik'cube mélangé, vu en format déployé

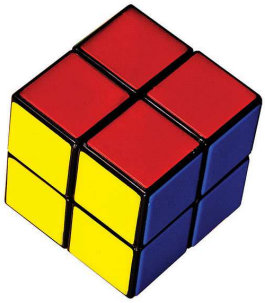


Figure 2: Exemple du rubik'cube rangé, vu en format 3D