TD – Vérification des propriétés LTL et CTL en NuSMV

C. Dima

Exercice 1: (Reprise d'un exercice du module d'*Initiation à la sécurité*) On considère un système de contrôle d'accès dans lequel on a trois utilisateurs, $U = \{a, b, c\}$ et trois fichiers, $F = \{x, y, z\}$. Les actions possibles sont la lecture et l'écriture d'un fichier par un utilisateur, plus une action de gestion de système par laquelle un utilisateur privilégié donne ou retire des droits (de lecture ou écriture). L'action de lire, d'écrire ou de donner un droit est signalée par l'état de l'utilisateur. Une seule action peut avoir lieu à la fois (on est dans un système monoprocesseur). On veut imposer une politique de sécurité dans laquelle les propriétés suivantes ont lieu :

1. a pourrait lire le fichier z seulement s'il n'a pas lu x ou y après que b aurait écrit là-dedans.
2. c, qui est le seul utilisateur privilégié, peut à tout moment donner ou retirer le droit d'écrire dans n'importe quel fichier à n'importe quel utilisateur. Ce droit d'écrire est toutefois inhibé pour a lorsque la première propriété a lieu.
3. Chaque action (lecture ou écriture) peut être lancée en exécution par n'importe quel utilisateur à n'importe quel instant, mais, en fonction des droits dans la matrice de contrôle d'accès, elle peut aboutir ou non.
4. Lorsque a écrit dans le fichier z, la propriété 1 est "resetée" (c'est à dire, les accès précédents en écriture de b ne comptent plus).

Modéliser le système de contrôle d'accès en NuSMV, spécifier les propriétés en LTL ou CTL et vérifier votre système pour chacune des propriétés. Au cas où une des propriétés ne sera pas satisfaite, modifier votre système en conséquence, en utilisant l'éventuelle trace affichée par NuSVM.

Exercice 2: (Pout cet exo il est utile d'ouvrir le *User manual* de NuSMV) NuSVM crée la représentation en BDD du système de transition en *aplatissant* celui-ci, c.à.d. en générant des variables booléennes pour chacune des modules/variables spécifiées, et pour chaque bit dans la représentation binaire de celles-ci, puis en construisant le BDD de la relation de transition sur la base de l'ordre de déclaration des variables.

Vous avez la possibilité de spécifier un autre ordre, sous la forme d'une liste des variables (qualifiées en fonction de leur appartenance aux différents modules), liste que vous pouvez mettre dans un fichier. Les commandes suivantes sont utiles dans ce but :

- `write_order -o nomfichier` : écrire dans le fichier `nomfichier` l'ordre utilisé sur les variables pour construire le BDD.
- `set input_order_file nomfichier` : l'ordre à considérer pour la prochaine construction d'un modèle sera celui donné dans `nomfichier` (à utiliser avant la commande `build_model` ou `go`, mais après `flatten_hierarchy`).
  Nota : `encode_variables -i nomfichierordre` est une combinaision de `encode_variables` et `set input_order_file`.

D'autre part, la commande `print_usage` donne un aperçu des ressources utilisés dans la dernière vérification de propriété faite (nbre de noeuds BDD, nbre de noeuds initiaux, etc.).

Pour le modèle et les propriétés du 1er exo, proposer d'auters ordres de variables (en les mettant dans des fichiers distincts), lancer la vérification des différentes propriétés CTL ou LTL, puis afficher les statistiques et comparer la taille des représentations BDD, ou, si c'est relevant, le temps d'exécution pour chacun des ordres proposés.

Attention, l'effet d'une commande `build_model` (qui construit le BDD correspondant à un ordre) ne peut être annulé que par une commande `reset` !

Exercice 3: Considérons un ascenseur pour un bâtiment à 4 étages + RdC (pour transport de marchandises), pour lequel chaque étage possède une porte d'ascenseur, un voyant indiquant si l'ascenseur a été appelé de l'étage respectif, plus un bouton d'appel. Il n'y a pas de bouton d'appel dans la cage de

l’ascenseur. À part la commande en cours de traitement, on peut mémoriser une 2e commande. Proposer un nombre minimal de propositions atomiques et des formules LTL ou CTL permettant de spécifier les propriétés suivantes :

1. L’ascenseur se met en marche seulement lorsqu’il y a une demande.
2. L’ascenseur se déplace d’étage en étage.
3. À n’importe quel instant, il est possible de ramener l’ascenseur au rez-de-chaussée.
4. Après que la cage de l’ascenseur arrive à destination, la commade en attente est mise à exécution en enlevée de la mémoire. Une autre commande peut, à ce moment, être prise en compte et mise dans la mémoire.
5. L’ascenseur ne peut pas se déplacer tant que toutes les portes soient fermées.
6. Toute commande prise en compte sera éventuellement satisfaite.
7. L’ascenseur peut être mis à l’arrêt (par exemple pour des réparations).
8. Lorsque cela arrive, aucune commande n’est prise en compte et la mémoire est vidée.
9. À la remise en marche, l’ascenseur est déplacé au RdC.

Rajouter d’autres propriétés fonctionnelles ou non-fonctionnelles du système et les traduire en CTL ou en LTL.
