Sujets de projet de Langages de Spécification

Catalin Dima

Cadre général

- Chaque sujet demande la modélisation en NuSMV d'une spécification décrite en français, plus la vérification de trois formules LTL, rédigées d'abord en français.
- Les rapports (partie NuSMV) sont à rendre le 11 janvier, par mail, à `dima@u-pec.fr`.
- Chaque rapport doit comprendre :
  - Le fichier `.nusmv` contenant le modèle NuSMV.
  - Un rapport décrivant le modèle : la correspondance entre variables et éléments de la spécification, les choix faits pour l'implémentation des fonctionnalités, les formules vérifiées et les résultats des vérifications, etc.

Producteurs/consommateurs multiples

- Deux processus produisent des données (entières) qu'ils placent sur un tapis roulant qui les relient à deux autres processus.
- La production de chaque donnée peut prendre entre 3 et 6 unités de temps.
  - Production = affectation aléatoire d'une valeur.
- Les deux processus doivent accéder au tapis roulant en *exclusion mutuelle* (formule LTL à vérifier !).
- Deux autres processus consomment les données, dans l'ordre de leur production.
- La consommation de chaque donnée prend elle aussi entre 3 et 6 unités de temps, et l'accès de chaque consommateur au tapis roulant se fait en exclusion mutuelle aussi, et en exclution mutuelle avec tous les producteurs.
- Le tapis roulant a un nombre de cases maximal, si un producteur essaie de placer une donnée sur un tapis plein, il sera bloqué jusqu'à la libération d'une case.
- Un consommateur qui essaie de récupérer une donnée sur un tapis vide sera lui aussi bloqué jusqu'à ce qu'un producteur y place une donnée (formule LTL à vérifier).
- Une troisième formule peut être proposée par vous-mêmes.

TCP simplifié

- Un processus client veut envoyér une donnée a un processus serveur et veut avoir un accusé de réception pour chaque donnée envoyée.
- Les données sont envoyées sur des "canaux" sur lesquels la durée de livraison de chaque paquet prend entre 1 et 10 secondes. Les paquets peuvent aussi se perdre.
- Le client envoie un paquet comprenant son identité, l'identité du serveur, un numéro de séquence (entier qui est incrémenté à chaque livraison avec succès) et la donnée envoyée (paquet SYN).
- Lorsque le serveur reçoit un paquet SYN, il renvoie un paquet SYN-ACK comprenant les adresses source et destination du paquet SYN, (mais inversées), et le numéro de séquence reçu.
- Lorsque le client reçoit un paquet SYN-ACK, il vérifie qu'il correspond bien avec le dernier paquet envoyé, et répond par un paquet ACK, qui contient à nouveau les deux adresses, le no. de séquence et la valeur 0.
- Chacun des processus (client ou serveur), attend au maximum 10s la réception de chaque paquet, sinon il oublie ce qu'il avait à faire.
- Le serveur doit pouvoir traiter 2 connexions à la fois (donc a une mémoire de taille 2 pour attendre les paquets ACK).
- Modéliser un système avec deux clients et un serveur.
- Formules à vérifier :
  - Chaque donnée (identifiable à l'aide de son no. de séquence) envoyée par un processus sera reçue par le serveur.
  - Aucun processus ne attend à l'infini l'arrivée d'une donnée.
  - Troisième formule de votre choix.

Ascenseur

Modéliser un ascenseur dans un bâtiment à 3 étages, ayant les fonctionnalités suivantes :

- Chaque étage est doté d'un bouton d'appel, d'un voyant indiquant l'état de l'ascenseur (marche/arrêt) et d'une porte (ouverte/fermée).
- L'ascenseur lui-même est doté de boutons d'appel pour chaque étage et d'une porte.
- L'ascenseur monte ou descend d'étage en étage (sans saut !) – formule LTL à vérifier !
- L'ascenseur a une mémoire de deux appels.
- Chaque appel est d'abord stocké en mémoire, puis traité, mais seulement lorsque l'ascenseur est à l'arrêt.
- Lorsque l'ascenseur est à l'arrêt et la mémoire n'est pas vide, il retire l'appel le plus ancien de la mémoire et le traite.
- Traiter un appel revient à fermer les portes et à monter/descendre vers l'étage désiré (sauf lorsqu'il s'agit d'un appel vers le même étage !).
- Les boutons peuvent être appuyés à n'importe quel instant (formule LTL à vérifier !).
- Troisième formule LTL de votre choix à vérifier.

Communication sur bus avec détection de collision

- Trois clients se partagent un seul canal de communication partagé.
- La transmission d'une donnée nécessite d'indiquer l'expéditeur et le destinataire.
- Toute transmission commence en actualisant un bit "désir-de-communication" propre à chaque client, en déhors du placement de la donnée et de l'identité à envoyer dans deux registres du canal.
- Le canal (vu comme un processus à part) calcule la somme des bits de désir et, seulement lorsque le résultat est 1, il met à 1 une variable signalant la présence d'une donnée (et une identité) sur le canal. Sinon, cette variable prend une valeur négative, signalant une collision.
- Chaque client vérifie, à chaque instant, s'il y a une donnée disponible sur le canal et si la donnée lui est destinée, et si oui, il la récupère et la met dans une variable locale.
- Le comportement des clients est aléatoire lorsqu'ils ne reçoivent pas de donnée : ils peuvent choisir de produire eux-mêmes une donnée, ou de ne rien faire.
- Formules LTL à vérifier :
  - Chaque donnée envoyée est bien reçue par le destinataire.
  - Absence de famine : aucun client ne se trouve en situation de désirer une infinité de fois de communiquer, mais de pouvoir le faire seulement un nombre fini de fois.
  - Une troisième formule LTL de votre choix.

FTP simplifié

- Un serveur possède des registres où on peut stocker des données, plus une base de données mettant en correspondance des noms de clients et des mots de passe.
- Un client qui demande de récupérer une donnée envoie son identité et son mot de passe au serveur, et aussi le nom du registre dont il veut récupérer la valeur.
- Le serveur vérifie que le client a bien envoyé le mot de passe correspondant, et, si la correspondance est correcte, il envoie la valeur désirée, sinon il envoie une valeur "erreur".
- Les clients (en nombre de 3) ont accès chacun à un canal qu'ils partagent avec le serveur.
- Le serveur ne peut traiter qu'une requête à la fois. Si plusieurs requêtes arrivent en même temps, le serveur les traite dans un ordre prédéfini (à fixer par vous-mêmes).
- Une requête non-satisfaite est réitérée jusqu'à sa satisfaction (formule LTL à vérifier).
- Les clients peuvent aussi demander la sauvegarde d'une donnée sur un registre.
- Les registres peuvent être *vides* : le serveur fournira alors au client le demandant une valeur particulière signalant cette situation.
- Chaque donnée envoyée est bien reçue par le destinataire (formule LTL à vérifier).
- Une troisième formule de votre choix.

Ordonnanceur de type tourniquet (Round-Robin)

- Un ordonnanceur de type tourniquet *réveille* pendant un intervalle de temps t un processus dans un pool de 4 processus en exécution, sur une architecture monoprocesseur.
- Lorsqu'un processus est "réveillé" par l'ordonnanceur, il exécute une série de t opérations, à définir plus bas.
- Les opérations se font sur un seul et unique registre (celui du processeur).
- Le premier processus incrémente le registre de 2 unités à chaque instruction, le 2e processus décrémente le registre de 3 unités à chaque instruction, et le 3e processus multiplie par 2 le registre.
- Mais chaque processus agit ainsi pour modifier la valeur qui représente le résultat d'une précédente opération effectuée par le même processus ! Du coup, après chaque intervalle de temps, l'ordonnanceur sauvegarde la valeur du registre dans une variable propre au processus qui vient de terminer ses opérations.
- Les processus peuvent décider d'utiliser moins de temps que ce qui leur est accordé par l'ordonnanceur.
- Le 4e processus peut se réveiller tout seul, et lorsqu'il le fait, il devient *prioritaire* : le processus qui était en train de s'exécuter est interrompu, la valeur courante du registre est sauvegardée, et le 4e processus exécute pendant une durée de temps aléatoire ses opérations – tripler la (c.à.d. **sa**) valeur de régistre.
- Formules LTL à vérifier :
  - Exclusion mutuelle : un seul processus s'exécute à chaque instant.
  - À chaque pas d'exécution des processus 1, 2 ou 3, le registre est soit décrementé de 3 unités, soit incrémenté de 2 unités, soit multiplié par 2.
  - Une troisième formule de votre choix.