TD modélisation et expression d'exigences

INF722

Problème : Modélisation d'un four à micro-onde

Le but de cet exercice est de proposer un premier modèle qui servira à décrire la logique de fonctionnement d'un four à micro onde. Le comportement du four sera raffiné en seconde partie du sujet. Le but est de caractériser les séquences d'états possibles au niveau du fonctionnement du four.

Une première tâche consiste à interpréter une description textuelle du comportement du four, puis à la représenter sous forme d'une structure de Kripke (graphe dont les nœuds sont étiquetés avec des valuations, parfois partielles, des variables d'état du four).

Nous allons d'abord présenter le comportement global attendu du four en langage naturel. Le fonctionnement du four est décrit de manière simplifiée de sorte à ce que le premier modèle produit soit relativement simple (taille < 10 états distincts).

Les intéractions suivantes sont possibles avec le four : ouvrir / fermer la porte, et un bouton commade le démarrage du four. De son coté, le four peut démarrer ou arrêter son dispositif de chauffe, et peut signaler le fait que la porte est ouverte lorsqu'un démarrage du four est demandé.

Nous souhaitons que le four respecte les comportements suivants :

- Lorsqu'une commande de démarrage est passée, elle peut avoir les conséquences suivantes :
  - si la porte est fermée et le dispositif de chauffe éteint, le four peut démarrer le dispositif de chauffe et le démarrage est considéré comme réalisé.
  - si la porte du four est ouverte alors la commande est annulée et un message d'erreur est affiché (idem si le message d'erreur était déjà affiché).
  - si la porte est fermée et le dispositif de chauffe allumé, le dispositif de chauffe est éteint.
- Lorsqu'un message d'erreur est affiché, seule la fermeture de la porte peut le faire disparaître.
- La porte peut être ouverte à tout moment (pas de verrouillage). Le fait que la porte soit ouverte déclenche l'arrêt du dispositif de chauffe.

Question 1 :

Proposer un ensemble de variables booléennes (on ne recherche pas le plus petit ensemble mais l'ajout d'une variable doit pouvoir être justifié) et une structure de Kripke qui permette de formaliser le comportement décrit ci-dessus

(recommandation : précisez uniquement dans chaque état les variables vraies, les autres seront supposées fausses. C'est un cas dégradé de structure ou l'on associe une unique valuation à chaque état et non un ensemble de valuations).

Question 2 :

Méthode : pour capturer une contrainte sous forme LTL :

1- identifier les termes contraignant les variables

2- Identifier les différents instants contraints

- identifier les états liés (instantanément, et entre différents instants)

- identifier si le motif se répète et si oui sur quel intervalle de temps

Choisir les opérateurs temporels adéquats en fonction du schéma (un dessin peut aider).

Se rappeler que les opérateurs => et U représentent des contraintes différentes (l'une est instantanée, pas l'autre)

a) Donnez la formulation PLTL indiquant qu'il est toujours possible de revenir à l'état où le four est porte fermée, en attente d'une commande de démarrage (le four peut toujours est remis dans un état où le dispositif de chauffe puisse être démarré).

Donnez un argument pour justifier que cette propriété est vraie sur votre modèle (on ne vous demande pas de réinventer un algorithme de preuve mais de faire appel à un argument simple de bon sens sur le graphe de votre modèle).

b) Donnez l'équivalent en PLTL de la contrainte qui exige qu'à partir de tout état où une commande de démarrage est passée alors le four finit toujours par allumer le dispositif de chauffage

c) En quoi le non respect de cette exigence ne prouve pas grand-chose ? Illustrez avec un contre exemple (exécution satisfaisant la négation de la formule).

d) On propose de raffiner cette exigence en : pour tout état où une commande de démarrage est déclenchée alors que le four est éteint, si la porte est fermée et le reste, alors le four finit par se mettre à chauffer

e) Formuler une exigence indiquant qu'une commande de démarrage passée alors que la porte n'est fermée entraîne l'apparition d'un signal d'alarme qui restera actif tant que la porte n'est pas fermée.

f) Peut on formuler une expression LTL pour indiquer que si le bouton de démarrage est enclenché alors que la porte est ouverte alors cette commande est ignorée (le four ne s'allumera pas) (attention ici la formulation est floue doit être clarifiée avant de formaliser!!)

Question 3 :

Nous vous demandons de modéliser le comportement du four via un automate étendu qui devrait reproduire le comportement déjà décrit tout en capturant les choses suivantes

- On suppose que la commande de démarrage déclenche l'exécution d'un programme de cuisson (e.g. une durée durant laquelle le four chauffe).

- Un programme de cuisson dure trente secondes (durée que vous modéliserez par un entier)

- Lors de l'ouverture de la porte en cours de chauffe le programme n'est plus annulé mais simplement suspendu. A la fermeture de la porte, le décompte des secondes doit pouvoir reprendre jusqu'à terminaison du programme de cuisson. Le seul moyen d'interrompre le programme reste l'appui sur la commande de démarrage.

Vous utiliserez le concept de garde pour indiquer la condition qui détermine la fin de la cuisson et la condition de « poursuite de la cuisson »

Ajouter sur les arcs les étiquettes suivantes aux endroits opportuns : Presscd, open, close

Question 4 :

Exprimez sous forme d'une formule LTL le fait qu'une cuisson ne puisse pas durer strictement plus de 30 s (on supposera la mise à jour de la variable indiquant l'écoulement du temps correcte).

Correction

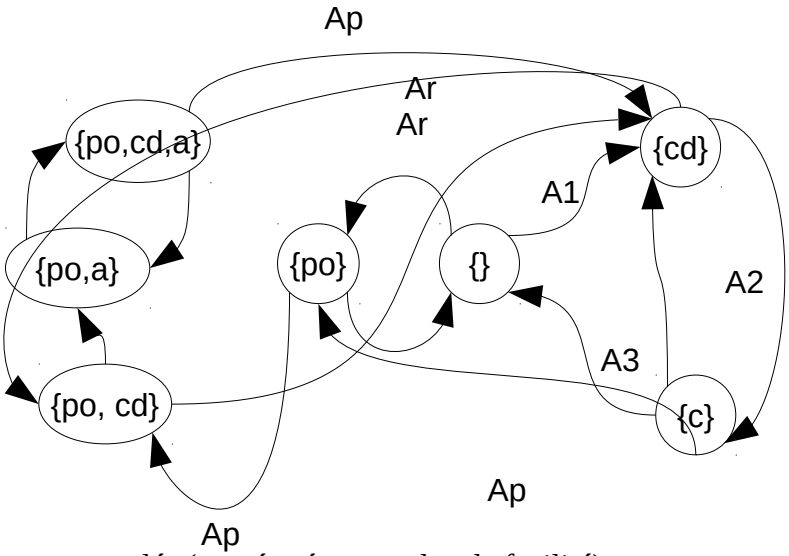
Question 1 :

Rappel une structure de Kripke est un modèle rudimentaire qui ne donne pas de nom aux changements d'états. Il faudra donc modéliser la commande de démarrage par une variable et indiquer ses conséquence en définissant les états successeur possible de ceux où cette variable est vraie. D'un point de vue pratique ceci capture bien le fait que si le démarrage est géré par un petit contrôleur la commande est vue potentiellement comme un booléen dans un programme.

L'ensemble de variable proposé est :

- po : vrai si la porte est ouverte
- c : vrai si le four chauffe
- cd : vrai si la commande de démarrage est passée (mais pas prise en compte)
- a : vrai si u message d'erreur est affiché

Il faut interpréter le graphe ci dessous tel qu'indiqué dans la recommandation du sujet (on indique uniquement les variables vraie mais un état = une valuation complète). La solution ci-dessous est une solution possible. Nous argumenterons pour expliquer pourquoi elle est acceptable.



Commentaire de quelques arcs clés (numérotés pour plus de facilité) :

Le cycke A1 A2 A3 identifie une cycle de cuisson standard

Les transitions Ap identifient les ouvertures de portes, Les transitions Ar identifient un retour à la normale. On notera que nous avons omis toutes les transitions bouclant sur les états (modélisant le fait que l'on ne fixe aucune échéance pour la progression du comportement du four – irréaliste mais plus facile dans un premier temps). On notera aussi que toutes les formules à formaliser par la suite ne sont pas vraies sur cette structure (en particulier la dernière). La spécification informelle ne décrivait pas ce qu'il se passait au moment où l'on ferme la porte.

Question 2 :

a) G F not (po) $\wedge$ not (cd) $\wedge$ not (a) $\wedge$ not (c) (explications donnée en TD)

b) G (cd => F c)

c) Un contre exemple trivial de cette exigence serait de prendre une exécution ou l'on maintient la porte ouverte, puis où l'on passe une commande qui sera ignorée pour finir dans l'état [po,a} et y rester. En gros, ce comportement est normal, c'est l'exigence qui n'est pas suffisamment précise.

d) G (cd $\wedge$ G not (po) => (F c))

e) Cette formule est un peu plus dure que les autres, décomposons :

Le signal d'alarme est actif tant que la porte n'est pas fermée :(a U not po)

Ceci ne doit être vérifié qu'a partir du moment où une commande est passée porte ouverte …. le problème réside dans le délai entre les deux instants : deux cas sont à considérés

i) le cas où l'on exige l'immédiateté de la réaction : G ((po $\wedge$ cd) => X (a U not po))

ii) le cas plus flexible où l'on ne borne pas la réaction : G ((po $\wedge$ cd) => ((po $\wedge$ cd) U (a U not po))

Cette seconde formule dit un peu plus que l'exigence mais ceci sera discuté en TD.

f) Oui. La difficulté réside dans la définition de « ignoré ». Forcer la structure à vérifier po => not (cd) serait trop contraignant. En revanche, il faudrait pouvoir garantir que si cd est vrai lorsque la porte est ouverte alors dès le moment où po est fermée, cd devient faux (effet ignoré).

Not F((po $\wedge$ cd) $\wedge$ (po U ((not po) $\wedge$ cd)))

Il n'existe pas d'instant dans le futur tel que la porte est ouverte et la commande activée est telle que la porte soit maintenue ouverte jusqu'à ce que la porte soit fermée et la commande activée.

Question 3 :

Un automate étendu permet d'augmenter l'expressivité des automates finis via l'ajout dans leur définition de variables entières. Ici, nous avons de la variable indiquant la durée de cuisson courante : T

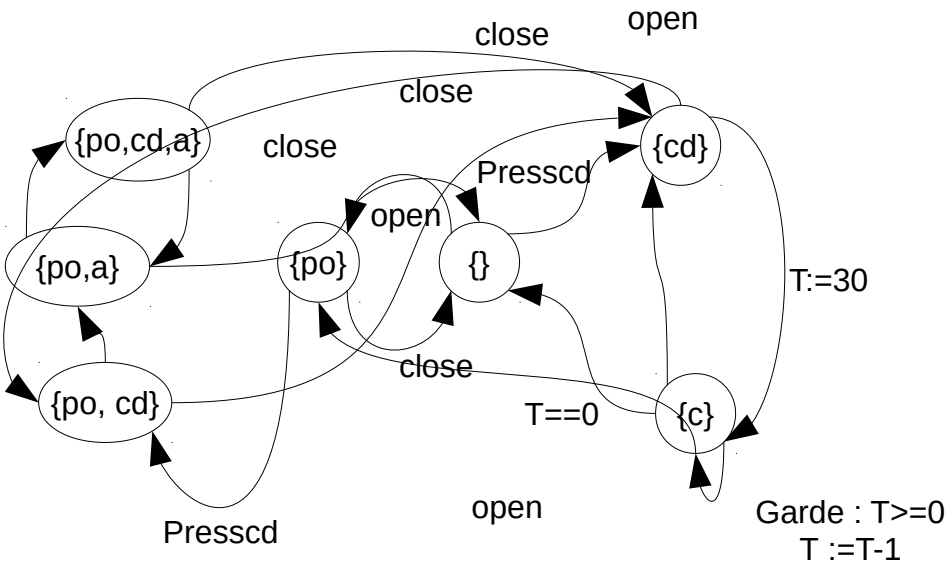
Il y aura deux manière d'utiliser T :

- initialisé à 30 et décrémentée jusqu'à 0

- initialisé à 0 et incrémenté jusqu'à 30

Nous choisissons la première solution (cela aura un impact sur la question 4 (cf page suivante)

Question 4 G (T>=0) ou G (T<=30)


close
open
close
{po,cd,a}
close
{cd}
Presscd
open
{po,a}
{po}
{}
T:=30
close
{po, cd}
T==0
{c}
open
Garde : T>=0
T :=T-1
Presscd