Ensimag
3ème année - filière ISI
Février 2017

Examen

Modélisation et Vérification de Systèmes Concurrents et Temps-Réel

Durée 2 heures
Tous documents autorisés

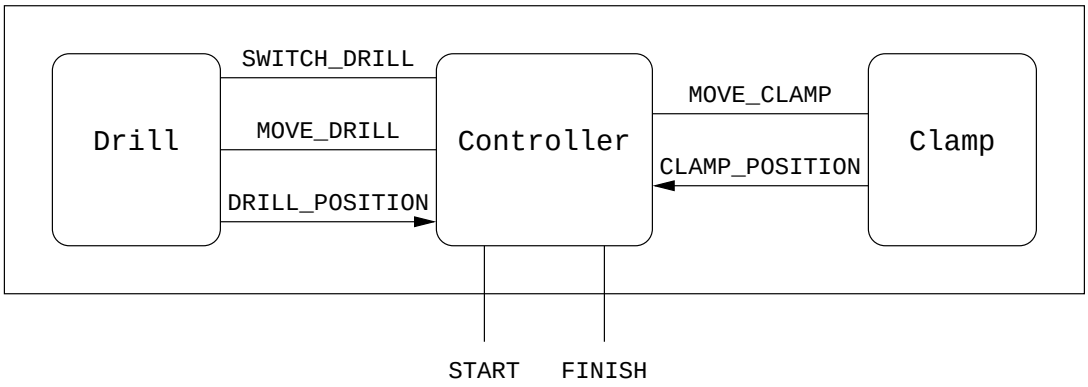
Avertissement : Il vous est demandé d'apporter le plus grand soin dans la rédaction. Vous serez jugés plus sur la QUALITÉ que sur la QUANTITÉ de vos réponses.

Le barême est donné à titre purement indicatif.

Partie I Modélisation en LNT (10 points)

On se propose de spécifier formellement en LNT le module de perçage d'un système industriel d'usinage de pièces métalliques appelé "*table tournante*".

L'architecture de la table tournante est illustrée sur la figure ci-dessous, qui met en évidence les communications et les synchronisations entre les différents éléments du système. Les flèches indiquent le sens de transmission des données et les traits sans flèches indiquent des synchronisations sans communication de données.



La table tournante est ainsi constituée des éléments suivants :

- Une pince, représentée par le processus `Clamp`, permet d'immobiliser une pièce positionnée dans le module. La pince peut être soit en position serrée, soit en position desserée. Elle change de position lorsqu'elle reçoit une commande représentée par la porte `MOVE_CLAMP`. Dès qu'elle a atteint sa nouvelle position, elle envoie un signal, représenté par une action sur la porte `CLAMP_POSITION`, l'offre Booléenne `TRUE` indiquant que la pince est en position serrée et `FALSE` indiquant que la pince est en position desserrée. On suppose que la pince est initialement en position desserrée.
- Une perceuse, représentée par le processus `Drill`, permet de percer l'objet enserré dans la pince. La perceuse peut être dans l'état marche ou arrêt et en position haute ou basse. Elle change d'état

dès qu'elle reçoit une commande représentée par une action sur la porte `SWITCH_DRILL` et change de position lorsqu'elle reçoit une commande représentée par une action sur la porte `MOVE_DRILL`. Dès qu'elle a atteint sa nouvelle position, elle envoie un signal, représenté par une action sur la porte `DRILL_POSITION`, l'offre Booléenne `TRUE` indiquant que la perceuse est en position basse et `FALSE` indiquant que la perceuse est en position haute. On suppose que la perceuse est initialement arrêtée et en position haute.

- Un contrôleur, représenté par le processus `Controller`, implémente l'interfaçage entre la table tournante et le reste du système d'usinage (que l'on ne cherchera pas à spécifier ici). Il reçoit du reste du système la commande représentée par une action sur la porte `START` qui lui indique qu'une pièce est positionnée pour être percée, émet successivement à destination de la pince et de la perceuse les commandes représentées par les actions sur les portes `SWITCH_DRILL`, `MOVE_DRILL` et `MOVE_CLAMP` décrites ci-dessus, synchronise ses actions en fonction des événements reçus, représentés par les portes `DRILL_POSITION` et `CLAMP_POSITION` décrites ci-dessus, et indique au reste du système la fin du perçage par l'émission d'un signal de fin de tâche représenté par une action sur la porte `FINISH`.

Question I.1 L'ensemble du système

Considérant que le profil des processus `Controller`, `Drill` et `Clamp` est déjà défini (voir les questions suivantes), compléter le corps du processus LNT `MAIN` suivant, qui décrit l'architecture illustrée dans la figure précédente, en considérant que la partie encadrée correspond à du comportement interne. On veillera à typer convenablement les portes de communication en utilisant soit le channel prédéfini `None` (qui indique que la porte ne comporte aucune offre de communication), soit le channel `Bool` défini ci-dessous (qui indique que la porte comporte une offre de communication de type `Bool`).

```
channel Bool is (Bool) end channel

process MAIN [START, FINISH : None] is
   (* partie a completer : 8-10 lignes environ *)
end process
```

Question I.2 Le processus Clamp

Le processus `Clamp` est défini comme suit. Dessiner le système de transitions étiquetées correspondant à ce processus.

```
process Clamp [MOVE_CLAMP : None, CLAMP_POSITION : Bool] is
   var
      locked:Bool
   in
      locked := false;
      loop
         MOVE_CLAMP;
         locked := not (locked);
         CLAMP_POSITION (locked)
      end loop
   end var
end process
```

Question I.3 Le processus Controller

Lorsqu'il reçoit une commande de perçage représentée par une action sur la porte `START`, le processus `Controller` a un fonctionnement cyclique consistant successivement à :

- commander à la pince de se placer en position serrée,
- une fois assuré que la pince est en position serrée, mettre la perceuse en marche,
- commander à la perceuse de se placer en position basse,
- une fois assuré que la perceuse est en position basse, lui commander de se replacer en position haute,
- une fois assuré que la perceuse est en position haute, l'arrêter,
- commander à la pince de se desserrer,
- enfin, une fois assuré que la pince est desserrée, émettre le signal de fin de tâche et attendre une nouvelle commande de perçage.

Compléter la définition de ce processus ci-dessous.

```
process Controller [START, FINISH, SWITCH_DRILL, MOVE_DRILL, MOVE_CLAMP : None,
                    DRILL_POSITION, CLAMP_POSITION : Bool] is
   (* partie a completer : 15 lignes environ *)
end process
```

Question I.4 Le processus Drill

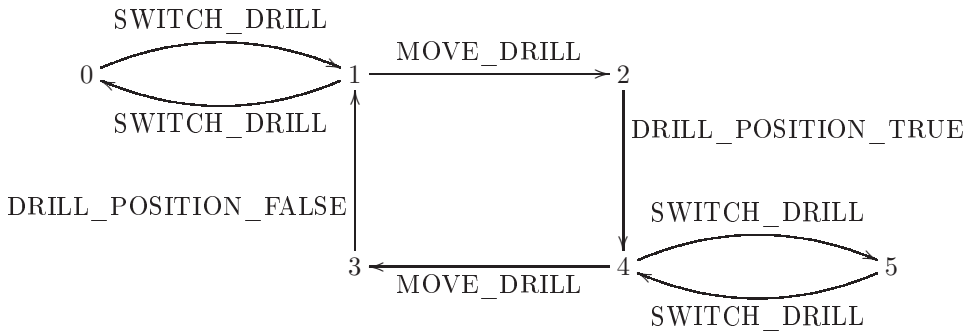
Le processus `Drill` a le comportement cyclique suivant. Lorsque la perceuse est en marche (et uniquement dans ce cas) le processus peut recevoir une commande de déplacement. Dans ce cas, la commande est suivie d'un signal indiquant la nouvelle position de la perceuse via la porte `DRILL_POSITION` puis le processus se tient prêt à recevoir une nouvelle commande. Lorsque la perceuse n'est pas en train de se déplacer, le processus peut également recevoir une commande de changement d'état (mise en marche ou arrêt). Dans ce cas, le processus change d'état et se tient prêt à recevoir une nouvelle commande.

Compléter la définition ci-dessous de ce processus.

```
process Drill [SWITCH_DRILL, MOVE_DRILL : None, DRILL_POSITION : Bool] is
   (* partie a completer : 15-20 lignes environ *)
end process
```

Partie II Logique temporelle (5 points)

On considère le système de transitions étiquetées suivant, dont l'état initial est l'état 0 :



Indiquer quels états satisfont chacune des formules de logique temporelle suivante :

1. $[\,\text{true}\,]\,\text{false}$
2. $\langle\,\text{MOVE\_DRILL}\,\rangle\,\text{true}$
3. $[\,\text{SWITCH\_DRILL}\,]\,\text{false}$
4. $[\,\text{DRILL\_POSITION\_TRUE}\,]\,\text{true}$
5. $\langle\,\text{true}^*.\text{DRILL\_POSITION\_TRUE}\,\rangle\,\text{true}$
6. $[\,\text{MOVE\_DRILL}.(\neg\text{DRILL\_POSITION\_FALSE})^*.\text{MOVE\_DRILL}\,]\,\text{false}$
7. $\mu X.\langle\,\text{DRILL\_POSITION\_TRUE}\,\rangle\,\text{true} \vee \langle\,\text{DRILL\_POSITION\_FALSE}\,\rangle\,\text{true} \vee \langle\,\text{MOVE\_DRILL}\,\rangle\,X$
8. $\nu X.\langle\,\text{SWITCH\_DRILL}\,\rangle\,X$

Parmi ces formules, une seule est toujours vraie quel que soit le système de transitions étiquetées sur lequel elle est évaluée. Laquelle ?

Question bonus : Donner une formule HML (sans point fixe mais avec des expressions régulières) équivalente à la propriété numéro 7.

Partie III Automates temporisés (5 points)

On se propose dans cette partie de spécifier une version temporisée de la table tournante en utilisant des automates temporisés. Le modèle d'automates utilisé permet d'une part le passage de valeurs lors de la synchronisation sur les actions selon les conventions du langage LNT et d'autre part l'utilisation de types et fonctions LNT pour la description et la manipulation des variables.

La spécification de la table tournante suit la spécification en LNT, en représentant chaque processus par un automate temporisé. Nous conservons donc dans cette partie les noms des portes, les noms des opérations et les noms des paramètres.

Les réponses attendues consistent en le dessin des automates temporisés accompagnés des déclarations de leurs variables (avec leur type) et horloges. Il vous est demandé de suivre les conventions syntaxiques suivantes:

- L'état initial est marqué par un cercle double (comme en UPPAAL).
- Toute transition possède une étiquette de la forme « $g/a;r$ », où g est la garde, a est l'action et r l'affectation d'horloges et de variables de la transition; « *true* » représente la garde toujours valide, « i » l'action interne et « *null* » l'affectation vide.

Question III.1 Le processus Clamp

Nous donnons ici une description du processus `Clamp` qui étend la spécification donnée dans la Question I.2, la partie écrite en *italique* correspondant aux contraintes temporelles.

Le processus `Clamp` fonctionne de manière cyclique en recevant une commande de déplacement, puis en émettant *après exactement 2 unités de temps (temps de déplacement de la pince)* un signal indiquant la nouvelle position atteinte.

Dessiner l'automate temporisé correspondant.

Question III.2 Le processus Controller

Nous reprenons ici la description du processus `Controller` donnée dans la Question I.3, la partie écrite en *italique* correspondant aux contraintes temporelles.

Lorsqu'il reçoit une commande de perçage représentée par une action sur la porte `START`, le processus `Controller` a un fonctionnement cyclique consistant successivement à :

- commander à la pince de se placer en position serrée *au plus tard après 1 unité de temps*,
- une fois assuré que la pince est en position serrée (*ce qui doit être le cas au plus tard après 3 unités de temps*), mettre la perceuse en marche *au plus tard après 1 unité de temps*,
- *après exactement 1 unité de temps (temporisation)*, commander à la perceuse de se placer en position basse,
- une fois assuré que la perceuse est en position basse (*ce qui doit être le cas au plus tard après 5 unités de temps*), lui commander de se replacer en position haute *au plus tard après 1 unité de temps*,
- une fois assuré que la perceuse est en position haute (*ce qui doit être le cas au plus tard après 5 unités de temps*), l'arrêter *au plus tard après 1 unité de temps*,
- *après exactement 1 unité de temps (temporisation)*, commander à la pince de se desserrer,
- enfin, une fois assuré que la pince est desserrée (*ce qui doit être le cas au plus tard après 3 unités de temps*), émettre le signal de fin de tâche *au plus tard après 1 unité de temps* et attendre une nouvelle commande de perçage.

Si les valeurs attendues ne sont pas reçues dans les délais, le processus entre dans un état d'erreur.

Dessiner l'automate temporisé correspondant.

Question III.3 Le processus Drill

Nous reprenons ici la description du processus Drill donnée dans la Question I.4, la partie écrite en *italique* correspondant aux contraintes temporelles.

Le processus `Drill` a le comportement cyclique suivant. Lorsque la perceuse est en marche (et uniquement dans ce cas) le processus peut recevoir une commande de déplacement. Dans ce cas, la commande est suivie *après 3 à 4 unités de temps (temps de déplacement de la perceuse, variable suivant la résistance de la pièce métallique)* d'un signal indiquant la nouvelle position de la perceuse via la porte `DRILL_POSITION` puis le processus se tient prêt à recevoir une nouvelle commande. Lorsque la perceuse n'est pas en train de se déplacer, le processus peut également recevoir une commande de changement d'état (mise en marche ou arrêt). Dans ce cas, le processus change *immédiatement* d'état et se tient prêt à recevoir une nouvelle commande. *Si elle est en marche depuis 20 unités de temps sans avoir changé de position, la perceuse se replace automatiquement dans son état d'arrêt.*

Dessiner l'automate temporisé correspondant.