Systèmes concurrents

décembre 2019
1h30, documents autorisés : notes et supports distribués en cours, TD, TP.
Tous appareils électroniques prohibés, sauf autorisation expresse.

Les exercices sont indépendants. Le barème est indicatif.
Les notes seront (éventuellement) écrêtées à 20, qui est la note maximale.
L'épreuve comporte **2 parties**, qui devront être **rendues séparément**.

Conseils :

— il est généralement préférable de **lire attentivement et complètement** l'énoncé des questions avant de répondre.

— résoudre un problème par sémaphores est souvent sensiblement plus chronophage que le résoudre par moniteurs ou rendez-vous.

1 Exclusion mutuelle (6 points)

On étudie le problème de l'exclusion mutuelle sur un système multiprocesseur à mémoire partagée. Il n'y a pas de temps partagé d'un processeur par plusieurs processus : un unique code s'exécute sur chaque processeur.

Soit l'algorithme suivant dû à Michael Fischer, où x est un registre atomique en mémoire partagé initialisé à 0 et chaque processeur a une identité i (différente de 0) :

```
repeat
    while x ≠ 0 do skip;
    x ← i;
    délai
until x = i;
-- section critique
x ← 0;
```

1. Considérer le cas avec deux processeurs. Montrer que les accès à x doivent être atomiques, c'est-à-dire que x ne doit pas être lu/écrit dans un cache local à chaque processeur sans mise à jour simultanée de la mémoire partagée.
2. On considère dans la suite que les accès à x sont atomiques. Toujours avec deux processeurs, montrer qu'avec un délai nul, il est possible que les deux processeurs entrent simultanément dans la section critique (donner un entrelacement des instructions qui conduit à cet état).
3. Sûreté : en déduire quel est le délai minimal avec deux processeurs pour que l'algorithme assure qu'au plus un processeur exécute la section critique.
4. Vivacité : dans ce cas avec deux processeurs, montrer que même si les deux processeurs demandent simultanément (ou à peu près) l'accès à la section critique, alors l'un des deux réussira nécessairement à l'obtenir.
5. Sachant que dans un système multiprocesseur à mémoire partagé, un seul accès mémoire (lecture ou écriture) est possible à un instant donné et prend une durée supposée fixe Δ, déterminer le délai minimal nécessaire avec N processeurs.
6. Vivacité : montrer que dans le cas avec $N > 2$ processeurs, il est possible (bien qu'improbable) qu'un processeur soit définitivement en famine et n'obtienne jamais l'accès (il suffit de considérer $N = 3$).

2 Transactions (4 points)

7. Les protocoles de contrôle de concurrence pessimistes présentés en cours assurent la sérialisabilité des transactions en se basant sur un ordre défini soit par des estampilles, soit par l'utilisation de verrous. Le protocole de contrôle de concurrence par certification présenté en cours [1] est un exemple de protocole de contrôle de concurrence optimiste basé sur l'utilisation d'estampilles.

   Serait-il envisageable de définir un protocole de contrôle de concurrence optimiste basé sur l'utilisation de verrous ? Justifiez votre réponse (si oui, expliquez comment ; si non, expliquez pourquoi).

8. Le chronogramme suivant représente l'exécution de quatre transactions T_1, T_2, T_3, T_4 opérant sur les variables partagées x, y et z. Les variables a, b, c, d sont locales à respectivement T_1, T_2, T_3, T_4. Dans ce chronogramme, le temps réel s'écoule de haut en bas, l'opération tdébut() lance une transaction, l'opération tfin() demande la validation de la transaction qui l'appelle.

   (a) Construire le graphe de dépendance correspondant à cette exécution.

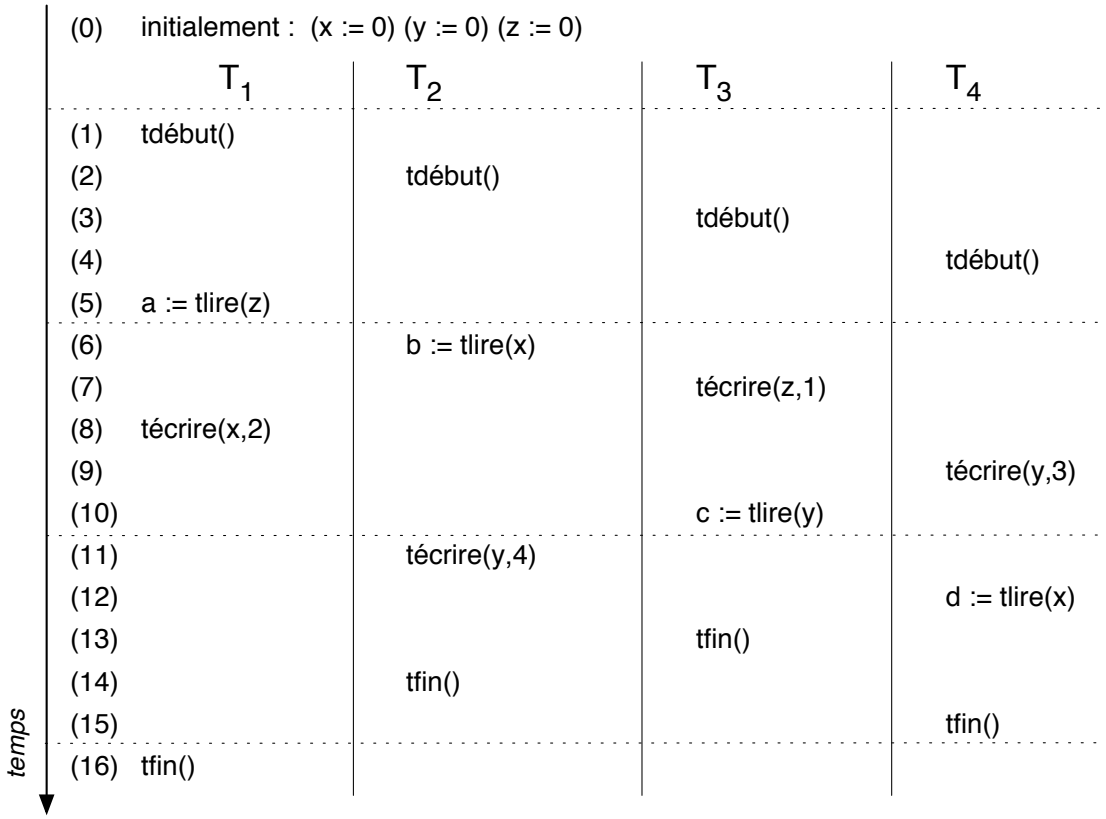
      i. dans le cas où l'on utilise une stratégie de propagation directe des écritures ;

      ii. dans le cas où l'on utilise une stratégie de propagation différée des écritures ;

   Précisez pour chacun des arcs le numéro de l'opération entraînant la création de l'arc, et indiquer, en justifiant votre réponse, si cette exécution est sérialisable.

   (b) On suppose que l'exécution des transactions est contrôlée selon un protocole de contrôle de concurrence par certification, et une propagation différée des écritures. Indiquez quel sera le résultat final de cette exécution (dernière valeur des différentes variables, transactions validées, transactions abandonnées).

   (c) Même question dans le cas d'un contrôle de concurrence continu par estampilles, en supposant que les estampilles sont attribuées selon l'ordre de lancement des transactions.



1. Huitième partie (Transactions), planches 29 et 30.

Merci de répondre à cette partie (3) sur une copie séparée.

3 Santa Claus 2.0

L'entreprise Gafa dispose d'un entrepôt dans lequel œuvre une armée de robots autonomes, ainsi que d'une flotte de drones. L'entrepôt possède une unique plateforme d'embarquement pouvant accueillir un seul drone à la fois.

L'activité d'un drone est cyclique. Un cycle du drone consiste à se recharger en énergie, puis à se mettre en place sur la plateforme lorsqu'elle est libre, ensuite à décoller une fois sa soute chargée, et enfin à livrer les clients. Lorsque la plateforme est occupée par un drone, les autres drones ayant besoin d'y accéder attendent leur tour.

L'activité d'un robot autonome est cyclique. Un cycle du robot consiste à se recharger en énergie, puis à préparer un colis à partir des articles disponibles dans l'entrepôt, ensuite à déposer ce colis sur un tapis de chargement, et enfin à embarquer ce colis dans la soute du drone garé sur la plateforme lorsque cela est possible, c'est-à dire lorsqu'un drone est effectivement garé sur la plateforme et que l'ajout du colis dans la soute ne met pas le drone en surcharge. Les drones peuvent en effet enlever une charge limitée à $MaxCharge$. Les colis sont conditionnés de sorte à ce que le poids d'un colis n'excède jamais $MaxCharge$. Lorqu'un robot charge la soute d'un drone, les autres robots prêts à charger attendent leur tour.

Un drone décolle dans le cas où l'ajout du prochain colis n'est pas possible à cause de la surcharge, ou lorsqu'il n'y a plus de colis à déposer et plus de robots en attente pour charger. Un drone ne peut cependant décoller à vide.

Les opérations à implanter sont

- apponter(), qui permet à un drone d'atterrir et se garer sur la plateforme lorsqu'elle est libre ;
- décoller(), qui permet à un drone chargé de quitter la plateforme ;
- déposer(int c), qui permet à un robot de déposer un colis de poids c sur le tapis de pesée ;
- embarquer(), qui permet à un robot de mettre en soute le colis se trouvant sur le tapis de pesée, si cela est possible ;

Les drones sont représentés par des processus ayant le comportement suivant :

```
processus Drone
répéter
    repos();    // recharge et maintenance
    apponter() ;
    décoller() ;
    livrer() [2];
sans fin
```

Les robots autonomes sont représentés par des processus ayant le comportement suivant :

```
processus Robot
répéter
    repos();    // recharge et maintenance
    c := prendre_colis();   // c : poids du colis pris
    déposer(c) ;
    embarquer() ;
sans fin
```

On suppose que les robots et les drones sont initialement au repos(), et que la quantité de colis à livrer est sans limite.

2. Rassurez vous : vous serez toujours livrés en temps et en heure, car la société Gafa peut vous localiser où que vous soyez, et anticiper vos moindres déplacements.

3.1 Sémaphores (7 points)

On suppose que l'on dispose de sémaphores (généraux), définis par une classe Semaphore, fournissant **uniquement** les méthodes down()/up() (ou P()/V() si vous préférez), et un constructeur permettant de fixer la valeur initiale du sémaphore.

Les variables partagées seront précédées du mot-clé global, les variables locales à un processus seront déclarées après l'identifiant du processus.

9. Donner le code des opérations apponter(), décoller(), déposer(int c), embarquer() de manière à assurer le fonctionnement décrit précédemment et à garantir que dès lors qu'une ou plusieurs opérations sont possibles, au moins l'une de ces opérations est exécutée.

3.2 Moniteurs (7 points)

On suppose que l'on dispose de **moniteurs de Hoare** : exclusion mutuelle garantie sur les opérations du moniteur (évitant d'avoir à programmer explicitement la prise ou la libération d'un verrou d'exclusion mutuelle), priorité au signalé, files FIFO associées aux variables condition.

Fournir l'implémentation d'un moniteur M proposant les 4 opérations apponter(), décoller(), déposer(int c), embarquer() de manière à assurer le fonctionnement décrit précédemment et à garantir que dès lors qu'une ou plusieurs opérations sont possibles, au moins l'une de ces opérations est exécutée.

On réalisera le moniteur en suivant la démarche vue en TD, basée sur l'évaluation de conditions d'état pour contrôler la progression des processus.

10. (2 points) Ecrire (en français) les conditions d'acceptation des différentes opérations.
11. (0,5 point) Définir les variables d'état permettant de représenter ces conditions.
12. (0,5 point) Etablir un invariant du moniteur liant ces variables d'état
13. (4 points) Programmer les opérations (et la fonction) du moniteur. Préciser (sous forme de commentaire dans le code) les pré/post conditions des signaler/attendre.

3.3 Processus communicants (4 points)

Répondre au choix avec des tâches Ada ou avec des activités communicantes par canaux (CSP/Go).

14. Construire une tâche serveur Ada / une activité de synchronisation, proposant des entrées / canaux analogues à celles du moniteur précédent, assurant le fonctionnement décrit précédemment et garantissant que dès lors qu'une ou plusieurs opérations sont possibles, au moins l'une de ces opérations est exécutée. .

    Utiliser au choix la méthodologie par conditions d'acceptation ou la construction d'un automate.