MPRI 2.3 2014/2015: Algorithms for Concurrent Systems
Midterm Exam, 24.11.2014

Open book (all paper documents allowed), no gadgets. Duration: 3h

Name:

Problem 1 (4 points)

Classify the following properties into safety/liveness. If a property is an intersection of the two, specify the corresponding safety and liveness properties. Justify your answers.

- Every process eventually outputs a value.

- No two processes output different values.

- Every process eventually outputs a previously proposed input of some process or crashes (stops taking steps).

- No two correct processes output different values.

Problem 2 (4 points)

We say that a property P *is stronger than* a property P if $P \subseteq P'$. What is the relation between *starvation-freedom* (SF) and *lock-freedom* (LF)? Explain why.

Problem 3 (4 points)

Give an algorithm that implements a safe 1WNR M-valued register (for some fixed M) using $\lceil \log M \rceil$ safe 1WNR binary registers. Provide a proof of correctness.

If we replace the safe binary registers with *regular* ones, do we get a *regular* M-valued register implementation?

Algorithmique distribuée avec mémoire partagée	**Examen Final**
MPRI	**Annales partie 2**

Tous les exercices sont indépendants.

Dans toutes les questions qui suivent n est le nombre total de processus dans le système (on suppose que $n > 1$), les processus sont nommés $p_0, ...p_{n-1}$.

Exercice 1.— Un objet atomique *bin-consensus* est un objet atomique qui permet à n processus de réaliser un consensus binaire. Un objet atomique *multi-consensus* permet à n processus de réaliser un consensus où les processus peuvent proposer une valeur prise dans un ensemble $V = \{0, \ldots, 2^k - 1\}$ $(k > 1)$. Ecrire un protocole qui implémente de façon *Wait Free* un objet *multi-consensus* en utilisant des objets *bin-consensus* et des registres atomiques.

On acceptera une solution qui utilise n objets *bin-consensus* mais on donnera des points supplémentaires pour une solution qui utilise $O(k)$ objets *bin-consensus* (on peut utiliser un nombre quelconque de registres atomiques).

Exercice 2.— Dans un système à mémoire partagée, on considère l'objet atomique "k-test and set" qui peut être utilisé par n processus, noté $(n, k)TS$. Il a une opération $TS.complete_k()$ qui ne peut être appelée qu'au plus une fois par chaque processus. Il satisfait la spécification séquentielle suivante:

- la première invocation de $TS.complete_k()$ retourne 1
- les autres invocations retournent 0 ou 1
- au plus k invocations retournent la valeur 1

1. Montrer qu'avec des objets $(n, 1)TS$ et des registres on peut faire du consensus wait free pour 2 processus.
2. Montrer qu'avec des objets $(n, 1)TS$ et des registres on ne peut pas faire du consensus wait free pour 3 processus.
3. Montrer qu'avec des objets $(n, 2)TS$ et des registres on ne peut pas faire du consensus wait free pour 2 processus.
4. Montrer qu'avec des objets consensus et des registres on peut faire des objets $(n, 1)TS$ wait free.

Exercice 3.— Dans un système à mémoire partagée de n processus, on considère le problème du "k accord", noté $(n, k)SA$. Le seule opération du k accord est $SA.propose_k(v)$ qui a un paramètre entier v (la valeur proposée) et retourne un entier (la valeur décidée). Cette opération ne peut être appelée qu'au plus une fois par chaque processus. Il satisfait la spécification suivante:

- au plus k valeurs différentes sont décidées,
- toute valeur décidée est une valeur proposée.

On dispose d'une implémentation wait-free du $(n, k)SA$.

Pour réaliser $(n, k)TS$ (défini dans l'exercice précédant), on considère l'algorithme suivant qui utilise un tableau R de registres partagés et des variables locales:
Code de p_i:

$TS.complete_k()$::
$R[i] = SA.propose_k(p_i)$
$X = \emptyset$
`for` j `from` 0 `to` $n-1$
$X = X \cup \{R[j]\}$
`if` $p_i \in X$ `then return` 1 `else return` 0

1. Montrer qu'au plus k processus retournent la valeur 1.
2. Montrer que si q correspond à la première valeur écrite dans le tableau R alors le processus q ne peut retourner que 1.
3. Montrer que l'algorithme implémente un objet (atomique) $(n,k)TS$ de façon wait-free.

Exercice 4.— Dans un système à mémoire partagée, $(n,k)TS$ et $(n,k)SA$ sont définis comme dans les deux précédents exercices.

1. Soient A et B deux registres (atomiques) partagés initialisés à 0. Le processus p écrit 1 dans A, lit B et affecte la valeur lue à sa variable locale x. Le processus q écrit 2 dans B, lit A et affecte la valeur lue à sa variable locale y.

   En supposant que p et q terminent, quelles peuvent être les valeurs du couple (x,y) après ces instructions?
2. Montrer qu'avec des objets $(k+1,k)TS$ et des registres on peut faire du $(k+1,k)SA$ wait free.
3. Montrer qu'avec des objets $(n,k)TS$ et des registres on peut faire du $(n,k)SA$ k-résilient (tolérant k pannes par arrêt (crash)).