Sciences
Université Paris Cité

UFR informatique MPRI - Année 2022-23

Algorithmique distribuée avec mémoire partagée

Devoir à rendre pour le 4 novembre 2022

Sauf mention explicite du contraire, pour tous les exercices on considère un ensemble de n ($n > 1$) processus $\{p_1, p_2, ..., p_n\}$ communiquant via des registres atomiques. Tous les exercices sont indépendants.

Exercice 1.— Montrer que le numéro de consensus d'un objet pile est 2.

Exercice 2.—
Un objet compteur_faible a comme état interne une valeur entière (initialement 0) et a 2 opérations *inc*, qui incrémente le compteur de 1 et *read*, qui retourne la valeur du compteur.
Un objet compteur a comme état interne une valeur entière (initialement 0) et a une operation *inc-read*, qui incrémente le compteur de 1 et retourne l'ancienne valeur.

1. Montrer que le numéro de consensus d'un compteur_faible est 1. Pour cela on pourra donner une implémentation wait-free linéarisable d'un compteur_faible (indication: l'implémentation peut utiliser des atomic snapshots).
2. Montrer qu'un objet compteur a comme numero de consensus 2.
3. (plus difficile) Donner un algorithme *obstruction-free* de consensus *binaire* pour n processus en utilisant seulement 2 objets compteur_faible (pas de registres).

Exercice 3.— Soit M un entier, donner un algorithme (et montrer qu'il est correct) qui implémente un registre mono-écrivain multi-lecteurs (SWMR) M-valué sûr (safe) en utilisant $O(logM)$ registres mono-écrivain multi-lecteurs (SWMR) binaires sûrs (safe).
Si on remplace les registres binaires sûrs par des registres binaires réguliers (regular) dans l'algorithme précédent, implémente-t-on un registre mono écrivain multi lecteurs (SWMR) M-valué régulier (regular)?

Exercice 4.— On considère un système à mémoire partagée où les seules pannes sont des pannes *initiales* (un processus en panne ne fait aucun pas de calcul).

1. Si t ($0 \leq t < n$) processus peuvent être initialement en panne, donner un algorithme de consensus dans ce système.

   Pourquoi la preuve d'impossibilité du consensus dans un système t-résilient (t processus peuvent tomber en panne au cours de l'exécution) ne s'applique pas ici?
2. Si maintenant le système est un système avec émission-réception de message (message-passing) avec des pannes initales uniquement, montrer qu'on ne peut réaliser le consensus si $t \geq n/2$, (on pourra utiliser un argument de partition). Si $t < n/2$ peut-on réaliser le consensus?

Exercice 5.— Dans cet exercice les divers objets considérés sont "one-shot" : ils ne peuvent être appelés qu'une seule fois par un processus. 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
$V = \{0, ..., 2^k - 1\} (k > 1)$.
Ecrire un protocole qui implémente de façon wait-free un objet *multi-consensus* en utilisant des objets *bin-consensus* et des registres atomiques. (indication: pour cette question et les questions suivantes, on rappelle que la seule possibilité pour qu'un consensus retourne à coup sûr une valeur est que tous les processus qui l'appellent proposent cette valeur)

1. Proposer une solution qui utilise n objets *bin-consensus*.
2. Proposer une solution qui utilise $|V|$ *bin-consensus*.
3. Proposer une solution qui utilise $O(k)$ objets *bin-consensus*.

Exercice 6.— 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 (one-shot). Il a la spécification 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 (atomiques) $(n, 1)TS$ wait-free.