Programowanie współbieżne

Lista zadań nr 8

Na ćwiczenia 9. grudnia 2021

Zadanie 1. Mamy trzy wątki: A, B, C oraz rejestry MRSW: XA, XB, XC. Każdy wątek może zapisywać swój rejestr oraz odczytywać wszystkie z nich. Poza tym, każdej parze wątków przypisujemy rejestr typu RMW (ang. *Read Modify Write*) udostępniający atomową operację **compareAndSet()**. Te rejestry to: RAB, RBC oraz RAC, a używać ich mogą jedynie wątki do nich przypisane. Wykaż, że nie istnieje nieczekająca implementacja protokołu binarnego konsensusu dla trzech wątków, używająca wyłącznie wymienionych wyżej zasobów.

Zadanie 2. Funkcja **double_cAS()** działa bardzo podobnie do **compareAndSet()** z tym, że zapisuje jednocześnie albo dwa rejestry albo żaden. Bardziej formalnie, funkcja ta ma sygnaturę **double_cAS(r1, r2, expected, update)**. Jeśli obydwa rejestry **r1** i **r2** mają wartość **expected** to zostają nadpisane wartością **update**. W przeciwnym przypadku wartości zapisane w tych rejestrach nie ulegają zmianie. Wszystko to odbywa się w sposób atomowy.

Rozważmy sytuację jak z zadania poprzedniego, z tym że w miejsce instrukcji **compareAndSet()** możemy używać **double_cAS()**. Czy istnieje nieczekająca implementacja protokołu konsensusu dla trzech wątków, używająca wyłącznie wymienionych wyżej zasobów?

Zadanie 3. Definiujemy n-ograniczoną funkcję **compareAndSet(r, expected, update)** tak: pierwszych n wywołań funkcji na rejestrze **r** ma semantykę taką samą, jak standardowa funkcja **compareAndSet()**, w szczególności wartościami zwracanymi są **true** lub **false**, zależnie od wykonania aktualizacji rejestru. Następne wywołania funkcji wprowadzają rejestr **r** w stan wadliwy, co sprawia że wartością zwracaną jest $\perp$. Pokaż, że poziom konsensusu n-ograniczonej funkcji **compareAndSet()** dla $n \geq 2$ to dokładnie n.

Zadanie 4. Podaj nieczekającą implementację dwuwątkowego obiektu 2/3-przypisania (tablica ma 3 elementy, każdy wątek zapisuje ustalone 2 z nich) używając trzech obiektów (rejestrów) oferujących funkcje **compareAndSet()** oraz **get()** oraz (ewentualnie) rejestrów atomowych MRMW.

Zadanie 5. Rozważmy następujący dwuwątkowy obiekt **QuasiConsensus** z metodą **decide(v)**, gdzie **v** jest wartością binarną. Jeśli obydwa wątki, A i B, wywołały **decide()** z tą samą wartością **v**, to wspólnie uzgodnioną wartością jest **v** — **decide()** zwraca **v**. Jeśli wątki wywołały **decide()** z różnymi argumentami to albo muszą uzgodnić jedną z nich, albo B otrzyma wartość 0 i A otrzyma wartość 1 (ale nie na odwrót).

Dokładnie jedno z poniższych zadań ma rozwiązanie. Wybierz odpowiednie i rozwiąż je.

1. Pokaż, że poziom konsensusu dla obiektów **QuasiConsensus** jest ≥ 2. Tzn. zaimplementuj dwuwątkowy protokół konsensusu używając obiektów **QuasiConsensus** i rejestrów atomowych.
2. Pokaż, że poziom konsensusu dla obiektów **QuasiConsensus** wynosi 1.

Zadanie 6. Oto jedna z równoważnych definicji niewstrzymywania (ang. *lock-freedom*). Współbieżna metoda jest **niewstrzymywana** jeśli w sytuacji gdy wątki znajdują się w jej wnętrzu dostatecznie długo, to wykonanie przynajmniej jednego z nich postępuje. W szczególności, jeśli niektóre wątki są uśpione dostatecznie długo, to któryś z pozostałych postępuje. Konstrukcje i dowody nieistnienia analizowane przez nas dotychczas dla algorytmów nieczekających, działają również dla niewstrzymywanych. Pokaż to na przykładzie dowodu niemożliwości skonstruowania obiektu dwuwątkowego konsensusu przy pomocy atomowych rejestów.

Zadanie 7. Mówimy, że wątek wykonuje metodę **w izolacji** w pewnym przedziale czasowym, jeśli żadne inne wątki nie wykonują instrukcji tej metody w tym przedziale czasowym. Współbieżna metoda jest **niehamowana** (ang. *obstruction-free*) jeśli każdy wątek który od pewnego momentu wykonuje metodę przez cały czas w izolacji, zakończy ją.

Niehamowanie jest warunkiem istotnie słabszym od niewstrzymywania czy nieczekania: podaj niehamującą dwuwątkową implementację protokołu konsensusu używającą jedynie rejestrów atomowych.