Programowanie współbieżne

Lista zadań nr 4

Na ćwiczenia 16 listopada 2022

UWAGA: Na liście może pojawić się jeszcze do dwóch nowych zadań.

Zadanie 1. Pokaż, że poniższa implementacja zamka dla n wątków spełnia warunek wzajemnego wykluczania. Uzasadnij i wykorzystaj następujący fakt: każdy wątek wykonujący metodę **lock()** znajduje się na jednym z n-1 poziomów, z których ostatni oznacza zajęcie zamka. Na poziomie n - i znajduje się jednocześnie co najwyżej i wątków.

Wskazówka: The Art of Multiprocessor Programming 2e, rozdział 2.5.

```
class Filter implements Lock {
    int[] level;
    int[] victim;
    public Filter(int n) {
        level = new int[n];
        victim = new int[n]; // use 1..n-1
        for (int i = 0; i < n; i++) {
            level[i] = 0;
        }
    }
    public void lock() {
        int me = ThreadID.get(); // returns 0..n-1
        for (int i = 1; i < n; i++) { // attempt to enter level i
            level[me] = i;
            victim[i] = me;
            // spin while conflicts exist
            while (( ∃ k != me) (level[k] >= i && victim[i] == me)) {};
        }
    }
    public void unlock() {
        int me = ThreadID.get();
        level[me] = 0;
    }
}
```

Zadanie 2. Pokaż, że metoda **lock()** klasy Filter spełnia warunek niezagłodzenia. Wywnioskuj stąd, że spełnia również warunek niezakleszczenia.

Zadanie 3. Pokaż, że nie istnieje taka stała r, że operacja przejścia na kolejny poziom w metodzie **lock()** klasy Filter ma własność r-ograniczonego czekania. Za sekcję wejściową przyjmij instrukcje ustalające poziom i ofiarę. Dlaczego ten wynik nie jest sprzeczny z własnością niezagłodzenia?

Zadanie 4. W definicji r-ograniczonego czekania definiujemy sekcję wejściową jako złożoną z *kilku* pierwszych instrukcji algorytmu.

1. Przypomnij dowód tego, że algorytm Petersena jest FCFS (czyli 0-ograniczony).
2. Zmieńmy teraz definicję sekcji wejściowej tak, by składała się po prostu z *pierwszej* instrukcji w algorytmie. Czy, przy zmienionej definicji, algorytm Petersena nadal jest FCFS?
3. Czy istnieje algorytm implementujący zamek (czyli wzajemnie wykluczanie), który jest FCFS przy powyższej definicji sekcji wejściowej? Przeprowadź dowód nie wprost rozważając przypadki, gdy pierwszą instrukcją jest: 1. odczyt tej samej komórki pamięci lub różnych komórek, w zależności od wątku, 2. zapis do różnych komórek, 3. zapis do tej samej komórki.
4. Wyciągnij wniosek, że zmodyfikowana definicja sekcji wejściowej jest *bezsensowna*.

Definicja 1. Formalną definicję **linearyzacji** można streścić w dwóch punktach:

1. Ciąg wywołania metod w programie współbieżnym powinien mieć taki efekt, jak gdyby te metody zostały wykonane w pewnym sekwencyjnym porządku, jedna po drugiej.
2. Efekt każdej metody w programie współbieżnym powinien wystąpić w pewnym punkcie czasu pomiędzy jej wywołaniem a powrotem (punkt linearyzacji).

Zadanie 5. Sprawdź, czy poniższe diagramy reprezentują linearyzowalne historie. Użyj nieformalnej definicji linearyzacji z Definicji 1.

Zadanie 6. Powtórz zadanie 5, tym razem używając formalnej definicji linearyzacji (slajd 132). Dla każdego diagramu zdefiniuj odpowiadającą mu historię H. Jeśli to możliwe, zdefiniuj historię G oraz legalną sekwencyjną historię S spełniające warunki z definicji.

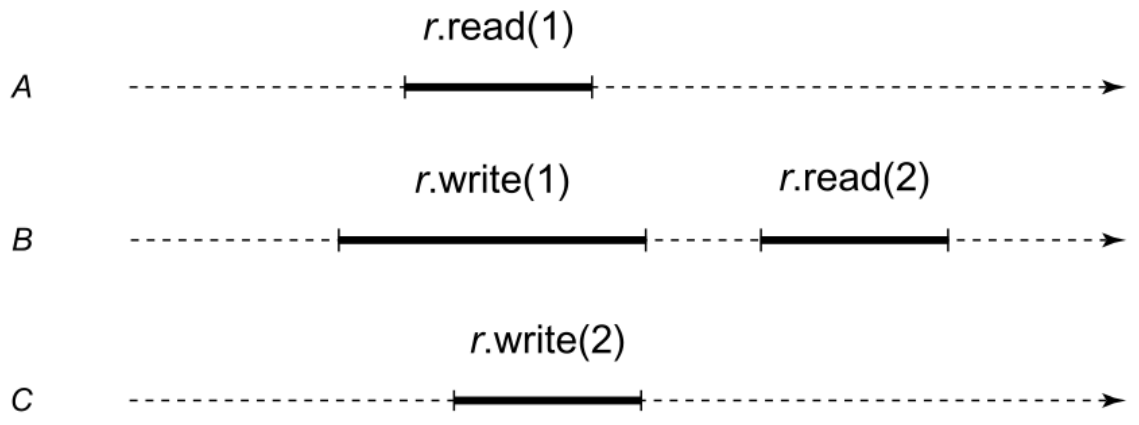

Diagram 1 (zapis/odczyt współdzielonej komórki pamięci r)

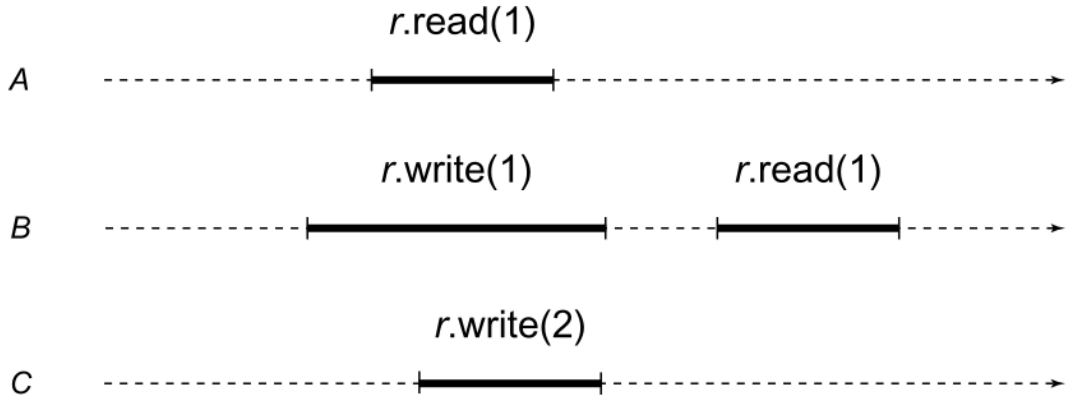

Diagram 2 (zapis/odczyt współdzielonej komórki pamięci r)

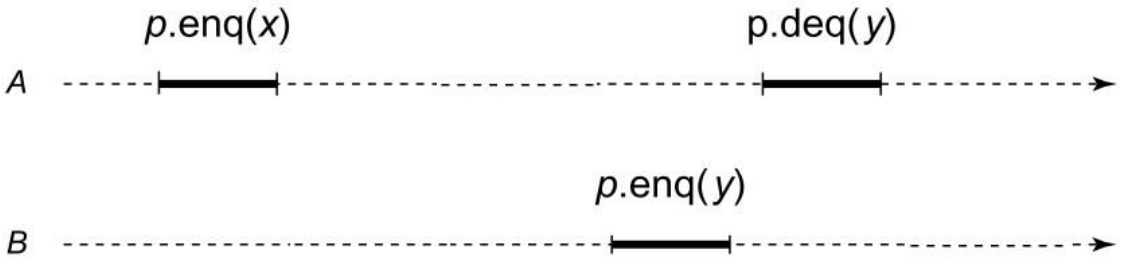

Diagram 3 (p jest kolejką FIFO)

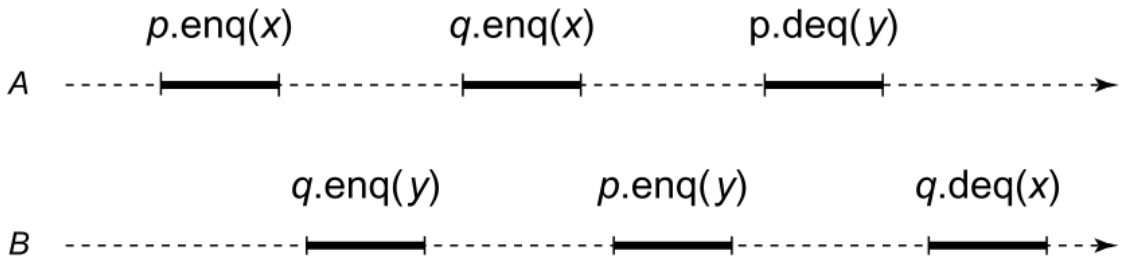

Diagram 4 (p i q są kolejkami FIFO)