Konrad Lisiecki 291649

Zadanie 1 - Algorytm Szymańskiego

Programowanie współbieżne i Rozproszone

Semestr letni 2013/14

Warszawa, 2014-03-17

Spis treści

Rozdział 1

Sumaryczne wyniki przedstawione w postaci tabeli

Własności algorytmów Czy odpowiedni wariant algorytmu Szymańskiego ma daną własność:

	2	3 (14,16,15)	3 (15,14,16)	3 (15,16,14)	3 (16,14,15)	3 (16,15,14)	5
Wzajemne wykluczanie	MA	MA	NIE MA	NIE MA	NIE MA	NIE MA	MA
Nieunikniona poczekalnia	NIE MA	NIE MA	NIE MA	NIE MA	NIE MA	NIE MA	NIE MA
Wyjście z poczekalni	MA	MA	NIE MA	NIE MA	NIE MA	NIE MA	NIE MA
Żywotność	NIE MA	MA	NIE MA	NIE MA	NIE MA	NIE MA	NIE MA
Liniowy czas oczekiwanie	MA	MA	NIE MA	NIE MA	NIE MA	NIE MA	NIE MA

Tabela 1.1: Własności wariantów algorytmu Szymańskiego

Rozdział 2

Sprawdzenie własności protokołu

Celem zadania jest zaimplementowanie Algorytmu Szymańskiego, który pozwala na zachowanie wzajemnego wykluczania dla więcej niż 2 procesów. Po zaimplementowaniu algorytmu można, za pomocą logiki LTL, sprawdzać wiele ciekawych własności dotyczących przebiegu wykonania procesów. Wśród nich możemy wyróżnić:

1. Wzajemne wykluczanie
2. Nieunikniona poczekalnia
3. Wyjście z poczekalni
4. Żywotność
5. Liniowy czas oczekiwania

W kolejnych podpunktach rozdziału przyjrzymy się bliżej tym własnościom.

2.1 Wzajemne wykluczanie

- program **MA** własność wzajemnego wykluczania. (w całym tekście, gdy program ma jakąś własność sprawdzamy to przez przejście i sprawdzenie wszystkich możliwych stanów). Formuła logiki LTL wygląda w następujący sposób:

Algorithm 1 Wzajemne Wykluczanie

1: $ltl\ sk\ \Box(counter < 2)$ ▷ Formuła LTL dla wzajemnego wykluczania

2.2 Nieunikniona poczekalnia

(każde wejście do sekcji krytycznej procesu i poprzedzone jest przez stan o własności we[i] && !chce[i]) - program **NIE MA** własności nieuniknionej poczekalni. Formuła logki LTL wygląda następująco:

Algorithm 2 Nieunikniona poczekalnia

1: $ltl\ np\ \Box(P[1]@sk \rightarrow nieunikniona\_poczekalnia[1])$ ▷ Formuła LTL dla nieuniknionej poczekalni

Komentarz do braku nieuniknionej poczekalni: Kiedy mamy tylko jeden proces, który chce wejść do sekcji krytycznej, to wchodzi tam nie przechodząc przez poczekalnię.

2.3 Wyjście z poczekalni

(jeśli któryś z procesów i ma własność we[i] && !chce[i], to w końcu któryś z pozostałych procesów j będzie miał własność wy[j]) - program **MA** własność wyjścia z poczekalni. Formuła logiki LTL wygląda jak poniżej:

Algorithm 3 Wyjście z poczekalni

1: *ltl wyj* $\Box(nieunikniona\_poczekalnia[0] \rightarrow \Diamond P[1]@tutaj \mid\mid \Diamond P[2]@tutaj \mid\mid \Diamond P[3]@tutaj)$ ▷ Formuła LTL dla wyjścia z poczekalni

2.4 Żywotność

Program **NIE MA** własności żywotności. Formuła logiki LTL wygląda jak poniżej:

Algorithm 4 Żywotność

1: *ltl zyw* $\Box(P[i]@przed\_pocz \rightarrow \Diamond P[i]@sk)$ ▷ Formuła LTL dla żywotności

Komentarz do braku żywotności: Rozpatrzmy następujący scenariusz: Załóżmy, że procesy 3 i 4 nie wyrażając chęci znalezienia się w sekcji krytycznej. W pewnym momencie proces 1 wykonuje cały przebieg algorytmu i zatrzymuje się dokładnie przed przypisaniem $chce[i] = false$ (instrukcja ta ma numer 16). Następnie procesor otrzymuje proces 2 i musi znaleźć się w poczekalni. Kolejno następuje przełączenie kontekstu, proces 1 kończy protokół końcowy i nie wyraża już zainteresowania znalezieniem się w sekcji krytycznej (pętli się we fragmencie własne sprawy). W ten sposób proces 2 nigdy nie wejdzie do sekcji krytycznej.

2.5 Liniowy czas oczekiwania

(podczas gdy jakiś proces czeka, żaden inny proces nie może wejść do sekcji krytycznej więcej niż stałą liczbę razy)- program **MA** własność liniowości czasu oczekiwania. Formuła LTL została wyrażona się jak poniżej:

Algorithm 5 Liniowy czas oczekiwania

1: *ltl lco* $\Box(licznik[0] \leqslant N)$ ▷ Formuła LTL dla liniowego czasu oczekiwania

Komentarz do tablicy licznik: Tablica licznik trzyma liczbę procesów, które weszły do sekcji krytycznej podczas oczekiwania przez dany proces.

Rozdział 3

Odporność własności algorytmu na zmianę kolejności przypisań

W celu sprawdzenia jak zachowują się poszczególne wersje algorytmu najpierw sprawdzałem jakie będą wyniki w przypadku gdy omawiane trzy istrukcje programu zostaną wykonane atomowo:

Algorithm 6 Atmomowe wykonanie 3 ostatnich instrukcji programu

```
1: atomic
2: {
3:       wy[i] = false;
4:       we[i] = false;
5:       chce[i] = false;
6: }
```

Po wprowadzeniu atomowości okazało się, że algorytm zyskał jedną własność, której poprzednio, nie miał, a mianowicie żywotność:

	ATOMIC EXECUTION
Wzajemne wykluczanie	MA
Nieunikniona poczekalnia	NIE MA
Wyjście z poczekalni	MA
Żywotność	MA
Liniowy czas oczekiwanie	MA

Tabela 3.1: Własności wariantów algorytmu Szymańskiego dla zatomizowanych ostatnich instrukcji

Z tego wynika, że musi istnieć inna kolejność wykonań, taka, że zajdzie własność żywotności. Okazało się, że permutacja $(14, 16, 15)$ zapewnia tę własność. Co więcej, wynikiem sprawdzenia pozostałych własności są 2 obserwacje:

1. Permutacja $(14, 16, 15)$ spełnia wszystkie własności oprócz nieuniknionej poczekalni. Oznacza to, że jest lepszym wariantem algotymu od tego, który był pierwotnie w zadaniu. Permutacja te spełnia własność, ponieważ przypisanie $chce[i] = false$ odbywa się przed przypisaniem $we[i] = false$. Wtedy to inny proces sprawdzający warunek na ominięcie poczekalni (druga pętla) otrzyma "zgode" na przejście od razu za poczekalnie.

2. Pozostałe permutacje (tzn. takie, dla których 14 instrukcja nie jest początkową) nie spełniają żadnych instrukcj, włączając w to własność wzajemnego wykluczania.

Wyniki analizay zostały przedstawione w poniższej tabeli:

	(14,15,16)	(14,16,15)	(15,14,16)	(15,16,14)	(16,14,15)	(16,15,14)
Wzajemne wykluczanie	MA	MA	NIE MA	NIE MA	NIE MA	NIE MA
Nieunikniona poczekalnia	NIE MA	NIE MA	NIE MA	NIE MA	NIE MA	NIE MA
Wyjście z poczekalni	MA	MA	NIE MA	NIE MA	NIE MA	NIE MA
Żywotność	NIE MA	MA	NIE MA	NIE MA	NIE MA	NIE MA
Liniowy czas oczekiwanie	MA	MA	NIE MA	NIE MA	NIE MA	NIE MA

Tabela 3.2: Własności wariantów algorytmu Szymańskiego dla poszczególnych permutacji ostatnich 3 instrukcji.

Rozdział 4

Możliwość "awarii" procesów, a własności algorytmu - pkt 5

W przypadku gdy istnieje możliwość awarii procesu, proces kończy swój normalny bieg i wraca do początku swojego wykonania. W przypadku zmodyfikowania modelu, jak w punkcie 4 zadania, algortym nie spełnia żadnych z własności, włącznie ze wzajemnym wykluczaniem. Po zaimplementowaniu tej części, postanowiłem odpowidnio zmodyfikować algorytm (zgodnie z załączonym artykułem o tym algorytmie). Po zmianie udało się uzyskać własność wzajmengo wykluczania. Implementacja znajduje się w pliku *prot2.pml*.

Wzajemne wykluczanie	MA
Nieunikniona poczekalnia	NIE MA
Wyjście z poczekalni	NIE MA
Żywotność	NIE MA
Liniowy czas oczekiwanie	NIE MA

Tabela 4.1: Własności algorytmu z możliwością "awarii".

Własności algorymu z awariami Jak zostało wspomniane wzajemne wykluczanie jest spełnione, natomiast pozostałe nie są spełnione. Jeśli chodzi o nieuniknioną poczekalnię i żywotność, to argumenty dlaczego własności te nie są spełnione są takie same jak w rozdziale drugim (bo może się zdarzyć, że procesy się nigdy nie będą psuły - czyli możemy powiedzieć, że oryginalny algorytm Szymańskiego jest szczególnym przypadkiem jego wersji z awariami). Jeżeli chodzi o własność trzecią (wyjście z poczekalni) to sprawa wydaje się oczywista, gdy uwzględnimy to, że procesy są podatne na awarie. Z tego wynika, że może zaistnieć sytuacja, kiedy procesy będą się "psuć" przed przypisaniem $wy[i] = true$ (instrukcja 13). Również dla liniowego czasu oczekiwania weryfikacja wykazała, że ta własność nie jest spełniona.