# Teoria Współbieżności
# Ćwiczenie 2

## 1 Cel ćwiczenia

Celem ćwiczenia jest wprowadzenie do analizy i dowodzenia poprawności algorytmów opisujących procesy działające współbieżnie.

## 2 Wprowadzenie

Każdy z przedstawionych procesów będzie wykonywał nieskończoną pętlę podzieloną na **sekcje lokalną** oraz **sekcje krytyczną**. Sekcja krytyczna może być wykonywana w danej chwili wyłącznie przez jeden proces a sekcji lokalna jest niezależna i związana ściśle z danym procesem.

Aby przedstawione algorytmy były uznane za działające poprawne muszą spełniać następujące warunki:

1. wzajemnego wykluczania - w sekcji krytycznej nie może działać więcej niż jeden proces
2. braku zakleszczenia - jeśli procesy chcą wejść do sekcji krytycznych w końcu któryś dostaje dostęp
3. braku zagłodzenia - brak scenariusza, w którym proces nigdy nie dostanie się do sekcji krytycznej

Aby formalnie pokać, że przedstawione procesy współbieżne nie spełniają powyższych punktów należy na przykład:

- podać listę istrukcji,
- diagram stanów (nie musi być cały, wystarczy część),
- tabelkę w stylu debbugera,

która nie spełni 1. 2. albo 3. . Zamiast całych instrukcji można napisać tylko numer instrukcji procesu; należy pamiętać, że zmienne również przedstawiają stan systemu.

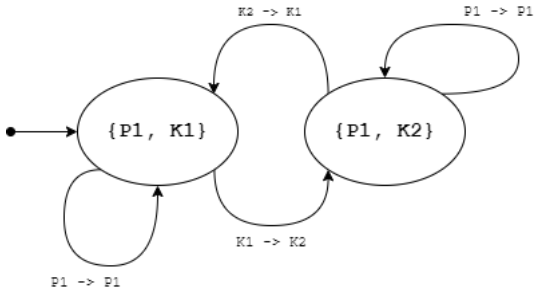
# 3 Przykłady

1. Przykład z diagramem stanów

| *Proces P* | | *Proces K* | |
|---|---|---|---|
| while true: | | while true: | |
| P1: | sekcja lokalna | K1: | sekcja lokalna |
| | | K2: | sekcja krytyczna |

Sekcja lokalna oraz sekcja krytyczna są zapisane skótowo; mogą w sobie zawierać wiele instrukcji. Px i Kx to numery instrukcji; dla uproszczenia główny while nie będzie numerowany.

System może być maksymalnie tylko w dwóch różnych stanach $\{P1, K1\}$ albo $\{P1, K2\}$. Algorytm jest poprawny ponieważ system spełnia warunki 1. 2. i 3. - widać to na diagramie stanów:




2. Przykład z tabelka

| *Proces P* | | *Proces K* | |
|---|---|---|---|
| while true: | | while true: | |
| P1: | sekcja lokalna | K1: | sekcja lokalna |
| P2: | sekcja krytyczna | K2: | sekcja krytyczna |

Algorytm nie spełnia warunku wzajemnego wykluczania ponieważ procesy mogą równocześnie znaleźć się w sekcjach krytycznych czyli w stanie $\{P2, K2\}$. Dowód:

| *Czas/Krok* | *Proces P* | *Proces K* | *Zmienne globalne* |
|---|---|---|---|
| T1 | P1 | K1 | brak |
| T2 | P1 | K2 | - |
| T3 | P1 | K1 | - |
| T4 | P1 | K2 | - |
| T5 | P2 | K2 | - |

# 4 Ćwiczenia

Dla poniższych algorytmów działających współbieżnie sprawdź czy działają poprawnie - czy spełniają warunki 1. 2. oraz 3. lub pokaż, że nie spełniają któregokolwiek z nich.

1. 

*Zmienne globalne*

dzialaj ← true

| *Proces P* | | *Proces K* | |
|---|---|---|---|
| | while true: | | while true: |
| P1: | sekcja lokalna | K1: | sekcja lokalna |
| P2: | dzialaj ← false | K2: | if dzialaj : |
| P3: | sekcja krytyczna | K3: | sekcja krytyczna |
| P4: | dzialaj ← true | | |

Oprócz numerów instrukcji przy opisie stanu systemu należy również uwzględnić zmienne globalne.

2. 

*Zmienne globalne*

dzialaj ← true

| *Proces P* | | *Proces K* | |
|---|---|---|---|
| | while true: | | while true: |
| P1: | sekcja lokalna | K1: | if dzialaj : |
| P2: | if dzialaj : | K2 | dzialaj ← false |
| P3: | dzialaj ← false | K3: | sekcja krytyczna |
| P4: | sekcja krytyczna | K4: | dzialaj ← true |
| P5: | dzialaj ← true | K5: | sekcja lokalna |

3.

| *Zmienne globalne* | |
|---|---|
| czyja_kolej ← ″P″ | |
| *Proces P* | *Proces K* |
| while true: | while true: |
| P1: sekcja lokalna | K1: sekcja lokalna |
| P2: await czyja_kolej = ″P″ | K2: await czyja_kolej = ″K″ |
| P3: sekcja krytyczna | K3: sekcja krytyczna |
| P4: czyja_kolej ← ″K″ | K4: czyja_kolej ← ″P″ |

Instrukcja await czeka aż warunek będzie prawdziwy.

4.

| *Zmienne globalne* | |
|---|---|
| turaP ← false | |
| turaK ← false | |
| *Proces P* | *Proces K* |
| while true: | while true: |
| P1: sekcja lokalna | K1: sekcja lokalna |
| P2: await turaK = false | K2: await turaP = false |
| P3: turaP ← true | K3: turaK ← true |
| P4: sekcja krytyczna | K4: sekcja krytyczna |
| P5: turaP ← false | K5: turaK ← false |

5.

| *Zmienne globalne* | |
|---|---|
| turaP ← false | |
| turaK ← false | |
| *Proces P* | *Proces K* |
| while true: | while true: |
| P1: sekcja lokalna | K1: sekcja lokalna |
| P2: turaP ← true | K2: turaK ← true |
| P3: await turaK = false | K3: await turaP = false |
| P4: sekcja krytyczna | K4: sekcja krytyczna |
| P5: turaP ← false | K5: turaK ← false |

6.

| *Zmienne globalne* | | | |
|---|---|---|---|
| turaP ← false | | | |
| turaK ← false | | | |
| *Proces P* | | *Proces K* | |
| | while true: | | while true: |
| P1: | sekcja lokalna | K1: | sekcja lokalna |
| P2: | turaP ← true | K2: | turaK ← true |
| P3: | while turaK : | K3: | while turaP : |
| P4: | turaP ← false | K4: | turaK ← false |
| P5: | turaP ← true | K5: | turaK ← true |
| P6: | sekcja krytyczna | K6: | sekcja krytyczna |
| P7: | turaP ← false | K7: | turaK ← false |

# Literatura

[1] Ben-Ari, Mordechai. Podstawy programowania współbieżnego i rozproszonego. Wydawnictwa Naukowo-Techniczne, 2009.

Uwagi: sluzalec@agh.edu.pl