Semafory

P (*passeren*) – wait
V (*vrijgeven, vrijmaken*) – signal

Edsger Dijkstra - Holender

Własności formalne operacji semaforowych

- Teoria aksjomatyczna Habermanna
- Dowodzenie poprawności algorytmów

Możliwość kontynuowania pracy przez dowolny proces przy wykonywaniu operacji *P* zależy od liczby wykonań operacji *P* i *V* w przeszłości oraz od wartości początkowej semafora.

C(s) – wartość początkowa
semafora ***s***, ***C(s)***>=0

ns(s)– liczba wykonań operacji
V (signal) na ***s***

nw(s) liczba wywołań operacji
P (wait) na ***s***

np(s) liczba przejść przez
operacje ***P*** (wait) na ***s***

(procesy kontynuowały pracę po P)

P(s):

```
nw(s):=nw(s) + 1;
if nw(s)<= C(s)+ns(s)
then np(s):=np(s)+1
```

Wykonanie operacji *P* nie pociąga za sobą zawieszenia procesu, o ile liczba wykonań tej operacji nie jest większa niż liczba wykonań operacji *V(ns)* zwiększona o wartość początkowa semafora

V(s):

if *nw(s) > C(s)+ns(s)*

then *np(s)*:=*np(s)*+1;

ns(s):=*ns(s) + 1*;

W wyniku działania operacji *V* jeden proces jest reaktywowany (o ile taki był)

Teoria procesów współbieżnych

TWIERDZENIE:

Efekt działania operacji P i V jest równoważny temu, że:

$$np(s)=min(nw(s),\ C(s)+ns(s))$$

jest niezmiennikiem wykonań operacji *P* i *V*.

Liczba zakończonych wykonań operacji *P* *(np-przejść)* jest nie większa niż liczba wywołań tej operacji, jak i nie większa niż suma wartości początkowej semafora oraz liczby wykonań operacji *V* *(ns)*.

Dowód indukcyjny:

$np(s)=min(nw(s),\ C(s)+ns(s))$ (1)

$np(s)= nw(s) < C(s)+ns(s)$ (2)

lub

$np(s)=C(s)+ns(s) < nw(s)$ (3)

lub

$np(s)= nw(s) = C(s)+ns(s)$ (4)

I) w stanie początkowym (1) wynika z definicji:

$np(s)= nw(s) = 0$ i $C(s)+ns(s)>=0$

II) (1) prawdziwe i wykonanie operacji $P(s)$

jeśli (2) to: $(np(s)++,\ nw(s)++)$

$$np(s) = nw(s) <= C(s) + ns(s)$$

jeśli (3) to: $(nw(s)++)$

$$np(s) = C(s) + ns(s) < nw(s)$$

jeśli (4) to: $(nw(s)++)$

(4)=>(3)

$$np(s) = C(s) + ns(s) < nw(s)$$

$P(s)$ – zachowuje niezmienniczość układu (2), (3), (4)

III) (1) prawdziwe i wykonanie operacji $V(s)$

jeśli (2) to: $(ns(s)++)$

$$np(s) = nw(s) < C(s) + ns(s)$$

jeśli (3) to: $(ns(s)++,\ np(s)++)$

$$np(s) = C(s) + ns(s) <= nw(s)$$

jeśli (4) to: $(ns(s)++)$

(4)=>(2)

$$np(s) = nw(s) < C(s) + ns(s)$$

$V(s)$ – zachowuje niezmienniczość układu (2), (3), (4)

Poprawność semaforowego rozwiązania problemu wykluczania wzajemnego

- Założenia:
  1. Niemożliwe jest inne wejście do SK niż przez wykonanie operacji *P*;
  2. Niemożliwe jest inne wyjście z SK niż przez wykonanie operacji *V*;
  3. Po wykonaniu operacji *P* na pewno będzie wykonana w skończonym czasie operacja *V* (SK jest skończona i nie prowadzi do zakleszczenia)

- Do udowodnienia:

1. W danej chwili tylko jeden proces może przebywać wewnątrz sekcji krytycznej;
2. Żaden proces nie może być zawieszany przy wejściu do sekcji krytycznej, jeśli nie ma w niej innego procesu.

(1)

np(s)= min(nw(s), C(s)+ns(s))

C(s)=1 => np(s)<=1+ns(s)   (*)

np(s) - l. procesów, które rozpoczęły wykonywanie SK

ns(s) - l. procesów, które opuściły SK

(2)

(a) Zawieszenie procesu => *np(s)< nw(s)*

z (*)   *np(s)=1+ns(s)*

(b) Pusta SK =>   *np(s)=ns(s)*

~((a)i(b))

Dowód poprawności rozwiązania problemu producent-konsument

```
var buf: array[1..N]of buffer
lp:integer:=1;
lk:integer:=1;
pełny, pusty, wp,wk:semaphore:=0,N,1,1;
```

```
procedure producent;
begin
repeat
produkowanie jednostki
wait (pusty)
wait (wp)
  fill(buf[lp]);lp:=lp mod N +1;
signal (wp);
signal (pełny);
end
end
```

```
procedure konsument;
begin
repeat
wait (pełny)
wait (wk)
   quit(buf[lk]);
   lk:=lk mod N +1;
signal(wk);
signal (pusty);
end
end
```

Założenia:

1. Operacje na buforze są dokonywane jedynie z procedur producent i konsument
2. O ile dowolny proces wykona pomyślnie obie operacje wait (*P*), to na pewno wykona również w skończonym czasie obie operacje signal(*V*)

Do udowodnienia:

1. Dowolne dwa procesy konsumenta i producenta nigdy nie będą współpracowały jednocześnie z tym samym polem buforowym (lp i lk nie będą wskazywały na to samo pole)
2. Nie wystąpi przepełnienie buforu kolejnymi wiadomościami ani pobieranie z pustych pól
3. Nie wystąpi zakleszczenie

(1)

Na buforze operacje mogą wykonywać co najwyżej 2 procesy (producent i konsument)

Po wait(pusty): //umieszczanie wiadomości w buforze

***ns(pełny) = np(pusty)-1 <= N +ns(pusty)*-1**

lp′=ns(pełny)+1 /ile razy zwiększano wartość *lp*

Po wait(pełny): //odbieranie wiadomości z bufora

np(pełny) = ns(pusty) + 1 <= ns(pełny)

lk′=np(pełny) /ile razy zwiększano wartość *lk*

Jeśli oba procesy jednocześnie współpracują z buforem:

np(pełny) <= ns(pełny)<=N +ns(pusty)-1<=*N* + *np(pełny)*-2

-np(pełny)
+1

0 <= *ns(pełny)*- *np(pełny)<= N*-2

1 <= lp′ - lk′ <= N -1

lp i *lk* nie wskazują na to samo pole buforowe

(2)

Aby nie było przepełnienia buforu (liczba wypełnień - liczba pobrań >N) ani niedomiaru (liczba pobrań> liczba wypełnień):

0 <= np(pusty)- np(pełny)<= N

np(pusty)—liczba wypełnień
np(pełny)-liczba pobrań

z producenta i twierdzenia:

ns(pełny) <= np(pusty)<= N +ns(pusty)

z konsumenta i twierdzenia:

ns(pusty) <= np(pełny)<= ns(pełny)

(3)

zakleszczenie:

- żaden producent nie przekazuje wiadomości:

$$np(pusty) = ns(pełny) \qquad (x)$$

- zawieszona dowolna liczba procesów producenta:

$$np(pusty) < nw(pusty) \qquad (y)$$

(x) i (y) => $\underline{ns(pełny)} = np(pusty) = \underline{\mathbf{N + ns(pusty)}}$ (o)

- żaden proces konsumenta nie współpracuje z buforem:

$$np(pełny) = ns(pusty) \qquad (xx)$$

- zawieszone procesy konsumenta

$$np(pełny) < nw(pełny) \qquad (yy)$$

(xx) i (yy) => $\underline{ns(pusty)} = np(pełny) = \underline{\mathbf{ns(pełny)}}$ (oo)

(o) i (oo) – wykluczają się