ZESZYTY NAUKOWE POLITECHNIKI ŚLĄSKIEJ 2002
Seria: AUTOMATYKA z. 135 Nr kol. 1555

Zbigniew BANASZAK
Uniwersytet Zielonogórski

ALGORYTMY WARIANTOWANIA PRZEBIEGU PROCESÓW WSPÓŁBIEŻNYCH

Streszczenie. Rozważany w pracy problem sterowania współbieżnie przebiegającymi procesami potokowymi współzawodniczącymi w dostępie do wspólnie wykorzystywanych zasobów systemu sprowadza się do wyznaczenia i przydziału reguł wyboru priorytetu do wspólnych zasobów oraz wyboru początkowej alokacji procesów takich, które gwarantują dopuszczalną (tzn. wykluczającą powstawanie zastojów) realizację wszystkich procesów. Dla tak sformułowanego problemu wyprowadzono warunki wiążące warianty alokacji reguł z daną strukturą rozdziału pojemności zasobów, warunki, których spełnienie gwarantuje poszukiwany, dopuszczalny przebieg procesów.

ALGORITHMS FOR CONCURRENT PRODUCTION FLOWS PROTOTYPING

Summary. The paper addresses control issues that are at the heart of concurrent processes competing to access to the common system resources. In particular, conditions sufficient for deadlock-free execution of a system of pipeline, repetitive manufacturing processes are considered. The relationship between system resource capacities and a set of assigned dispatching rules as well as assumed initial processes allocation is discussed. Finally algorithms for concurrent production flows prototyping are considered.

1. Wprowadzenie

Procesy występujące w różnych obszarach zastosowań, związanych np. z przepływem danych w sieciach komputerowych, przepływem pojazdów w ruchu miejskim, przepływami partii wyrobów w systemach wytwórczych, itp., łączy potrzeba ich współbieżnej realizacji. W sytuacji, gdy marszruty procesów przebiegają przez różne, wzajemnie nie przecinające się podzbiory zasobów systemu, wówczas wszystkie procesy składowe wykonują się równolegle,

nie oddziałując na siebie. Ich realizacje są limitowane jedynie przez przepustowości zasobów. W przypadku jednak, gdy marszruty poszczególnych procesów przechodzą przez pewne, wspólnie wykorzystywane zasoby systemu, wówczas w zależności od ich pojemności (rozumianej np. jako zdolności do jednoczesnej, niezależnej obsługi kilku procesów), przepływ procesów staje się uzależniony od wyboru zasad regulujących dostęp do tej właśnie kategorii zasobów.

Łatwo zauważyć, że przypadek, w którym pojemności zasobów wspólnych pozwalają na wzajemnie niezależne wykonywanie się procesów, odpowiada sytuacji organizacji równoległego przepływu procesów. W praktyce trudno jest jednak wskazać na system, którego pojemności zasobów nie są ograniczone. Z kolei, przyjęcie zasady niezsynchronizowanego dostępu procesów w sytuacji występowania ograniczeń zasobowych systemu prowadzi do powstawania zastojów – wzajemnej blokady procesów. Implikuje to potrzebę synchronizacji procesów konkurujących w dostępie do wspólnych zasobów. Limitowany dostęp do zasobów oznacza więc dodatkowe (w porównaniu z przepływem równoległym) ograniczenie realizacji procesów. Wynika stąd też, że przepustowość systemu o limitowanych pojemnościach zasobów wspólnych jest dodatkowo ograniczana (w porównaniu z przepływami równoległymi) przez zasadę regulującą dostęp procesów konkurujących do tych zasobów.

Warto również zauważyć, że przyjęcie reguły synchronizującej dostęp procesów pociąga za sobą, w ogólnym przypadku, potrzebę wyboru początkowej, dopuszczalnej alokacji procesów do zasobów. Nie dla każdej bowiem początkowej alokacji procesów przyjęta struktura reguł limitujących dostęp do zasobów zapewnia oczekiwany, nie prowadzący do zastojów, przebieg procesów. W szczególności oznacza to, że w systemie może być jednocześnie realizowana tylko ograniczona liczba procesów, liczba warunkowana wartością dopuszczalnego stanu początkowego, strukturą przyjętych reguł decyzyjnych oraz alokacją pojemności zasobowych systemu. Oznacza to także, że wprowadzenie stałych (nie zmieniających się w czasie) reguł synchronizacji procesów z jednej strony zapewnia pożądany przepływ procesów, z drugiej zaś zapewnia powtarzalność, cykliczność tego przepływu. Cykliczny charakter przepływu procesów wynika ze skończonej liczby dopuszczalnych alokacji procesów (skończonej przestrzeni stanów) oraz działania reguł synchronizacji – reguły te przyporządkowują poszczególnym stanom systemu zawsze te same stany. Mechanizmami synchronizującymi dostęp procesów mogą być również reguły wyboru

priorytetu, tzn. reguły determinujące kolejność dostępu procesów do wspólnie dzielonego zasobu.

W przedstawionym kontekście niniejsza praca stanowi kontynuację badań omawianych w [1], [2], [3], [4], [8], badań związanych z wyznaczaniem warunków wystarczających dla cyklicznego zachowania systemu współbieżnie przebiegających procesów potokowych. Wpisując się w nurt badań związanych z metodami unikania zastojów [5], [6], [7], wyprowadza warunki umożliwiające budowę procedur sterowania rozproszonego gwarantujących cykliczny, nie prowadzący do zastojów przebieg procesów potokowych. Warunki te pozwalają m.in. odpowiedzieć na pytania: Czy w systemie o znanym sposobie przydziału pojemności zasobów i znanej początkowej alokacji procesów możliwa jest taka ich realizacja, że przy zadanej strukturze reguł wyboru priorytetu nie dojdzie do zastoju realizowanych w nim procesów? Czy w systemie o znanym sposobie przydziału pojemności zasobów i znanej strukturze reguł wyboru priorytetu możliwa jest taka realizacja procesów, że dla zadanej początkowej alokacji procesów nie dojdzie do ich zastoju?

2. Sformułowanie problemu

Uprzedzając sformułowanie problemu rozważmy przykład ilustrujący związki łączące wcześniej wspomniane: strukturę przydziału reguł wyboru priorytetu, wybór stanu początkowego oraz dostępną alokację pojemności zasobowych systemu. Przyjmijmy następujące oznaczenia:

n – liczba procesów,

Z(n, m) – system realizujący n potokowych procesów współbieżnych, w którym liczbę kopii (elementów każdego procesu potokowego) określa wektor m, np.: dla Z(n,m), gdzie n = 2, m = (2, 1) - oznacza dwa procesy potokowe, przy czym w przypadku pierwszego z nich w systemie jednocześnie realizowane są dwie jego kopie, podczas gdy drugiego z nich tylko jedna,

q – liczba jednorodnych zasobów,

$R = \{r_1, r_2, \ldots, r_q\}$ – zbiór zasobów, R_s, R_u podzbiory zasobów, odpowiednio współdzielonych nie współdzielonych przez różne procesy, $R_s \subseteq R$, $R_u \subset R$, $R_s \cap R_u = \emptyset$, $R_s \cup R_u = R$,

$$\begin{array}{cccc} r_1, & r_2, & \ldots, & r_q. \end{array}$$

$S_l = (S_l(r_1), S_l(r_2), \ldots, S_l(r_q))$ – l-ty stan, $S_l(r_k)$ – k-ta współrzędna l-ego stanu,

$$S_i(r_k) = \begin{cases} p^i_j, \text{ jeżeli j-ta kopia i-tego procesu jest przydzielona do k-tego zasobu} \\ \Lambda, \text{ jeżeli żadna kopia procesu nie jest przydzielona do k-tego zasobu} \end{cases}$$

c_k - pojemność k-ego zasobu,

c - pojemność wszystkich zasobów; $c = \sum_{k=1}^{q} c_k$,

Jeżeli $\forall k \in \{1,2,\ldots,q\}$: $c_k = 1$, to $c = q$.

c_d - pojemność dodatkowa, $c_d = c_{d1} + c_{d2} + \ldots + c_{dq}$, c_{dk} – dodatkowa pojemność przydzielona do k-ego zasobu,

κ_k - liczba procesów konkurujących o dostęp do k-ego zasobu,

K^i - liczba zasobów wykorzystywanych przez i-ty proces (liczba zasobów marszruty i-tego procesu),

$\sigma_k = (p^1_1, p^2_2, \ldots, p^i_j, \ldots, p^n_1, \ldots, p^n_m)$ – reguła wyboru priorytetu określająca porządek, w którym kopie procesów $p^1_1, p^2_2, \ldots, p^i_j, \ldots, p^n_1 \ldots$ mogą korzystać z k-tego zasobu. W rozważanym przypadku, po m-tym dostępie n-tego procesu do k-tego zasobu, tzn. po wykonaniu się operacji kopii $p^n_{m(n)}$, gdzie m(n) jest liczbą kopii n-tego procesu, kolejny dostęp ma kopia p^1_1, następnie p^1_2, itd.

$P^i(d)$ – i-ty proces potokowy składający się z d kopii, d – liczba kopii jednocześnie realizowanych w systemie, każda kopia i-tego procesu potokowego przemieszcza się wzdłuż tej samej marszruty M^i.

$M^i = (r_{i1}, r_{i2}, \ldots, r_{iw})$ – marszruta i-tego procesu (sekwencja zasobów systemu, przez które przechodzi i-ty proces), $\mathcal{M}^i$ - zbiór zasobów występujących w i-tej marszrucie,

$H = (P^1, P^2, \ldots, P^n)$ - wektor hierarchii priorytetów określający porządek procesów, w którym procesy te występują w regułach wyboru priorytetu. Priorytet procesu P^i wyznacza pozycja zajmowanej przez niego współrzędnej. Przykładowo, przyjmijmy, że kopia p^1_1 przebiega przez zasoby r_1, r_3, r_{m-1}; p^2_1 przez r_2, r_3, r_4, r_m, oraz p^n_1 przez $r_1, r_2, r_3, r_{m-1}, r_m$, gdzie m jest łączną liczbą zasobów dzielonych. Jeżeli $H = (P^1, P^2, \ldots, P^n)$, wówczas $\sigma_1 = \{\ldots, p^1_1, p^n_1, \ldots\}$, $\sigma_2 = \{\ldots, p^2_1, p^n_1, \ldots\}$, $\sigma_3 = \{\ldots, p^1_1, p^2_1, p^2_1, \ldots\}$, $\sigma_4 = \{\ldots, p^2_1, \ldots\}$, $\sigma_{m-1} = \{\ldots, p^1_1, p^n_1, \ldots\}$, $\sigma_m = \{\ldots, p^2_1, p^n_1, \ldots\}$.

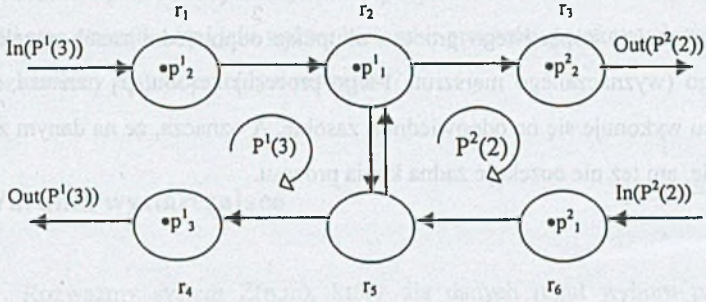
$L_i(r_a, r_b)$ - odcinek marszruty i-tego procesu potokowego łączący zasoby r_a i r_b, $r_a, r_b \in R_s$.

2.1. Przykład ilustracyjny

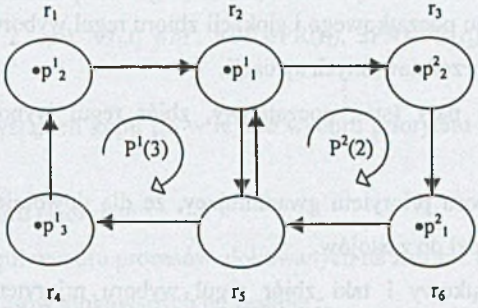
Dany jest system Z(2,(3,2)) zawierający dwa procesy potokowe $P^1(3)$, $P^2(2)$. $P^1(3)$ zawiera trzy kopie p^1_1, p^1_2, i p^1_3, a $P^2(2)$ dwie kopie p^2_1 , p^2_2. Tak więc łączna liczba kopii procesów wynosi p = 5. Niech $R_s = \{r_2, r_5\}$ i $R_u = \{r_1, r_3, r_4, r_6\}$, zatem q = 6. Niech $q_2 = 2$ i $q_1 = q_3 = q_4 = q_5 = q_6 = 1$, zatem c = 7. Ponadto, $\kappa_1 = 1$, $\kappa_2 = 2$, $\kappa_3 = 1$, $\kappa_4 = 1$, $\kappa_5 = 1$, $\kappa_6 = 1$ oraz $K^1 = 4$, $K^2 = 4$.

Strukturę marszrut procesów potokowych przedstawia rys. 1a). Poszukiwana jest para (stan początkowy, zbiór reguł wyboru priorytetu) gwarantująca cykliczny przebieg procesów współbieżnych o zadanej liczbie kopii. Model struktury odpowiedniego systemu procesów cyklicznych przedstawia rys.1 b).

a)



b)



Rys. 1. Przykład systemu dwóch procesów potokowych: struktura marszrut a), model procesów cyklicznych b)

Fig. 1. An example of two pipeline processes; the routes structure a), the model of pipeline cyclic processes (b)

Rozważmy stan początkowy $S_0 = (p^1_2, p^1_1, p^2_2, p^1_3, \Lambda, p^2_1)$ i zbiór reguł wyboru priorytetu $\{\sigma_2 = \{p^1_1, p^1_2, p^1_3, p^2_1, p^2_2\}, \sigma_5 = \{p^2_1, p^2_2, p^1_1, p^1_2, p^1_3\}\}$. Przyjęty stan prowadzi

do zastoju systemu, w którym kopia p^1_1 żąda dostępu do zasobu r_5, kopia p^1_2 żąda dostępu do zasobu r_2, z kolei kopia p^1_3 oczekuje na dostęp do r_1, podczas gdy p^2_1 oczekuje na r_2, a p^2_2 na r_5. Uzyskaną w ten sposób, zamkniętą pętlę wzajemnych oczekiwań ilustruje tablica 1.

Tablica 1

Sekwencja stanów osiągalnych ze stanu S_0

	S_0	S_1
r_1	p^1_2	p^1_2*
r_2	p^1_1	p^1_1*
r_3	p^2_2	p^2_2*
r_4	p^1_3	p^1_3*
r_5	Λ	p^2_1
r_6	p^2_1	Λ

p^i_j* oznacza, że j-ta kopia i-tego procesu okupując odpowiedni zasób oczekuje na dostęp do kolejnego (wyznaczanego marszrutą i-tego procesu) zasobu, p^i_j oznacza, że j-ta kopia i-tego procesu wykonuje się na odpowiednim zasobie, Λ oznacza, że na danym zasobie ani nie wykonuje się, ani też nie oczekuje żadna kopia procesu.

2.2. Problem

W rozważanej klasie systemów, przyjmując zadaną strukturę przydziału pojemności, w zależności od wyboru stanu początkowego i alokacji zbioru reguł wyboru priorytetu, mamy do czynienia z jedną z niżej przedstawionych sytuacji:

1. Dla dowolnie wybranej pary (stan początkowy, zbiór reguł wyboru priorytetu) nie dochodzi do zastojów.
2. Istnieje zbiór reguł wyboru priorytetu gwarantujący, że dla dowolnie wybranego stanu początkowego nie dochodzi do zastojów.
3. Istnieje taki stan początkowy i taki zbiór reguł wyboru priorytetu, dla których w przepływie nie dochodzi do zastojów.
4. Nie istnieje taka para (stan początkowy, zbiór reguł wyboru priorytetu), która gwarantowałaby, że w przepływie procesów nie dojdzie do zastoju.

Rozważany dalej problem inspirowany jest potrzebą pełnego wykorzystania dostępnych zasobów systemu. W szczególności sprowadza się on do wyznaczania takich sterowań specyfikowanych parami (stan początkowy, zbiór reguł wyboru priorytetu), które gwarantują brak występowania zastojów procesów. Łatwo zauważyć, że jego rozwiązanie

umożliwia rozwiązanie bardziej ogólnego problemu, problemu minimalizacji pojemności zasobowych systemu. W przypadku gdy znane są rozwiązania sprowadzające się do 2 lub 3 z wyżej wymienionych opcji, zawsze można próbować poszukiwać sterowania dla pomniejszonej o jeden sumarycznej pojemności zasobowej systemu. Warunek stopu poszukiwań wyznacza sytuacja odpowiadająca 4 z wcześniej wymienionych opcji.

Reasumując, rozważany w pracy problem sformułowany jest następująco: Dany jest system współbieżnie realizowanych procesów potokowych, tzn. znane są marszruty procesów oraz alokacje pojemności zasobów. Poszukiwane są warunki, które powinna spełniać para (stan początkowy, zbiór reguł wyboru priorytetu) dla zagwarantowania pozbawionego zastojów przepływu procesów. Problem ten dekomponuje się na dwa podproblemy:

1. Jaki zbiór reguł wyboru priorytetu i jaki wektor hierarchii procesów potokowych gwarantuje pozbawiony zastoju przepływ procesów?
2. Jaki stan początkowy i jaki wektor hierarchii procesów potokowych gwarantuje pozbawiony zastoju przepływ procesów?

3. Warunki wystarczające

Rozważmy system Z(n,m), który dla danych reguł wyboru priorytetu oraz stanu początkowego posiada przebieg cykliczny. Alokacja zbioru reguł wyboru priorytetu spełnia warunek: $\forall i \in \{1,2,...,n\}$, $\forall r_i, r_j \in M'$, $\forall PR \in PR(r_i)$, $\exists PR \in PR(r_j)$, $\forall p^j_d \in P(r_i)$: $F_i(p^j_d) = F_k(p^j_d)$, gdzie:

$F_i(p^j_d)$ – ilość wystąpień kopii p^j_d w regule wyboru priorytetu procesu (regule alokowanej na zasobie r_i),

PR - reguła wyboru priorytetu,

$PR(r_i)$ - zbiór reguł wyboru procesów alokowanych na zasobie r_i,

$P(r_i)$ - zbiór procesów alokowanych na zasobie r_i.

Twierdzenie

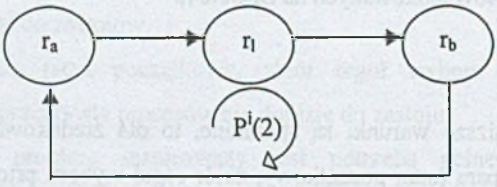
Jeżeli poniższe warunki są spełnione, to dla zredukowanej pojemności zasobowej systemu istnieje para (stan początkowy, zbiór reguł wyboru priorytetu) gwarantująca wolny od zastojów przepływ procesów.

(i) $\exists P^i(d)$, $i \in \{1, ..., n\}$, i $\exists r_a, r_b \in R_s$ pojawiające się w $L_k(r_a, r_b)$,

(ii) $L_k(r_a, r_b) : q_{ra} = 1 \wedge q_{rb} \geq k_{rb}$, i $H = (P^i(d), \ldots)$,

(iii) $\sigma_{ra} = \sigma_{rb} = \{[P^i(d)], [P^{i+1}(d)], \ldots, [P^m(d)], \ldots, [P^i(d)], [P^{i+1}(d)], \ldots, [P^m(d)]\}$, $\forall S_0 : S_0(r_b) = \Lambda$.

Dla dowodu twierdzenia należy zauważyć, że pojemność zasobu r_b jest wystarczająca dla alokacji wszystkich kopii procesu potokowego. Dzięki regułom wyboru priorytetu związanym z zasobami r_a, r_b oraz pozycji procesu potokowego $P^i(d)$ w H (tzn. jego hierarchii względem innych procesów), kopie tego procesu nie są blokowane przez kopie należące do innych procesów. Z warunku (iii) wynika, że stan początkowy nie może być przyjmowany w sposób arbitralny. Jedynie kopia procesu alokująca pierwszy z zasobów marszruty $L_i(r_a, r_b)$ (czyli albo r_a albo r_b) może zaalokować w następnym kroku drugi zasób (czyli odpowiednio albo r_b albo r_a). Co więcej, w marszrucie $L_i(r_a, r_b)$ zasób r_b nie może być zaalokowany w stanie początkowym. Ma to na celu uniknięcie zakleszczenia procesów poprzez zastój na zasobie r_a o jednostkowej pojemności (ma to szczególne znaczenie w przypadku procesów potokowych korzystających z tej samej marszruty $L_i(r_a, r_b)$, lecz przebiegających w przeciwnych kierunkach). Warto zauważyć, że jeżeli pomiędzy zasobami r_a i r_b występuje jeden lub więcej zasobów nie współdzielonych, to alokacje procesów na zasobach r_a oraz r_b (σ_{ra} i σ_{rb}) mogą być przesunięte wśród kopii danego procesu $P^i(d)$, tzn. kopie procesu w regule wyboru priorytetu mogą różnić się miejscami w tych regułach. Na przykład, zakładając, że zbiór kopii procesu $P^i(d)$ zawiera dwie kopie, które posiadają w swoich marszrutach zasoby r_a i r_b oraz zasób nie współdzielony r_l umiejscowiony miedzy r_a i r_b, a także przyjmując, że $S_0(r_a) = p^i_1$ i $S_0(r_l) = p^i_2$, wówczas $\sigma_{ra} = \{p^i_1, p^i_2, \ldots\}$ oraz $\sigma_{ra} = \{p^i_2, p^i_1, \ldots\}$. Pozwala to uniknąć zablokowania marszruty $L_i(r_a, r_b)$ (zob. rys. 2). Tak więc, kopie kolejnych procesów potokowych będą mogły zajmować zasoby, a cykl dostępu kolejnych kopii z procesów z występujących w regule wybory priorytetu powtarzał się będzie bez zakłóceń. W konsekwencji wszystkie kopie każdego procesu współdzielącego zasoby r_a , r_b wykonają pełny cykl.



Rys. 2. Przykład kopii procesu $P^i(d)$ posiadających w swoich marszrutach alokację na zasobie nie współdzielonym r_l umiejscowionym pomiędzy zasobami dzielonymi r_a i r_b

Fig. 2. An example of a copy of the pipeline process $P^i(d)$ following the route consisting the $r_l \in R_u$, placed among $r_a, r_b \in R_s$

4. Algorytmy syntezy procedur sterowania

Warunki przedstawionego twierdzenia pozwalają przedstawić algorytmy syntezy procedur sterowania przebiegiem współbieżnych procesów potokowych. Algorytmy te umożliwiają syntezę odpowiednio: zbioru reguł wyboru priorytetu, stanu początkowego. W szczególności, dla danego systemu Z(n,m), alokacji pojemności zasobów, stanu początkowego S_o, wektora H, oraz pary zasobów r_a, r_b, umożliwiają wyznaczenie zbioru reguł wyboru priorytetu bądź też dla danego zbioru reguł wyboru priorytetu $\{\sigma_{ra}, \sigma_{rb}, \ldots, \sigma_{rm}\}$, wektora H oraz pary zasobów r_a, r_b, umożliwiają wyznaczenie stanu początkowego S_o. Odpowiednie algorytmy mają następującą postać:

Algorytm 1

1. Jeżeli istnieje $L_i(r_a, r_b)$ należąca do M^i, $r_a, r_b \in R_s$, to przejdź do kroku 2, w przeciwnym wypadku przejdź do 9.
2. Niech $s = q'_{ra} + q'_{rb}$ (przed dokonaniem redukcji pojemności tych zasobów) oraz $q_{ra} = 1$ i $q_{rb} = k_{rb}$.
3. Jeżeli $q_{ra} + q_{rb} < s$, przejdź do kroku 4, w przeciwnym wypadku przejdź do 10.
4. Jeżeli $S_0(r_b) = \Lambda$, przejdź do kroku 6, w przeciwnym wypadku przejdź do 11.
5. Jeżeli $P^i(d)$ nie istnieje w H, dodaj go do H, w przeciwnym wypadku przejdź do kroku 6.
6. Dla każdego p^i_j wykonuj:

   Dodaj p^i_j do σ_{ra} i σ_{rb} biorąc pod uwagę kolejność, w jakiej p^i_j zajmuje r_a i r_b względem innych kopii procesu $P^i(d)$.
7. Dla każdego $P^k(d)$, którego kopie przechodzą przez zasób r_a i/lub r_b oraz $k \neq i$ wykonuj:

   Jeżeli $P^k(d)$ nie istnieje w H, dodaj go do H.

   Dla każdego p^k_l wykonuj:

   Dodaj p^k_l do σ_{ra} i σ_{rb} zgodnie z pozycją $P^k(d)$ w H.
8. Sukces. Przejdź do kroku 12.
9. Nie znaleziono marszruty. Przejdź do kroku 12.
10. Redukcja zasobów jest niemożliwa, przejdź do kroku 12.
11. Założone S0 jest niedopuszczalne.
12. Stop.

Algorytm 2

1. Jeżeli istnieje $L(r_a, r_b)$ należąca do M^i, $r_a, r_b \in R_s$, to przejdź do kroku 2, w przeciwnym wypadku przejdź do 8.
2. Niech $s = q'_{ra} + q'_{rb}$ (przed dokonaniem redukcji pojemności tych zasobów) oraz $q_{ra} = 1$ i $q_{rb} = k_{rb}$.
3. Jeżeli $q_{ra} + q_{rb} < s$ przejdź do kroku 4, w przeciwnym wypadku przejdź do 9.
4. Jeżeli $P^i(d)$ istnieje w H przejdź do kroku 6, w przeciwnym wypadku przejdź do 10.
5. Dla każdego p^i_j wykonuj:

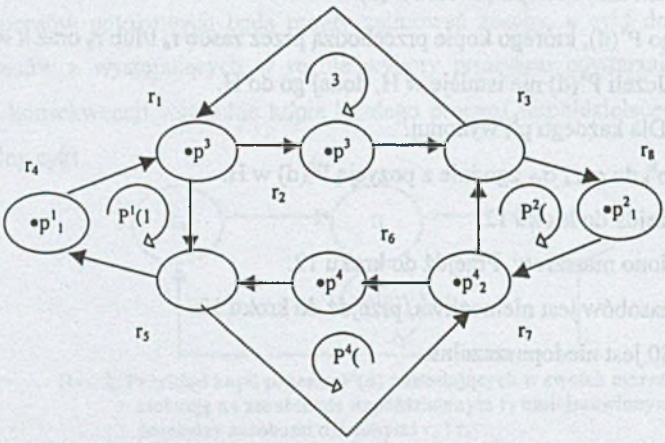
   Dodaj p^i_j do σ_{ra} i σ_{rb} biorąc pod uwagę kolejność w jakiej p^i_j zajmuje r_a i r_b względem innych kopii procesu $P^i(d)$.
6. Dla każdego $P^k(d)$, którego kopie przechodzą przez zasób r_a i/lub r_b oraz $k \neq i$ wykonuj:

   Jeżeli $P^k(d)$ nie istnieje w H, przejdź do kroku 10.

   Dla każdego p^k_l wykonuj:

   Jeżeli p^k_l nie występuje w σ_{ra} i σ_{rb} zgodnie z pozycją $P^k(d)$ w H, przejdź do kroku 11.
7. Niech $S_0(r_b) = \Lambda$. Sukces. Przejdź do kroku 12.
8. Nie znaleziono marszruty. Przejdź do kroku 12.
9. Redukcja zasobów jest niemożliwa, przejdź do kroku 12.
10. Niedopuszczalne H. Przejdź do kroku 12.
11. Niedopuszczalne σra i/lub σrb.
12. Stop.

W celu ilustracji działania algorytmu 1 rozważmy system, którego model przedstawia rys.3.



Rys. 3. Model systemu potokowych procesów cyklicznych

Fig. 3. A model of the system composed of pipeline cyclic processes

Załóżmy $S_0 = (p^3_2, p^3_1, \Lambda, p^1_1, \Lambda, p^4_1, p^4_2, p^2_1)$. Pojemności wszystkich zasobów, z wyjątkiem r_1, r_3, r_5 i r_7 równych 2, wynoszą 1. Należy wyznaczyć zbiór reguł wyboru priorytetu uniemożliwiający powstanie zastoju procesów współbieżnych.

Wykonajmy kroki podane w algorytmie 1:

- Niech $L(r_a, r_b)$ będzie marszrutą łącząca zasoby r_1 i r_3. Zatem $r_a = r_1$, $r_b = r_3$.
- Niech $s = q'_{r1} + q'_{r3}$, zatem $s = 4$.
- Niech $q_{r1} = 1$ oraz $q_{r3} = 2$.
- Ponieważ $s > q_{r1} + q_{r3}$, kontynuujemy wykonywanie algorytmu.
- Ponieważ $S(r_3) = \Lambda$, kontynuujemy wykonywanie algorytmu.
- Ponieważ H jest puste, dodajemy $P^3(2)$ do H, zatem $H = (P^3(2))$.
- Każdą kopię procesu $P^3(2)$ dodajemy do σ_{r1} i σ_{r3}, zatem $\sigma_{r1} = \{p^3_2, p^3_1\}$ oraz $\sigma_{r3} = \{p^3_1, p^3_2\}$.
- Ponieważ $P^1(1)$ i $P^2(1)$ nie istnieją w H, dodajemy je do H, zatem $H = (P^3(2), P^1(1), P^2(1))$.
- Procesy $P^1(1)$ i $P^2(1)$ dodajemy do σ_{r1} i σ_{r3}, zatem $\sigma_{r1} = \{p^3_2, p^3_1, p^1_1\}$ oraz $\sigma_{r3} = \{p^3_1, p^3_2, p^2_1\}$.

Procedurę tę powtarza się dla: $r_a = r_7$ oraz $r_b = r_5$:

- Niech $L(r_a, r_b)$ będzie marszrutą łącząca zasoby r_5 i r_7. Zatem $r_a = r_7$, $r_b = r_5$.
- Niech $s = q'_{r7} + q'_{r5}$, zatem $s = 4$.
- Niech $q_{r7} = 1$ oraz $q_{r5} = 2$.
- Ponieważ $s > q_{r7} + q_{r5}$, kontynuujemy wykonywanie algorytmu.
- Ponieważ $S(r_5) = \Lambda$, kontynuujemy wykonywanie algorytmu.
- Ponieważ H jest puste, dodajemy $P^4(2)$ do H, zatem $H = (P^4(2))$.
- Każdą kopię procesu $P^4(2)$ dodajemy do σ_{r7} i σ_{r5}, zatem $\sigma_{r7} = \{p^4_1, p^4_2\}$ oraz $\sigma_{r5} = \{p^4_1, p^4_2\}$.
- Ponieważ $P^1(1)$ i $P^2(1)$ nie istnieją w H, dodajemy je do H, zatem $H = (P^4(2), P^1(1), P^2(1))$.
- Procesy $P^1(1)$ i $P^2(1)$ dodajemy do σ_{r7} i σ_{r5}, zatem $\sigma_{r7} = \{p^4_1, p^4_2, p^2_1\}$ oraz $\sigma_{r5} = \{p^4_1, p^4_2, p^1_1\}$.

W rezultacie otrzymujemy następujący zbiór reguł wyboru priorytetów: $\{\sigma_{r1} = \{p^3_2, p^3_1, p^1_1\}, \sigma_{r3} = \{p^3_1, p^3_2, p^2_1\}, \sigma_{r5} = \{p^4_1, p^4_2, p^1_1\}, \sigma_{r7} = \{p^4_1, p^4_2, p^2_1\}\}$. Łatwo sprawdzić, że dla uzyskanej w ten sposób pary (stan początkowy, zbiór reguł wyboru priorytetu) przepływ

procesów ma charakter cykliczny i nie prowadzi do zastoju. Ilustrację stanów osiągalnych ze stanu S_0 przedstawia tablica 2.

Tablica 2

Sekwencja stanów osiągalnych ze stanu S_0

	S_0	S_1	S_2	S_3	S_4	S_5	S_6	S_7	S_8	S_9	S_{10}	S_{11}
r_1	p^3_2	p^3_2*	p^3_2*	Λ	Λ	p^3_1	p^3_1*	p^3_1*	Λ	Λ	p^1_1	p^1_1*
r_2	p^3_1	Λ	Λ	p^3_2	p^3_2*	p^3_2*	p^3_2*	Λ	P^3_1	p^3_1*	p^3_1*	p^3_1*
r_3	Λ	P^3_1	p^3_1*	p^3_1*	p^3_1*	Λ	Λ	p^3_2	P^3_2*	p^3_2*	p^3_2*	p^3_2*
r_4	p^1_1	p^1_1*	p^1_1*	p^1_1*	p^1_1*	p^1_1*	p^1_1*	p^1_1*	P^1_1*	p^1_1*	Λ	Λ
r_5	Λ	Λ	p^4_1	p^4_1*	p^4_1*	p^4_1*	Λ	p^4_2	P^4_2*	p^4_2*	p^4_2*	p^4_2*
r_6	p^4_1	p^4_1*	Λ	Λ	p^4_2	p^4_2*	p^4_2*	Λ	Λ	p^4_1	p^4_1*	p^4_1*
r_7	p^4_2	p^4_2*	p^4_2*	p^4_2*	Λ	Λ	p^4_1	p^4_1*	P^4_1*	Λ	Λ	p^2_1
r_8	p^2_1	p^2_1*	p^2_1*	p^2_1*	p^2_1*	p^2_1*	p^2_1*	p^2_1*	P^2_1*	p^2_1*	p^2_1*	Λ

	S_{12}	S_{13}	S_{14}	S_{15}	S_{16}	S_{17}	S_1	S_2	S_3	S_4	S_5	S_6
r_1	p^1_1*	Λ	p^3_2	p^3_2*	p^3_2*	p^3_2*	p^3_2*	p^3_2*	Λ	Λ	p^3_1	p^3_1*
r_2	p^3_1*	p^3_1*	p^3_1*	p^3_1*	p^3_1*	p^3_1*	Λ	Λ	p^3_2	p^3_2*	p^3_2*	p^3_2*
r_3	p^3_2* p^2_1	p^3_2* p^2_1*	Λ p^2_1*	Λ p^2_1*	Λ p^2_1*	Λ	p^3_1	p^3_1*	P^3_1*	p^3_1*	Λ	Λ
r_4	Λ	Λ	Λ	Λ	p^1_1	p^1_1*	p^1_1*	p^1_1*	P^1_1*	p^1_1*	p^1_1*	p^1_1*
r_5	p^4_2*	p^4_2* p^1_1	p^4_2* p^1_1*	p^1_1*	Λ	Λ	Λ	p^4_1	P^4_1*	p^4_1*	p^4_1*	Λ
r_6	p^4_1*	p^4_1*	p^4_1*	p^4_1*	p^4_1*	p^4_1*	p^4_1*	Λ	Λ	p^4_2	p^4_2*	p^4_2*
r_7	Λ	Λ	Λ	p^4_2	p^4_2*	p^4_2*	p^4_2*	p^4_2*	P^4_2*	Λ	Λ	p^4_1
r_8	Λ	Λ	Λ	Λ	Λ	p^2_1	p^2_1*	p^2_1*	P^2_1*	p^2_1*	p^2_1*	p^2_1*

W przedstawionej sekwencji alokacji procesów można wyróżnić przebieg przejściowy, wyznaczony przez jednostanową sekwencję S_0 oraz ustalony przebieg cykliczny, wyznaczony przez sekwencję: $S_1, S_2, \ldots, S_{17}$.

5. Zakończenie

Przedstawione warunki umożliwiają budowę procedur sterowania rozproszonego w systemach potokowych procesów współbieżnych. Procedury sterowania rozproszonego, tzn. pary (stan początkowy, zbiór reguł wyboru priorytetu), wyznaczane są w trybie iteracyjnym bądź to w oparciu o istniejącą strukturę systemu i arbitralnie zadany stan początkowy, bądź też na podstawie istniejącej struktury systemu i arbitralnie zadanego zbioru reguł wyboru

priorytetu. Istotną cechą zaproponowanych algorytmów wyznaczania procedur sterowania jest możliwość minimalizacji sumarycznej pojemności zasobowej systemu. Oznacza to, że rozważając system, którego struktura (tzn. alokacja pojemności zasobowych) gwarantuje, że dla dowolnie wybranej pary (stan początkowy, zbiór reguł wyboru priorytetu) nie dochodzi do zastojów [9], można wyznaczyć jeden z alternatywnych, gwarantujący oczekiwaną jakość przebiegu procesów, wariantów struktury systemu, wariant o zmniejszonej w porównaniu ze strukturą wyjściową, sumarycznej pojemności zasobów. Wyprowadzone warunki są warunkami wystarczającymi dla syntezy procedur sterowania. Implikuje to wielomianowy charakter złożoności obliczeniowej zaproponowanych algorytmów syntezy tych procedur.

LITERATURA

1. Banaszak Z. A., Tang X. Q., Wang S.C., Zaremba M.B.: Logistics models in flexible manufacturing. Computers in Industry, Vol.43, issue 3, 2000, 237-248.
2. Banaszak Z. A., Polak M.: Deadlock-free distributed control for repetitive flows. Proc. of the 6th Int. Workshop on Discrete Event Systems, October 2-4, 2002, Zaragoza, Spain (w druku).
3. Banaszak Z., Polak M.: Buffers vs. Initial State and Priority Rules Allocation. Proc. of the 8th International Symposium on „Methods and Models in Automation and Robotics", Kaszyński R. Ed., 2-5 September, 2002, Szczecin, Poland (w druku).
4. Banaszak Z. A.: Distributed Bottlenecks Control for Repetitive Production System, Journal of Intelligent Manufacturing (A special issue on Stochastic Algorithms in Manufacturing and Logistics), Vol.8, No.5, 1997, pp.415-424.
5. Brzeziński J.: Ocena stanu globalnego w systemach rozproszonych, Ośrodek Wydawnictw Naukowych PAN, Poznań 2001.
6. Majdzik P., Banaszak Z.: Procedura syntezy sterowania rozproszonego w systemach produkcji rytmicznej. Zeszyty Naukowe Politechniki Śląskiej, ser. Automatyka, z.124, nr kol. 1390, 1998, s. 99-109.
7. Lawley M. A., Reveliotis S.A., Ferreira P.M.: A correct and scalable deadlock avoidance policy for flexible manufacturing systems. IEEE Trans. on Robotics and Automation, Vol.14, No.5, 1998, pp.796-809.
8. Spiridon A. Reveliotis, Mark A. Laweley, Placid M. Ferreira: Polynomial-complexity deadlock avoidance policies for sequential resource allocation systems. IEEE Trans. on Automatic Control, Vol.42, No.10, 1997, pp.1344-1357
9. Zaremba M.B., Banaszak Z.A., Majdzik P., Jędrzejek K. J.: Distributed Flow control design for repetitive manufacturing processes. Proc. of the 14th Triennial Worlds Congress of IFAC, July 5-9, Beijing, P.R. China, 1999, Vol. A, pp.1-6.

Recenzent: Dr hab. inż. Mirosław Zaborowski, Prof. Pol. Śl.

Abstract

The paper addresses control issues that are at the heart of concurrent processes competing for access to a common set of system resources. In particular, conditions sufficient for deadlock-free execution of a system of pipeline, repetitive manufacturing processes are considered. The relationship between system resource capacities and a set of assigned dispatching rules as well as assumed initial processes allocation is discussed. Consequently, the considered control problem can be seen as a problem of defining of an initial state and a set of dispatching rules that locally constrain the way the concurrent workflows interact with each other, so as to guarantee a desired performance of a whole system. The methodology presented enable one to predict a system behavior, given buffer capacities allocation, system initial state, and protocols that govern the access to the shared resources (i.e., dispatching rules). Finally algorithms for concurrent production flows prototyping are considered. The results obtained can be treated as an extension of the author's earlier works, which were limited to the systems composed of a set of simple cyclic processes, to more general ones containing pipeline processes, i.e. the sets of copies of the same processes that follow each other on the same set of resources.