5. CONTROLUL PROCESELOR CONCURENTE

Obiective

Acest capitol este dedicat tratării separate a celor mai reprezentative aspecte legate de controlul şi gestiunea proceselor concurente. Tratarea este gradată, începând cu definiţii şi concepte de bază. Se pune un accent deosebit pe prezentarea unor algoritmi de excludere mutuală atât în context centralizat cât şi în context distribuit, analiza situaţiilor de blocare şi sincronizarea cu semafoare.

Problemele tratate în acest capitol asigură baza teoretică pentru analiza sau dezvoltarea unor sisteme de operare în regim de multiprogramare sau multiprelucrare, atât pentru sisteme monoprocesor, cât şi pentru sisteme cu prelucrare paralelă, de exemplu pentru sisteme multiprocesor.

5.1 Definiţii şi concepte de bază

Activităţile paralele se pot desfăşura atât în sisteme de tip MIMD, SIMD cât şi în cadrul sistemelor convenţionale, de tip SISD. Un exemplu de activităţi paralele într-un sistem de calcul convenţional, de tip Von Neumann îl constituie efectuarea simultană a operaţiilor din UCP, cu cele din subsistemul de intrări/ieşiri prin DMA sau canal de I/E. De asemenea, activităţi paralele se întîlnesc la nivelul microoperaţiilor care constituie o microinstrucţiune în cazul unei unităţi de comandă microprogramată.

În general pentru creşterea performanţelor unui sistem de calcul se prevăd mai multe procesoare de diferite tipuri, cum ar fi:

- procesoare centrale de prelucrare;
- procesoare de intrări/ieşiri;
- procesoare specializate (coprocesoare matematice, coprocesoare neurale).

Pe de altă parte mai multe programe sau părţi ale unui aceluiaşi program pot fi executate în paralel. Comunicaţia între acestea poate fi asigurată printr-un schimb de mesaje.

Chiar dacă există un singur procesor (nu o structură MIMD), care este partajat între mai multe programe, vom admite că, din punct de vedere logic aceste programe se execută în paralel disputându-şi accesul la diferite resurse fizice ale sistemului.

Proces secvenţial (task, sarcină) este o activitate care constă din execuţia, pe un procesor, a unui program cu un set de date specificat.

Vom numi astfel de programe, procese concurente.

Deşi pare că fiecare proces are procesorul şi datele sale, în realitate mai multe procese pot utiliza în comun un procesor, o secvenţă de cod sau o structură de date.

Un *proces* poate fi specificat prin relaţia sa cu exteriorul:

- intrările necesare;
- funcţia executată;
- ieşirile generate;
- starea la un moment dat;
- timpul de execuţie.

Comportarea intrinsecă, internă, nu va fi specificată decât în situaţii excepţionale, pentru a facilita înţelegerea interacţiunii cu celelalte procese.

Unui proces i se asociază două evenimente:

1°. $\overline{P}$ - iniţiere proces;

2°. $\underline{P}$ - terminare proces.

Dacă notăm cu $t(\cdot)$ timpul de apariţie a unui eveniment, vom considera că:

$t(\underline{P}) - t(\overline{P}) \neq 0$ şi finit

cu condiţia ca toate resursele necesare execuţiei lui P să fie disponibile în momentul t($\overline{P}$).

În continuare vom considera procesele *neinterpretate*, ceea ce este echivalent cu faptul că pentru o mulţime dată de procese să ne intereseze secvenţele evenimentelor de iniţiere şi terminare fără să ne preocupe particularităţile de execuţie ale proceselor.

În ceea ce priveşte granularitatea unui proces, aceasta poate fi:

- aplicaţie;
- program;
- modul de program;
- procedură;
- instrucţiune;
- microinstrucţiune;
- microoperaţie.

Sistemul de calcul în care se execută procesele constă dintr-o mulţime de resurse:

- procesoare;
- dispozitive de I/E;
- biblioteci de programe;
- proceduri;
- variabile;
- fişiere de date, etc.

şi este caracterizat de o mulţime Σ de stări.

În funcţie de granularitatea proceselor, procesorul poate fi:

- calculator universal;
- UCP într-un sistem multiprocesor;
- UAL;
- unitate de comandă;
- microsecvenţiator într-o unitate de comandă.

Atât posibilităţile sistemului, cât şi funcţiile proceselor date sunt reprezentate prin tranziţii permise în Σ, de forma $P_i \rightarrow P_j$, (P_i, P_j ε Σ) definite pentru iniţierea şi terminarea proceselor.

Iniţierea unui proces corespunde unei tranziţii de stări care reflectă:

- preluarea resurselor necesare;
- iniţializarea stării sistemului (a resurselor);
- citirea parametrilor de intrare.

Terminarea unui proces corespunde unei tranziţii de stări care reflectă:

- eliberarea sau suspendarea temporară a utilizării resurselor;
- scrierea parametrilor de ieşire;
- dacă valori ale parametrilor sau stări interne sunt asociate cu resursele, terminarea unui proces trebuie însoţită de valoarea stării resurselor.

Fie $\boldsymbol{P} = \{ P_1, P_2, ..., P_n \}$ o mulţime de procese iar < o relaţie de ordine parţială, nereflexivă, pe $\boldsymbol{P}$ (denumită şi relaţie de precedenţă pe mulţimea de procese $\boldsymbol{P}$).
Perechea $\mathbf{C} = (\boldsymbol{P}, <)$ este denumită *sistem de procese*.

Relaţia < reprezintă precedenţele în execuţia proceselor. Astfel, $P_I<P_j$ înseamnă că P_i trebuie terminată înainte de iniţierea lui P_j.

> Procesele din ***P*** sunt *independente* dacă pentru
> **C** = (***P*** , <) avem < = φ; unde φ este mulţimea vidă.

O reprezentare grafică convenabilă a unui sistem de procese se obţine utilizând grafuri aciclice direcţionate, *gad*. Procesele sunt reprezentate prin noduri iar relaţia de precedenţă prin mulţimea arcelor. Relaţia de precedenţă dată de mulţimea arcelor poate fi definită ca fiind cea mai mică relaţie în ***P*** a cărui închidere prin tranzitivitate este <. Deoarece < reprezintă o ordonare în timp a proceselor, vom spune că două procese P_i şi P_j, pot fi concurente dacă şi numai dacă ele sunt independente, adică între ele nu există o relaţie de precedenţă.

Exemplu:

Două procese P_i si P_j sunt *independente* dacă P_i nu este nici predecesor, nici succesor al lui P_j.

Un proces P din sistemul de procese **C** este pe *nivelul k*, sau are nivelul k, dacă cea mai lungă cale de la P la un nod terminal are lungimea k. Un nod terminal are nivelul 1.

> O *secvenţă de execuţie* a unui sistem de n procese, **C** = (***P***,<) este orice şir $\alpha = a_1\ a_2.....a_{2n}$ de evenimente de iniţiere şi terminare a proceselor cu respectarea relaţiilor de precedenţă impuse de <.

Altfel spus:

- pentru orice P ε ***P***, $\overline{P}$ şi $\underline{P}$ apar o singură dată în α ;
- dacă $a_i = \underline{P}$ şi $a_j = \underline{P}$, atunci i < j ;
- dacă $a_i = \underline{P}_k$ şi $a_j = \overline{P}_l$,cu $P_k<P_l$ atunci i < j ;

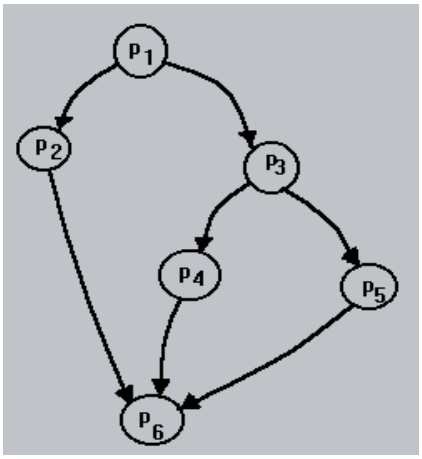

Figura 5.1. Exemplu de sisteme de sarcini reprezentate prin graf

Exemple de secvenţe de execuţie pentru sistemul de procese reprezentat prin graful din Figura 5.1.

$$\alpha_1 = \overline{P}_1 \underline{P}_1 \overline{P}_3 \underline{P}_3 \overline{P}_4 \underline{P}_4 \overline{P}_5 \underline{P}_5 \overline{P}_2 \underline{P}_2 \overline{P}_6 \underline{P}_6$$

$$\alpha_2 = \overline{P}_1 \underline{P}_1 \overline{P}_2 \overline{P}_3 \underline{P}_3 \overline{P}_4 \overline{P}_5 \underline{P}_5 \underline{P}_4 \underline{P}_2 \overline{P}_6 \underline{P}_6$$

O secvenţă de execuţie parţială este orice prefix al unei secvenţe de execuţie.

Mulţimea secvenţelor de execuţie reprezintă mulţimea tuturor secvenţelor de evenimente (de iniţiere şi terminare) care conduc la finalizarea lui **C** respectându-se relaţia <.

Un proces P este *activ* după o secvenţă de execuţie parţială $\alpha = a_1 a_2 .....a_k$, dacă există $i \leq k$ astfel că $a_i = \overline{P}$, dar pentru orice $j \leq k$, $a_j \neq \underline{P}$

Fie Σ spaţiul stărilor pentru un sistem de calcul.

Secvenţa stărilor corespunzătoare secvenţei de execuţie

$\alpha = a_1 a_2 .....a_{2n}$

este dată prin:

$\sigma = s_0 s_1 s_2 .....s_{2n}$, $s_j \varepsilon \Sigma$, $0 \leq j \leq 2n$, iar s_0 este starea iniţială dată.

Tranziţia stării definită de evenimentul a_i este reprezentată de:

$s_{i-1} \longrightarrow s_i$

O tranziţie pentru o pereche de stări-eveniment (s,a) va fi definită numai dacă evenimentul a poate să apară când sistemul este în starea s.

Un sistem de procese reprezentat printr-un graf cu un sigur nod iniţial şi un singur nod terminal se numeşte *închis*.

Dacă sistemul de n procese nu este închis se pot prevedea procesele P_0 şi P_{n+1} inoperante pentru a obţine un sistem închis.

Două sisteme de procese închise $\mathbf{C}_1$ şi $\mathbf{C}_2$ pot fi *concatenate* ($\mathbf{C}_1.\mathbf{C}_2$), graful asociat obţinându-se prin introducerea unui arc de la nodul terminal al lui $\mathbf{C}_1$ la nodul iniţial al lui $\mathbf{C}_2$.

O secvenţă de execuţie a lui ($\mathbf{C}_1.\mathbf{C}_2$) este orice şir

$\alpha = \alpha_1.\alpha_2$

Combinarea paralelă a două sisteme $\mathbf{C}_1$ şi $\mathbf{C}_2$, $\mathbf{C}_1 \mid \mathbf{C}_2$, care nu au procese în comun,($\forall P_i \in \mathbf{C}_1$ şi $\forall P_j \in \mathbf{C}_2$, $P_i \neq P_j$) constă în simpla lor reuniune.

de secvenţe de execuţie α_1 a lui $\mathbf{C}_1$, α_2 a lui $\mathbf{C}_2$.

Graful lui $\mathbf{C}_1 \mid \mathbf{C}_2$ are ca subgrafuri disjuncte grafurile lui $\mathbf{C}_1$ şi $\mathbf{C}_2$.

Orice proces din $\mathbf{C}_1$ este independent de orice proces din $\mathbf{C}_2$.
Dacă $a = a_1 a_2 .....a_{2m}$ şi $b = b_1 b_2 .....b_{2n}$ sunt secvenţe de execuţie pentru $\mathbf{C}_1$ respectiv pentru $\mathbf{C}_2$, o secvenţă de execuţie a lui $\mathbf{C}_1 \mid \mathbf{C}_2$ se formează astfel:

$c = c_1 c_2 .....c_{2[m+n]}$

unde :

1°. $c_1 = a_1 \mid b_1$ (fie a_1 fie b_1)

2°. Dacă $a_1 a_2 .....a_i$ şi $b_1 b_2 .....b_j$ sunt elemente ale lui $c_1 c_2 .....c_{i+j}$ cu $i+j<2(m+n)$ atunci :

$c_{i+j+1} = a_{i+1} \mid b_{j+1}$ (fie a_{i+1} fie b_{j+1})

Combinarea paralelă a secvenţelor de execuţie apare frecvent în proiectarea sistemelor, în special a sistemelor cu prelucrare paralelă, uneori sub o formă diferită de cea prezentată anterior în sensul că unele sisteme implică secvenţe repetitive.

Sistemele care conţin secvenţe repetitive pot fi modelate fie prin grafuri aciclice fie prin grafuri ciclice.

Astfel, se pot modela k cicli ai unui sistem **C** prin concatenarea

$$\mathbf{C}^k = \mathbf{C}_1 \cdot \mathbf{C}_2 \cdot \ldots \cdot \mathbf{C}_k$$

O secvenţă de execuţie α a lui $\mathbf{C}^k$ este de forma:

$$\alpha = \alpha_1 \alpha_2 \ldots . \alpha_k$$

unde α_i este o secvenţă de execuţie a lui **C** pentru iteraţia i.

Un sistem de procese se numeşte *ciclic* dacă este de forma $\mathbf{C}^k$ pentru $k > 1$, sau dacă este o combinare paralelă de sisteme închise în care cel puţin unul este ciclic.

5.2 Proprietatea de determinare a sistemelor de procese

Un sistem format din procese care se execută în paralel şi cooperează pentru realizarea unor operaţii logice şi de calcul, şi produce acelaşi rezultat indiferent *de durata* de execuţie a fiecărui proces independent, sau de ordinea în care acestea se execută, formează *un sistem de procese funcţional*, total asincron (independent de viteza de execuţie) sau *sistem determinat*.

Un sistem de procese este *nedeterminat* dacă rezultatele produse de procese independente depind de ordinea în care acestea se execută.

În Figura 5.2 se arată un exemplu de sistem nedeterminat:

P_1: $R_1 \leftarrow$ BUSFN (M; DCD (ADR))

P_2: M * DCD (ADR) $\leftarrow R_2$

P_1 - citeşte din locaţia de memorie de la adresa ADR;
P_2 - scrie în locaţia de memorie de la adresa ADR.

Nedeterminarea se poate rezolva introducând o relaţie de precedenţă adecvată între procesele care erau independente.

Pentru a stabili condiţiile necesare şi suficiente ca un sistem să fie determinat, vom considera un model simplificat în care sistemul fizic este privit ca o mulţime ordonată de locaţii de memorie:

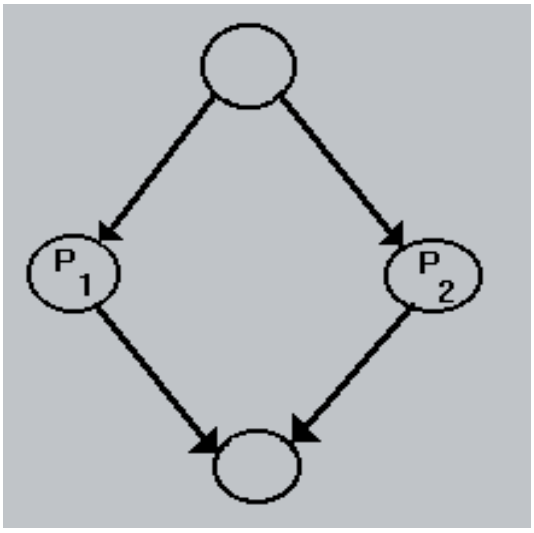

Figura 5.2. Exemplu de sistem nedeterminat

$\mathbf{M} = (M_1, M_2, ..., M_m)$

ce conţin orice valoare dintr-o mulţime de valori **V**.

Stările sistemului vor fi definite de valorile care se găsesc în memorie la un moment dat:

De exemplu dacă:

$\alpha = a_1 a_2 ..... a_{2n}$ şi

$\sigma = s_0 s_1 ..... s_{2n}$

reprezintă o secvenţă de execuţie, respectiv secvenţa de stări corespunzătoare,
iar $M_i[k]$ reprezintă valoarea din celula M_i imediat după evenimentul a_k; starea sistemului va fi:

$s_k = [M_1[k], M_2[k], ..., M_m[k]]$

Mulţimea tuturor stărilor poate fi definită astfel:

$\mathbf{\Sigma} = \mathbf{V}^m \qquad \mathbf{V}^m = \quad [\mathbf{V} \times \mathbf{V} \times ..... \times \mathbf{V}]$, produs cartezian.

Pentru a formaliza efectul execuţiei unui proces asupra celulelor de memorie vom considera că fiecărui proces P i se asociază funcţia:

$f_p : V^d \longrightarrow V^r$

unde:

$d = \left| D_p \right|$ cardinalul domeniului valorilor de intrare, D_p;
$r = \left| R_p \right|$ cardinalul domeniului valorilor de ieşire, R_p.

Pentru o stare iniţială s_0 şi o secvenţă de execuţie α, secvenţa corespunzătoare de stări:

$\sigma = s_0 s_1 ..... s_{2n}$ este definită în felul următor :

Fie $D_p = (M_{x1}, M_{x2}, ..., M_{xd})$ domeniul valorilor de intrare;
$R_p = (M_{y1}, M_{y2}, ..., M_{yr})$ domeniul valorilor de ieşire;

1°. dacă $a_{k+1} = \overline{P}$, atunci $M_i[k+1] = M_i[k], \quad 1 \le i \le m$;

2°. dacă $a_{k+1} = \underline{P}$, şi $a_l = \overline{P} \quad , l \le k$, atunci stările locaţiilor de memorie din domeniul de valori al lui P la momentul k+1 sunt:

$$[M_{y1}[k+1], M_{y2}[k+1], ..., M_{yr}[k+1]] = f_p[M_{x1}[l], M_{x2}[l], ..., M_{xd}[l]]$$

şi

$$M_i[k+1] = M_i[k] \quad \text{pentru} \quad (\forall)\, M_i \notin R_p$$

Altfel spus, dacă a_{k+1} este un eveniment de iniţiere nu apare o schimbare a stării, adică $s_{k+1} = s_k$.

Dacă însă a_{k+1} este un eveniment de terminare a lui P, $s_{k+1} \neq s_k$ doar în domeniul R**s**, noile valori din R**s** fiind determinate de f_p pe baza valorilor din domeniul de definiţie din momentul imediat precedent iniţierii lui P .

Secvenţa de stări $\sigma = s_0 s_1 ..... s_{2n}$ ce rezultă dintr-o secvenţă de execuţie, poate fi reprezentată sub forma unui tablou de dimensiuni m * (2n+1) având pe *linii* celulele de *memorie M*, iar pe *coloane stările*.

Exemplu:

Să considerăm un sistem format din două procese, iar relaţia de precedenţă fiind mulţimea vidă, aşa cum se arată în Figura 5.3.

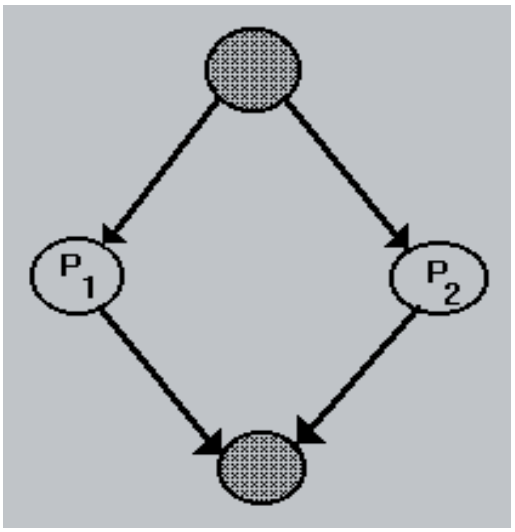

Figura 5.3. Sistem de sarcini nedeterminat

$\mathbf{C} = (\{P_1, P_2\}, \varphi)$

$\mathbf{M} = (M_1 , M_2)$

$D_{p1} = D_{p2} = R_{p1} = R_{p2} = \mathbf{M}$

Valorile iniţiale corespunzătoare stării iniţiale s_0:

$M \leftarrow (1,2)$

Sistemului i se atribuie două interpretări:

Procese	Interpretare 1	Interpretare 2
P_1	$(M_1,M_2) \leftarrow (1,M_2)$	$(M_1,M_2) \leftarrow (M_1+M_2,M_2)$
P_2	$(M_1,M_2) \leftarrow (2,M_2)$	$(M_1,M_2) \leftarrow (M_1,M_1+M_2)$

Fie două secvenţe de execuţie α_1 şi α_2

$\alpha_1 = \overline{P}_1 \underline{P}_1 \overline{P}_2 \underline{P}_2$ şi

$\alpha_2 = \overline{P}_2 \underline{P}_2 \overline{P}_1 \underline{P}_1$

Să analizăm comportarea celor două secvenţe de execuţie conform cu cele două interpretări ale sistemului:

	$\alpha_1 = \overline{P}_1 \underline{P}_1 \overline{P}_2 \underline{P}_2$	$\alpha_2 = \overline{P}_2 \underline{P}_2 \overline{P}_1 \underline{P}_1$
	$\sigma = s_0\ s_1\ s_2\ s_3\ s_4$	$\sigma = s_0\ s_1\ s_2\ s_3\ s_4$
Interpretare 1	M_1 1 1 1 1 2 M_2 2 2 2 2 2	M_1 1 1 2 2 1 M_2 2 2 2 2 2
Interpretare 2	M_1 1 1 3 3 3 M_2 2 2 2 2 5	M_1 1 1 1 1 4 M_2 2 2 3 3 3

Se observă că sistemul de procese **C** nu este determinat pentru nici una din cele două interpretări .

O reprezentare mai convenabilă constă din secvenţa de valori pe care un sistem de procese dat **C** o înscrie într-o celulă de memorie M_i, în timpul unei secvenţe de execuţie α .

Această reprezentare se notează sub formă vectorială:

$V(M_i,\alpha) = (V_1,V_2,.....,V_p)$

Considerând secvenţa de execuţie

$\alpha = a_1a_2.....a_{2n}$ şi
$\sigma = s_0s_1...s_{2n}$, secvenţa de stări corespunzătoare,
iar $\varepsilon=\varnothing$ şirul vid,
secvenţa de valori corespunzătoare se defineşte astfel:

$V(M_i,\varepsilon) = M_i(\varnothing)$ în lipsa unei secvenţe de execuţie, iar

$$V(M_i,a_1a_2\cdots a_k) = \begin{cases} \left(V(M_i,a_1a_2\cdots a_{k-1}),(M_i(k))\right) & \text{dacă } a_k = \underline{P}, \quad M_i \varepsilon R_p \\ \left(V(M_i,a_1a_2\cdots a_{k-1})\right) & \text{altfel} \end{cases}$$

Deci se iau în considerare numai valorile înscrise în celulele de memorie din R_p la terminarea proceselor din **C**.

Trebuie notat că $V(M_i,\alpha)$ şi $V(M_j,\alpha)$ nu trebuie să fie de aceeaşi lungime deoarece M_i poate fi în R pentru anumite procese iar M_j pentru altele (în număr diferit).

Pentru *secvenţa de valori*: $V(M_i,\alpha) = (v_1,v_2,.....,v_p)$ se defineşte *valoarea finală*

$F(M_i,\alpha) = v_p$.

Rezultă că pentru secvenţa de execuţie
$\alpha = a_1a_2...a_k$
$s_k = [F(M_1,\alpha),F(M_2,\alpha),...,F(M_m,\alpha)]$

Exemplu: Să reconsiderăm exemplul precedent:
Secvenţa de stări arătată mai sus, poate fi reprezentată astfel printr-un tablou:

	$\alpha_1 = \overline{P}_1\underline{P}_1\overline{P}_2\underline{P}_2$			$\alpha_2 = \overline{P}_2\underline{P}_2\overline{P}_1\underline{P}_1$		
	$V(M,\varepsilon)$	$V(M,a_1a_2)$	$V(M,a_1..a_4)$	$V(M,\varepsilon)$	$V(M,a_1a_2)$	$V(M,a_1..a_4)$
Interpretare 1	M_1 1	1	2	M_1 1	2	1
	M_2 2	2	2	M_2 2	2	2
Interpretare 2	M_1 1	3	3	M_1 1	1	4
	M_2 2	2	5	M_2 2	3	3

Un sistem de procese **C** = (***P***,<) este *neinterpretat* dacă se cunosc doar < ,D şi R pentru fiecare proces. O *interpretare* pentru **C** constă din specificarea funcţiei f_p pentru fiecare $P_i \varepsilon \boldsymbol{P}$.

În continuare vom considera sisteme de procese neinterpretate.

Definiţie: Un *sistem de procese **C** este determinat* dacă pentru orice stare iniţială s_0 dată, $V(M_i,\alpha)=V(M_i,\alpha')$, $1 \le i \le m$, pentru toate secvenţele de execuţie α ,α' din **C**.
(Secvenţele de valori depind numai de valorile iniţiale în s_0.)

În exemplul anterior, interpretarea 1 este nedeterminată deoarece P_1 şi P_2 sunt în dispută

Definiţie: Procesele P şi P' sunt *neinterferente* dacă:
- P este succesor sau predecesor a lui P', sau
- $R_p \cap R_{p'} = R_p \cap D_{p'} = D_p \cap R_{p'} = \varnothing$

pentru a scrie în M_1, iar în interpretarea 2 un proces scrie într-o celulă citită de cealalt.

Mulţimea $\boldsymbol{P} = \{ P_1,P_2,...P_n \}$ se spune că este formată din procese *mutual neinterferente* dacă pentru $\forall$ i,j ($i \neq j$) P_i si P_j sunt neinterferente. Pentru a arăta că un sistem de procese mutual neinterferente este determinat vom prezenta *teorema de suficienţă şi necesitate.* Înainte de a prezenta această teoremă vom stabili o lemă utilizată în cadrul teoremei de necesitate şi suficienţă.

Demonstraţie :

Deoarece P nu are succesori în **C** conform cu < , α' satisface relaţiile de precedenţă din **C** şi deci trebuie să fie o secvenţă de execuţie validă.

Deoarece P scrie numai în celulele ce aparţin lui R_p şi deoarece pentru orice P' iniţiată în β_3, după terminarea lui P în α, $R_p \cap D_{p'} = \varnothing$ fiind neinterferente, orice astfel de procese P' găsesc aceleaşi valori în $D_{p'}$ atât pentru α cât şi pentru α'.

Lemă: Fie **C** un sistem de n procese mutual neinterferente iar P un proces terminal al lui **C**.

Dacă $\alpha = \beta_1 \overline{P} \beta_2 \underline{P} \beta_3$ este o secvenţă de execuţie validă a lui **C**, atunci

$\alpha' = \beta\beta_2\beta_3 \overline{P}\ \underline{P}$ este deasemenea o secvenţă de execuţie a lui **C** pentru care s_0 dată,

$V(M_i,\alpha)=V(M_i,\alpha')$, $\forall\ 1 \le i \le m$

Deci pentru $M_i \notin R_p$ rezultă $V(M_i,\alpha)=V(M_i,\alpha')$

Pentru orice $M_j \varepsilon D_p$, $V(M_j,\beta_1)=V(M_j,\beta_1\beta_2\beta_3)$ deoarece nici un proces P' din $\beta_1\beta_2\beta_3$ nu scrie în D_p deci $R_{p'} \cap D_p = \varnothing$) fiind mutual neinterferente. Stării $F(M_j,\beta_1) = F(M_j,\beta_1\beta_2\beta_3)$ pentru $\forall M_j \varepsilon D_p$, iar P scrie aceeaşi valoare în $\forall$ $M_i \varepsilon R_p$ atât pentru α cât şi pentru α'.

Considerând că v este valoarea înscrisă de P în $M_i \varepsilon R_p$ pentru α, rezultă că:

$V(M_i,\alpha) = V(M_i,\beta_1 \overline{P} \beta_2 \underline{P})$ nu există P' ε β_3 care scrie în R_p

$V(M_i,\alpha) = (V(M_i,\beta_1 \overline{P} \beta_2),v)$ P scrie valoarea v în M_i

$V(M_i,\alpha) = (V(M_i,\beta_1),v)$ nu există P'ε β_2 care scrie în R_p

$V(M_i,\alpha) = (V(M_i,\beta_1\beta_2\beta_3),v)$ nu există P'ε$\beta_2\beta_3$ care scrie în R_p

$V(M_i,\alpha) = (V(M_i,\beta_1\beta_2\beta_3 \overline{P}\ \underline{P})$ P scrie v în M_i

$V(M_i,\alpha) = V(M_i,\alpha')$

Teorema de suficienţă:
Sistemele de procese formate din procese mutual neinterferente sunt determinate.

Demonstraţie: (Prin inducţie)

Pas 1: Pentru un sistem de procese format dintr-un singur proces este evident.

Pas 2: Presupunem că afirmaţia este adevărată pentru sisteme cu mai puţin de n procese.

Pas 3: Presupunem sistemul **C** = (***P***,<) format din n procese.

Dacă sistemul **C** are o singură secvenţă de execuţie validă, el este determinat.

Presupunem 2 secvenţe de execuţie α_1, α_2 în **C** şi P proces terminal în ***P***.

Conform lemei putem forma 2 secvenţe de execuţie α'_1 şi α'_2 astfel:

$\alpha'_1 = \alpha'_1 \overline{P}\ \underline{P}$; $\alpha'_2 = \alpha'_2 \overline{P}\ \underline{P}$,

în condiţiile în care

$V(M_i, \alpha_1) = V(M_i, \alpha'_1)$, $1 \le i \le m$

$V(M_i, \alpha_2) = V(M_i, \alpha'_2)$, $1 \le i \le m$

α'_1 şi α'_2 sunt secvenţe de execuţie pentru un sistem cu n-1 procese: **C**' = (***P***\P, <'), relaţia de precedenţă <' fiind obţinută din < eliminând relaţia de precedenţă ce implică pe P.

$V(M_i, \alpha'_1) = V(M_i, \alpha'_2)$ din ipoteza de inducţie.

Deci se poate presupune că valorile din D_p sunt aceleaşi, atât pentru α'_1 cât şi pentru α'_2, respectiv $F(M_i, \alpha'_1) = F(M_i, \alpha'_2)$ pentru $M_i \in D_p$. Rezultă astfel că pentru α'_1 şi α'_2, procesul P scrie aceeaşi valoare V pentru orice celulă $M_i \in R_p$.

Pentru $\mathbf{M_i \notin R_p}$:

$V(M_i, \alpha_1)$ = Lemă

$= V(M_i, \alpha'_1)$ = $M_i \notin R_p$

$= V(M_i, \alpha'_1)$ = ip. de inducţie

$= V(M_i, \alpha'_2)$ = $M_i \notin R_p$

$= V(M_i, \alpha'_2) =$ Lemă

$= V(M_i, \alpha_2)$

Am arătat că pentru $M_i \notin R_p$ $V(M_i, \alpha_1) = V(M_i, \alpha_2)$

Pentru $\mathbf{M_i \in R_p}$:

$V(M_i, \alpha_1) =$ Lemă

$= V(M_i, \alpha'_1) =$

$= (V(M_i, \alpha'_1), v) =$ ip. de inducţie

$= (V(M_i, \alpha'_2), v) =$

$= V(M_i, \alpha'_2) =$ Lemă

$= V(M_i, \alpha_2)$

Teorema de necesitate:
Fie **C** un sistem de procese astfel că pentru fiecare $P \in \boldsymbol{P}$ f_p nu este specificată, dar D_p şi $R_p \neq \varnothing$ sunt date.
C este determinat pentru orice interpretare a proceselor sale numai dacă acestea sunt mutual neinterferente.

Am arătat astfel că sistemul **C** este determinat dacă procesele sunt mutual neinterferente.

Demonstraţie:

Presupunem că P, P' sunt interferente, independente (adică nu există nici o relaţie de ordine între ele). Atunci există două secvenţe de execuţie valide:

$\alpha = \beta_1 \overline{P}\ \underline{P}\ \overline{P}'\underline{P}'\beta_2$ şi

$\alpha' = \beta_1 \overline{P}'\underline{P}'\overline{P}\ \underline{P}\ \beta_2$

Presupunem că există o celulă de memorie $M_i \in R_p \cap R_{p'}$ şi putem alege f_p şi $f_{p'}$ (două interpretări, pentru P şi P'), astfel încât pentru:

f_p: P să scrie în M_i valoarea u;

$u \neq v$

f_p: P' să scrie în M_i valoarea v;

În ceea ce priveşte valorile din R_p şi $R_{p'}$ pentru secvenţele α şi α' putem spune că:

$V(M_i, \alpha) = V(M_i, \beta_1 \overline{P}\ \underline{P}\ \overline{P}'\underline{P}') = (V(M_i, \beta_1), u, v);$

$u \neq v$

$V(M_i, \alpha') = V(M_i, \beta_1 \overline{P}'\underline{P}'\overline{P}\ \underline{P}) = (V(M_i, \beta_1), v, u);$

Deci în celula M_i, secvenţele α si α' înscriu secvenţe de valori diferite ceea ce înseamnă că sistemul de procese **C** nu este determinat.

Deci este necesar ca $R_p \cap R_{p'} = \varnothing$ pentru că altfel rezultă că **C** este nedeterminat.

Presupunem că există o celulă $M_j \in D_p \cap R_{p'}$.

Presupunem că există o celulă $M_i \in R_p$.

Atunci, putem alege o interpretare a lui P', o funcţie $f_{p'}$ astfel încât $F(M_j, \beta_1) \neq (M_j, \beta_1 \overline{P}'\underline{P}')$. Rezultă în acest caz că P citeşte valori diferite pentru α şi α' (având în vedere că $M_j \in D_p \cap R_{p'}$).

Putem alege în acest caz o funcţie f_p astfel încât procesul P să scrie în M_i, valoarea u pentru secvenţa α şi valoarea v pentru secvenţe α', cu $u \neq v$.

În acest caz putem spune că:

$V(M_i, \alpha) = V(M_i, \beta_1 \overline{P}\ \underline{P}\ \overline{P}'\underline{P}') =$

$= V(M_i, \beta_1 \overline{P}\ \underline{P}) = R_p \cap R_{p'} = \varnothing$

$= (V(M_i, \beta_1), u)$ P scrie u pentru α

$V(M_i, \alpha') = V(M_i, \beta_1 \overline{P}'\underline{P}'\overline{P}\ \underline{P}) =$

$= (V(M_i, \beta_1 \overline{P}'\underline{P}'), v) =$

$= (V(M_i, \beta_1), v)$ P scrie v pentru α'

deci $V(M_i, \alpha) \neq V(M_i, \alpha')$ deci **C** nu este determinat.

Rezultă că: $D_p \cap R_{p'} = \varnothing$

Similar se poate arăta că $D_{p'} \cap R_p = \varnothing$

O aplicaţie foarte importantă a teoremelor arătate anterior se referă la *paralelismul maxim* în sisteme de procese în care constrângerile de precedenţă sunt impuse *numai de cerinţele de determinare*.

Din definiţia dată pentru determinare rezultă că fiecare proces produce o singură secvenţă de valori pentru o celulă de memorie M_i în condiţiile unei stări iniţiale date.

Două sisteme cu aceeaşi mulţime de procese P sunt *echivalente* dacă sunt determinate şi dacă pentru aceeaşi stare iniţială produc aceeaşi secvenţă de valori.

Un sistem de procese **C** şi graful asociat G implică un *paralelism maxim* dacă **C** este determinat şi dacă eliminarea unui arc oarecare (P, P') din G, face ca procesele P şi P' să devină interferente.

Astfel dacă (P , P') este un arc într-un graf cu maximum de paralelism atunci:

$(R_p \cap R_{p'}) \cup (R_p \cap D_{p'}) \cup (D_p \cap R_{p'}) \neq \varnothing$

Având un sistem de procese **C**, determinat, este util să construim sistemul echivalent **C'** cu maximum de paralelism.

Teoremă :
Dacă plecând de la un sistem de procese **C**= (***P***, <) se construieşte **C'** = (***P***, <') unde relaţia <' este închiderea prin tranzitivitate a relaţiei:

$Z=\{(P,P') \varepsilon < \mid (R_p \cap R_{p'}) \cup (R_p \cap D_{p'}) \cup (D_p \cap R_{p'}) \neq \varnothing\}$

atunci **C'** este unicul sistem echivalent cu **C** care implică maximum de paralelism.

Exemplu :
Fie sistemul de procese $\mathbf{C} = (\boldsymbol{P}, <)$ unde
$\boldsymbol{P} = \{P_1,P_2,P_3,P_4,P_5,P_6,P_7,P_8\}$ iar relaţia de precedenţă < specificată de graful G, prezentat în Figura 5.4.

Să se construiască sistemul de procese $\mathbf{C}'=(\boldsymbol{P}',<')$ echivalent, care realizează maxim de paralelism, având în vedere că sistemul este caracterizat prin:

$M = \{ M_1,M_2,M_3,M_4,M_5 \}$ iar domeniile D_p şi R_p sunt specificate prin tabelul :

M	Domeniu de definiţie pentru procesele:	Domeniu de valori pentru procesele:
M1	1,2,7,8	3
M2	1,7	5
M3	3,4,8	1
M4	3,4,5,7	2,7
M5	6	4,6,8

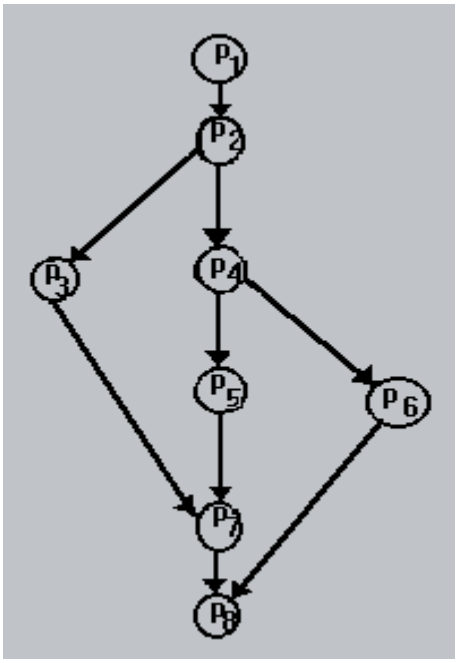

Figura 5.4. Sistem de sarcini

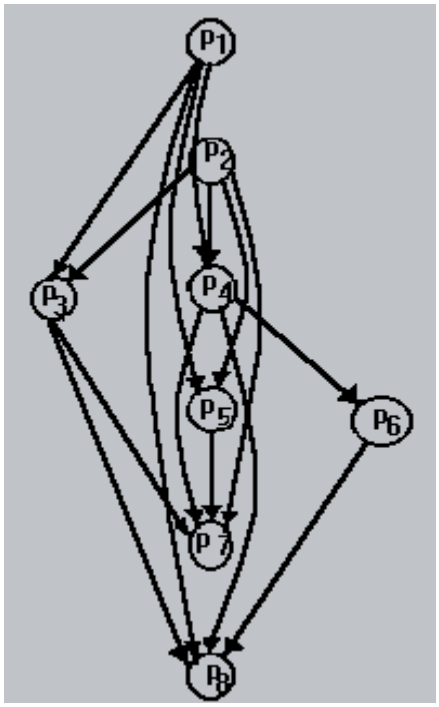

Figura 5.5 a) Graful G

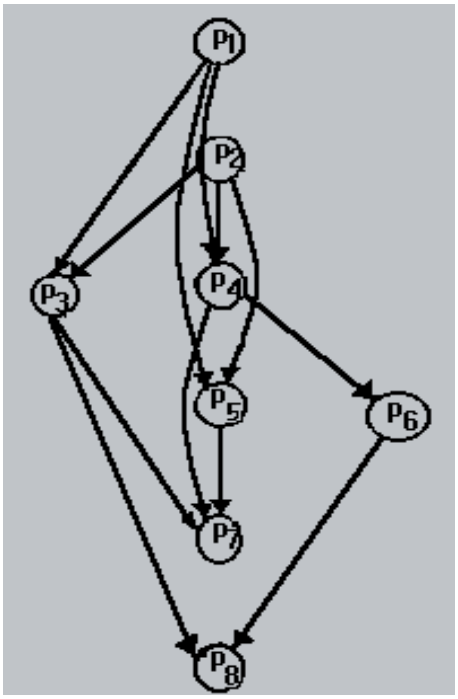

Figura 5.5 b) Graful G'

Relaţia Z, conform definiţiei de mai sus va fi dată de mulţimea arcelor:

Z : { (1,3) (1,4) (1,5) (1,8)
(2,3) (2,4) (2,5) (2,7)
(3,7) (3,8)
(4,6) (4,7) (4,8)
(5,7)
(6,8) }

Rezultă graful asociat G, prezentat în Figura 5.5.a. care organizat pe niveluri şi eliminând arcele excluse prin tranzitivitate rezultă graful G' al sistemului echivalent **C**' prezentat în Figura 5.5.b şi Figura 5.5.c.

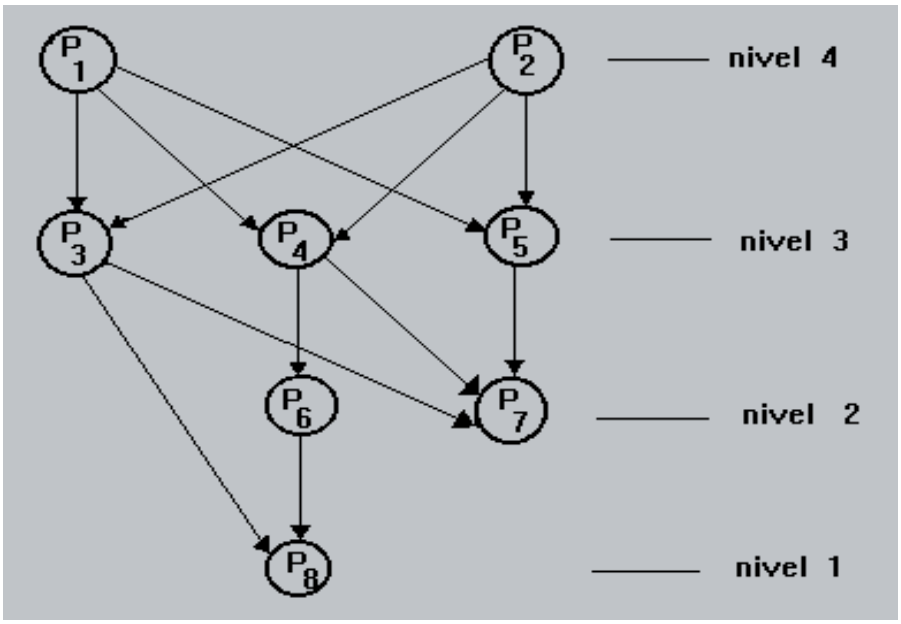

Figura 5.5.c) Graful G' rearanjat

Sistemul **C'** echivalent se execută în patru perioade faţă de 6 perioade cât era necesar pentru sistemul **C**, considerând că avem un număr suficient de procesoare.