Sisteme de operare
Interblocari (Deadlocks)

Prof. univ. dr. Constantin POPESCU
Departamentul de Matematica si Informatica,
Universitatea din Oradea, Romania

Agenda

- Resurse
- Obtinerea accesului la resurse
- Conditii pentru interblocare
- Modelarea interblocarilor

Introducere

- Resursele pot fi folosite doar de un singur proces la un moment dat
- Exemple de resurse ale unui calculator:
  - Imprimanta
  - Banda magnetica
  - Unitate de inscriptionare a CD-ului
  - Memoria
  - Intrari in tabelele interne ale sistemului
  - Inregistrari ale bazelor de date

Introducere

- Sistemul de operare permite unui proces accesul exclusiv la anumite resurse
- Cand doua sau mai multe procese sunt blocate spunem ca apare o interblocare (deadlock)
- Interblocarile pot aparea pe resursele hard sau pe resursele soft
- Exemplu:
  - Avem un sistem de baze de date
  - Procesul A blocheaza inregistrarea R1
  - Procesul B blocheaza inregistrarea R2
  - Apoi fiecare proces incearca sa blocheze inregistrarea celuilalt
  - Apare astfel o inteblocare

Resurse

- O resursa poate fi folosita doar de un singur proces in orice moment de timp
- Resursele sunt de doua tipuri:
  - Preemptibile
  - Non-preemptibile
- O resursa preemptibila este o resursa care poate fi luata de la procesul care o detine fara efecte nedorite
- Memoria este un exemplu de resursa preemptibila

Resurse preemptibile

- Exemplu:
  - Consideram un sistem cu 32 MB de memorie, o imprimanta si doua procese de 32 MB
  - Fiecare proces doreste sa tipareasca ceva
  - Procesul A cere si primeste acces la imprimanta
  - Inainte de a termina prelucrarea, i se termina cuanta de timp si este scos din memorie si pus pe disc (swapped out)
  - Procesul B ruleaza acum si incearca fara succes sa acceseze imprimanta
  - Avem o situatie de interblocare potentiala

Resurse preemptibile

- A aparut o interblocare potentiala deoarece A are imprimanta, iar B are memoria
- Si nici unul nu poate continua fara resursa detinuta de celalalt
- Solutia: memoria poate fi preemptata de la B prin punerea acestuia pe disc si prin aducerea lui A in locul său (swapped in)
- Acum procesul A poate rula, tipari si apoi elibera imprimanta
- Nu apare astfel nici o interblocare

Resurse non-preemptibile

- Este o resursa ce nu poate fi luata de la detinatorul ei fara ca executia acestuia sa esueze
- Exemplu: unitatile de inscriptionare CD-uri
- In general, interblocarile implica resurse non-preemptibile
- Interblocarile care implica resurse preemptibile pot fi rezolvate de obicei prin realocarea resurselor de la un proces la altul
- Secventa de evenimente pentru folosirea unei resurse:
  - Cerere resursa
  - Folosire resursa
  - Eliberare resursa

Resurse non-preemptibile

- Daca resursa nu este disponibila cand este ceruta
- Procesul care doreste acces la resursa este fortat sa astepte
- Un proces pentru care cererea de acces la resursa a esuat va sta intr-o bucla de cerere a resursei
- Acest proces nu este blocat, practic este ca si blocat
- Nu poate face lucruri utile
- Resursele sunt cunoscute sistemului de operare sub forma unor fisiere speciale
- Si doar un proces le deschide la un moment dat
- Acestea sunt deschise prin apelul de sistem *open*
- Daca fisierul este deja deschis, apelantul este blocat pana cand detinatorul actual il va debloca

Introducere in interblocari

- Definitia formala:

O multime de procese se afla in interblocare daca fiecare proces din multime asteapta un eveniment care poate fi furnizat numai de alt proces din multime

- Evenimentul pe care il asteapta fiecare proces este eliberarea unor resurse
- Care in prezent sunt detinute de un alt membru al multimii
- Nici un proces nu poate rula
- Nici un proces nu poate elibera nici o resursa
- Nici un proces nu poate fi deblocat

Conditii pentru interblocare

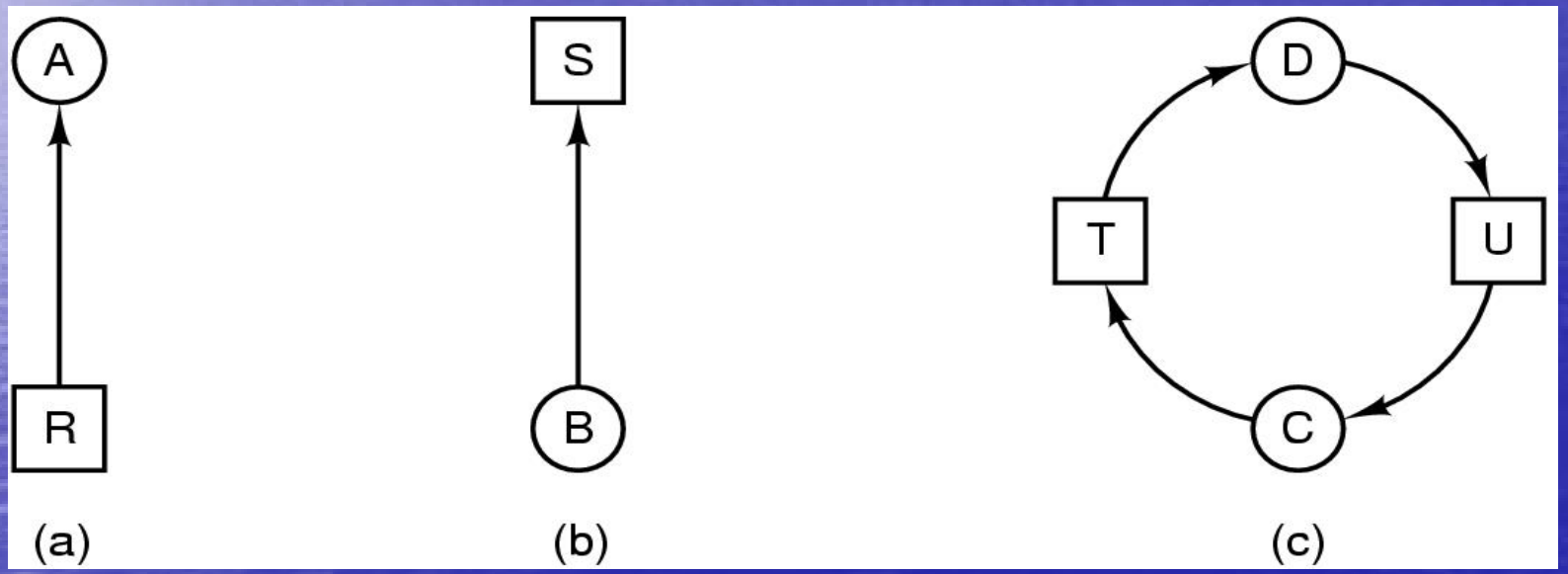
- Trebuie sa fie indeplinite patru conditii pentru a fi posibila interblocarea:
  1. Conditia de excludere mutuala. Fiecare resursa este fie alocata unui singur proces, fie disponibila
  2. Conditia de detinere si asteptare. Procesele care detin resurse pot cere noi resurse
  3. Condita de lipsa preemptiei. Resursele la care s-a obtinut deja accesul nu pot fi luate cu forta de la un proces. Ele trebuie sa fie eliberate in mod explicit de la procesul care le detine
  4. Conditia de asteptare circulara. Trebuie sa existe un lant circular de doua sau mai multe resurse, fiecare din ele asteptand la o resursa detinuta de urmatorul membru al lantului

Conditii pentru interblocare

- Important!
  - Toate aceste patru conditii trebuie sa fie indeplinite
  - Pentru a fi posibila aparitia interblocarilor
  - Daca una dintre ele nu este indeplinita nu este posibila nici o interblocare

Modelarea interblocarilor

- Cu ajutorul grafurilor orientate
- Procesele-reprezentate prin cercuri
- Resursele-reprezentate prin patrate



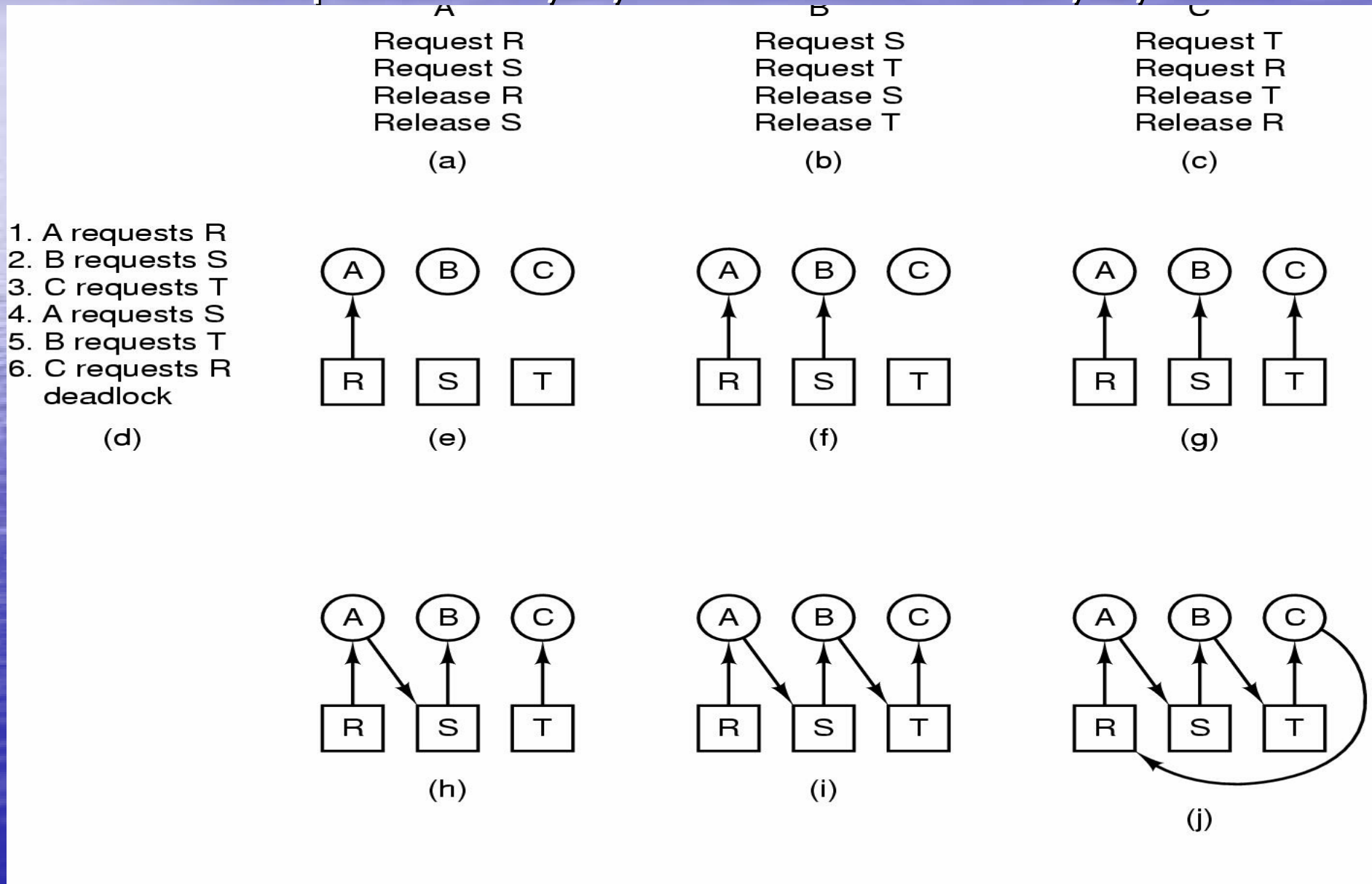
- Resursa R este in prezent alocata procesului A
- Procesul B asteapta eliberarea resursei S
- Interblocare-procesele C si D.

Modelarea interblocarilor

- Procesul C asteapta dupa resursa T, care este in prezent detinuta de procesul D
- Procesul D nu va elibera resursa T deoarece asteapta dupa resursa U, detinuta de C
- Ambele procese asteapta la nesfirsit
- Un ciclu in graf arata ca exista o interblocare, ce implica procesele si resursele din ciclu
- In figura precedenta ciclul este C-T-D-U-C

Modelarea interblocarilor

- Ex.: modul in care grafurile de resurse pot fi folosite
- Avem trei procese A, B, C si trei resurse R, S, T



Modelarea interblocarilor

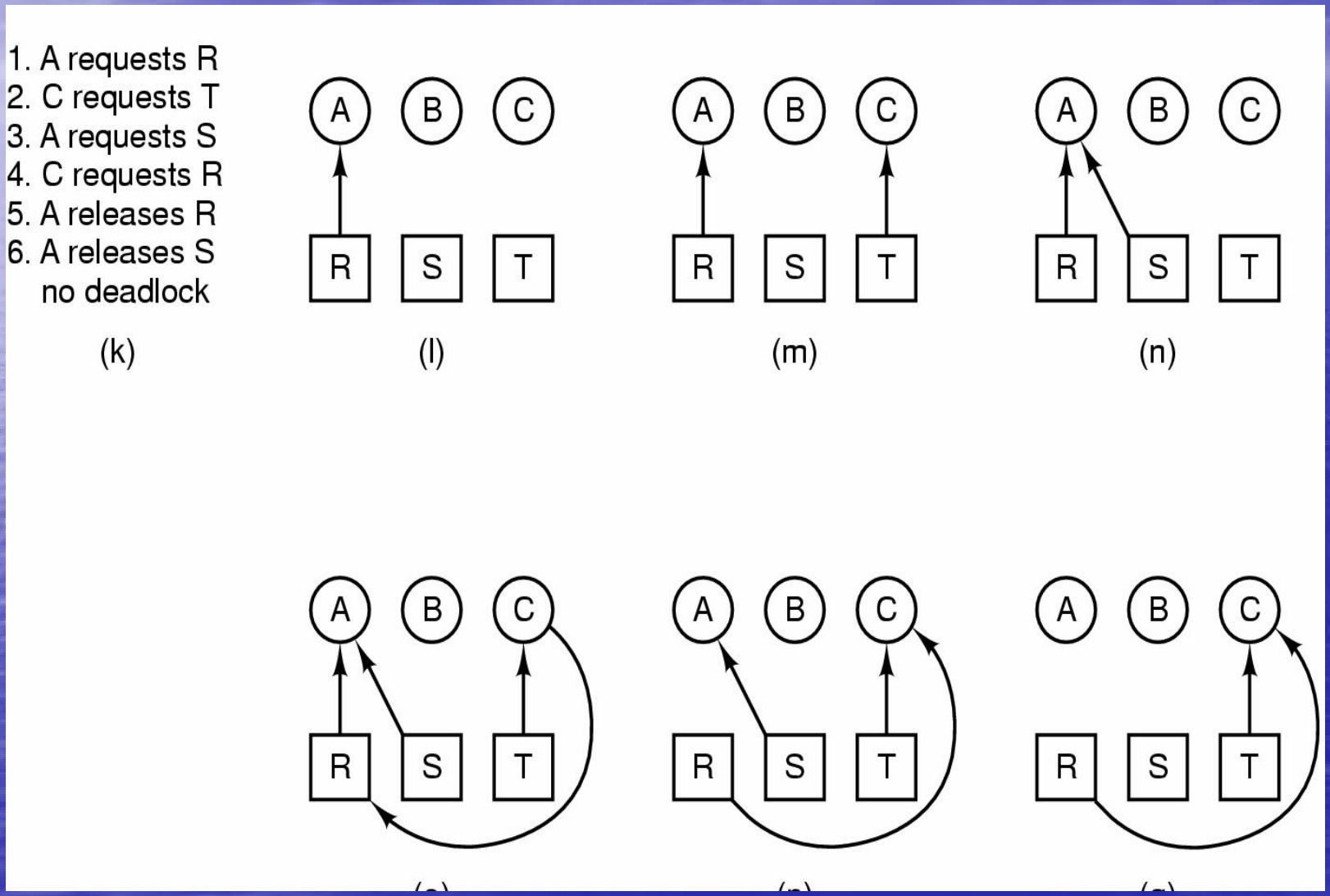
- In cazurile (a), (b) si (c) sistemul de operare este liber sa execute oricand orice proces care nu este blocat
- Mai intai ruleaza A pana isi termina treaba
- Apoi ruleaza B pana la terminarea lui
- In final ruleaza procesul C
- Aceasta ordonare nu duce la interblocare
- Executia secventiala a proceselor nu e optima

Modelarea interblocarilor

- Avem cererile de resurse din fig. (d)
- Grafurile rezultate sunt cele din fig. (e)-(j)
- Dupa cererea nr. 4, A se blocheaza in asteptarea lui S
- In urmatorii 2 pasi B si C se blocheaza
- Se ajunge la interblocare: fig. (j)

Modelarea interblocarilor

Procesul B este suspendat



Modelarea interblocarilor

- Dupa pasul q, procesului B ii poate fi acordata resursa S
- Deoarece A a terminat si C are tot ce-i trebuie
- Chiar daca B s-ar bloca pana la urma cerand acces la T, nu poate aparea interblocare
- B va astepta doar pana cand C va termina

Strategii impotriva interblocarilor

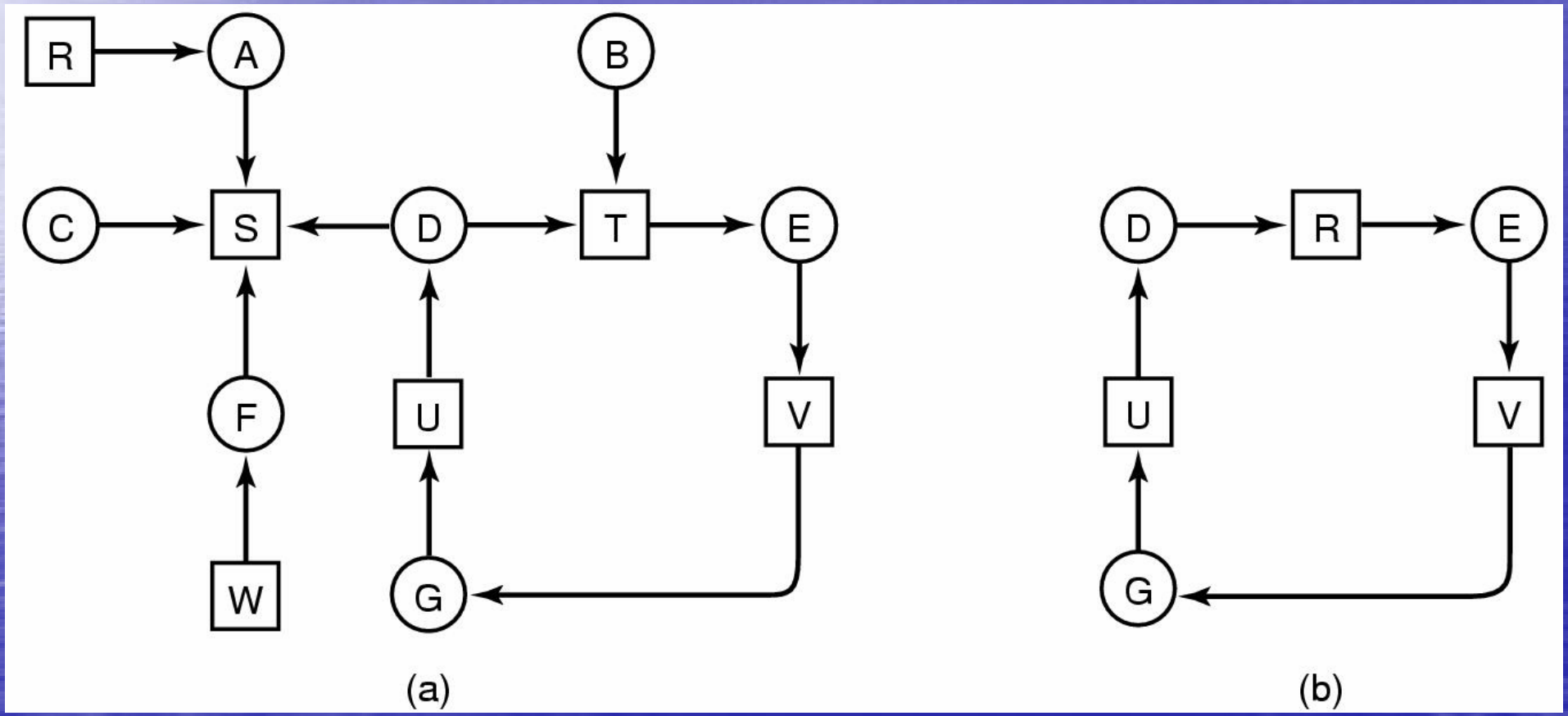
- Ignorarea in totalitate a problemei
- Detectarea si restaurarea
- Evitarea dinamica printr-o alocare cu atentie a resurselor
- Prevenirea
  - Prin negarea uneia dintre cele patru conditii necesare pentru aparitia interblocarii

Detectarea interblocarii si restaurarea

Detectarea interblocarii pentru o singura resursa din fiecare tip

- Exista o singura resursa din fiecare tip
- Starea sistemului d.p.d.v. al resurselor cerute, respectiv detinute:
  - Procesul A detine resursa R si cere resursa S
  - Procesul B nu detine nici o resursa, dar cere resursa T
  - Procesul C nu detine nici o resursa, dar cere resursa S
  - Procesul D detine resursa U si cere resursele S si T
  - Procesul E detine resursa T si cere resursa V
  - Procesul F detine resursa W si cere resursa S
  - Procesul G detine resursa V si cere resursa U

Detectarea interblocarii pentru o singura resursa din fiecare tip



Detectarea interblocarii pentru o singura resursa din fiecare tip

- Graful de resurse in fig. (a)
- Graful ce contine un ciclu in fig. (b)
- Procesele D, E si G se afla in interblocare
- Procesele A, C si F nu sunt in interblocare
- Deoarece S poate fi alocata oricaruia dintre ele, care apoi termina si o elibereaza
- Apoi celelalte doua o pot lua pe rand si se pot termina

Detectarea interblocarii pentru o singura resursa din fiecare tip

- Exista algoritmi care sa detecteze ciclurile din grafurile orientate
- Este folosita o singura structura de date – o lista de noduri L
- Arcele vor fi marcate pentru a indica faptul ca ele au fost deja parcurse
- Cu scopul de a preveni inspectarea repetata a lor

Detectarea interblocarii pentru o singura resursa din fiecare tip

- Algoritmul:
  1. Pentru fiecare nod N din graf, executa urmatorii pasi avandu-l pe N ca nod de start
  2. Initializeaza L la lista vida si seteaza toate arcele ca fiind nemarcate
  3. Adauga nodul curent la sfarsitul lui L si verifica daca nodul apare in L de doua ori. Daca apare, graful are un ciclu si algoritmul se termina
  4. Pornind de la nodul curent, verifica daca mai exista arce nemarcate care pornesc din el. Daca da, executa pasul 5; daca nu, executa pasul 6

Detectarea interblocarii pentru o singura resursa din fiecare tip

5. Alege la intamplare un arc nemarcat care pleaca din acel nod si marcheaza-l. Apoi, mergand pe acel arc, ia urmatorul nod si reia de la pasul 3
6. Am ajuns la o fundatura. Scoate nodul din lista si reia de la pasul 3 considerand nodul anterior ca fiind nodul curent Daca acest nod este nodul initial, graful nu contine cicluri si algoritmul se termina.

Detectarea interblocarii pentru mai multe resurse din fiecare tip

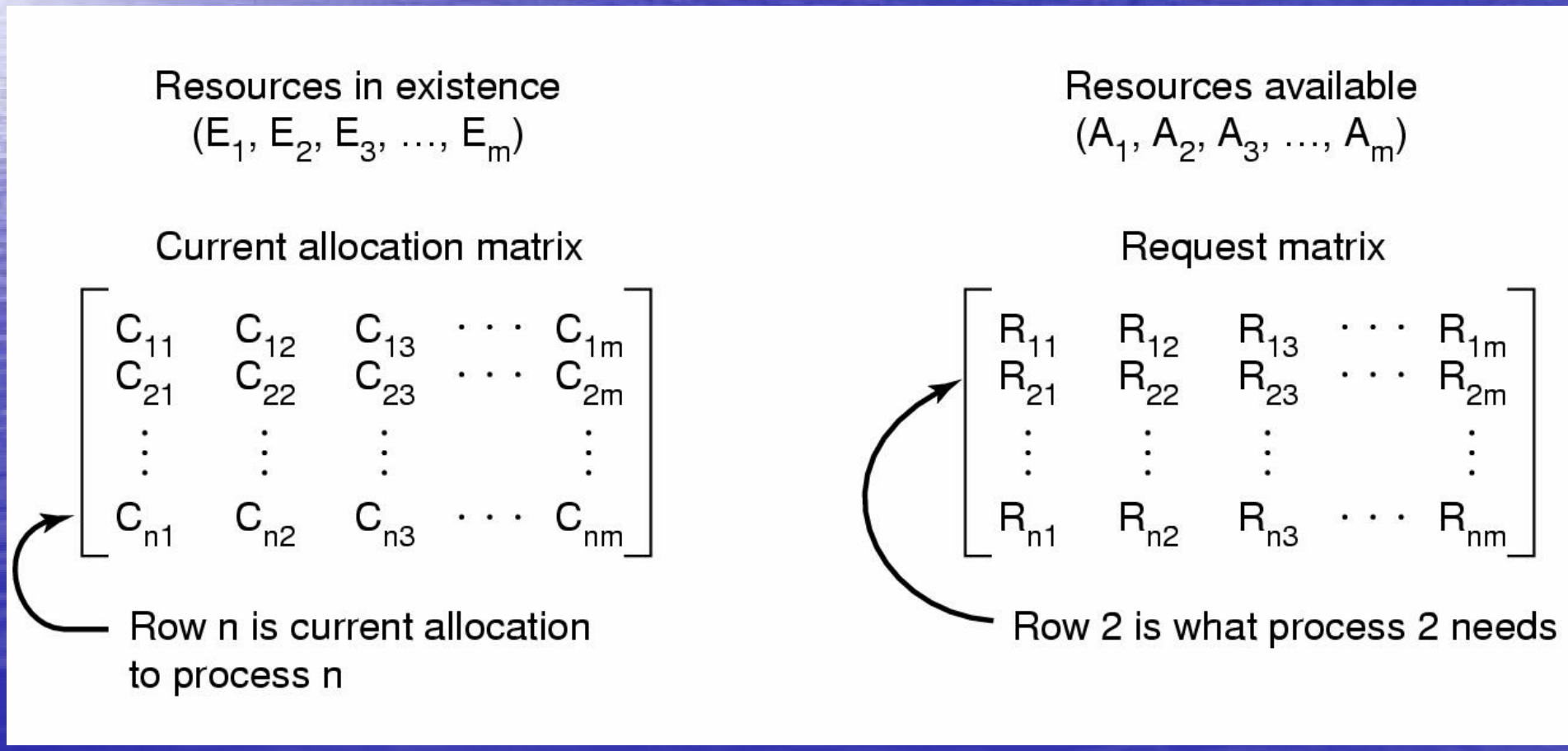
- Fie *n* procese notate *P1, P2, …, Pn*
- Fie *m* numarul de clase de resurse
- *E1* - resurse de clasa 1
- *E2* – resurse de clasa 2,…,*En* - resurse de clasa *n*
- *E* este vectorul de resurse existente
- *E* furnizeaza numarul total de instante ale fiecarei resurse existente
- De ex., daca resursele de clasa 1 sunt unitatile de banda magnetica, atunci *E1=2* inseamna ca sistemul are 2 unitati de banda magnetica

Detectarea interblocarii pentru mai multe resurse din fiecare tip

- In orice moment unele resurse sunt alocate si nu sunt disponibile
- Fie *A* vectorul de resurse disponibile
- *Ai* fiind numarul instantelor resursei *i* care sunt disponibile in prezent (nu sunt alocate nici unui proces)
- Daca ambele unitati de banda magnetica sunt alocate, *A1* va fi 0
- Fie *C* matricea de alocare curenta
- Randul *i* din matricea *C* arata cate instante ale fiecarei clase de resurse detine procesul *Pi*
- *Cij* – numarul de instante ale resursei *j* care sunt detinute de procesul *i*

Detectarea interblocarii pentru mai multe resurse din fiecare tip

- Fie *R* matricea de cereri
- *Rij* reprezinta numarul de instante ale resursei *j* pe care procesul *Pi* le cere



Detectarea interblocarii pentru mai multe resurse din fiecare tip

- Fiecare resursa este fie alocata, fie disponibila:

$$\sum_{i=1}^{n} C_{ij} + A_j = E_j \quad (\forall j = 1,\ldots,m)$$

- Adica, daca adunam toate instantele resursei *j* care au fost alocate cu toate instantele disponibile ale ei
- Va rezulta numarul total de instante existente pentru acea clasa de resurse

Detectarea interblocarii pentru mai multe resurse din fiecare tip

- Initial fiecare proces este nemarcat
- Pe parcurs ele vor fi marcate indicand faptul ca ele isi pot termina treaba si deci nu se afla in interblocare
- Orice proces nemarcat dupa terminarea algoritmului se afla in interblocare

Detectarea interblocarii pentru mai multe resurse din fiecare tip

- Algoritmul de detectare a interblocarii:
  1. Cauta un proces nemarcat Pi, pentru care randul i din R este mai mic sau egal cu A
  2. Daca este gasit un asemenea proces
     - aduna randul i din C la A,
     - marcheaza procesul si reia de la pasul 1
  3. Daca nu exista un asemenea proces, algoritmul se termina

Detectarea interblocarii pentru mai multe resurse din fiecare tip

- Avem 3 procese si 4 clase de resurse
- Procesul 1 are un scanner, procesul 2 are 2 unitati de banda magnetica si o unitate CD-ROM, procesul 3 are un trasator (plotter) si 2 scannere

Tape drives Plotters Scanners CD Roms

$$E = (4 \quad 2 \quad 3 \quad 1)$$

Tape drives Plotters Scanners CD Roms

$$A = (2 \quad 1 \quad 0 \quad 0)$$

Current allocation matrix

$$C = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 2 & 0 & 0 & 1 \\ 0 & 1 & 2 & 0 \end{bmatrix}$$

Request matrix

$$R = \begin{bmatrix} 2 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 2 & 1 & 0 & 0 \end{bmatrix}$$

Detectarea interblocarii pentru mai multe resurse din fiecare tip

- Fiecare proces are nevoie de resurse suplimentare- vezi matricea R
- Doar cererea celui de-al treilea proces poate fi satisfacuta
- Asadar, procesul 3 ruleaza si apoi elibereaza resursele sale, rezultand: A=(2,2,2,0)
- Procesul 2 poate rula si apoi elibereaza resursele sale, rezultand: A=(4,2,2,1)
- Acum poate rula si procesul 1
- Astfel, sistemul nu contine nici o interblocare

Restaurarea in urma interblocarii

- Algoritmul a detectat o interblocare
- Este necesara restaurarea sistemului astfel incat sa functioneze din nou
- Restaurarea prin preemtiune
  - Ia o resursa de la detinator si o aloca altui proces
  - Depinde in mare masura de natura resursei
- Restaurarea prin revenire (Rollback)
  - Verifica fiecare proces periodic (se creaza puncte de reluare)
  - Se utilizeaza datele salvate ale procesului, astfel incat sa poata fi repornit mai tarziu din acelasi punct

Restaurarea prin distrugerea proceselor

- Metoda cea mai primitiva si cea mai simpla
- O posibilitate este distrugerea unui proces dintr-un ciclu
- Celelalte procese isi vor putea continua executia
- Distrugerea se poate repeta pana cand ciclul este terminat
- Cand este posibil, este mai bine sa fie distruse procese care pot fi executate din nou de la inceput fara efecte nedorite

Nici una dintre metode nu este atractiva in mod special !

Evitarea interblocarilor

- Exista un algoritm care sa poata evita interblocarea facand tot timpul alegerea potrivita?
- Raspunsul este: DA
- Dar numai daca anumite informatii sunt disponibile in avans
  - Fiecare proces trebuie sa “declare”, in avans, de ce resurse are nevoie
  - Si care este numarul maxim de resurse de care are nevoie

Stari sigure si nesigure

- Starea unui sistem inseamna:
  - Cate resurse sunt alocate fiecarui proces
  - Care este numarul maxim de resurse pe care un proces le poate cere
  - Cate resurse sunt disponibile
- O stare este sigura daca:
  - Sistemul nu este in interblocare in acel moment
  - Exista o ordine de planificare in care fiecare proces poate rula pana la terminare

Stari sigure si nesigure

- Ex.: Se foloseşte o singură resursă (cu un numar total de 10 instante ale resursei)

	Has	Max
A	3	9
B	2	4
C	2	7

Free: 3

(a)

	Has	Max
A	3	9
B	4	4
C	2	7

Free: 1

(b)

	Has	Max
A	3	9
B	0	–
C	2	7

Free: 5

(c)

	Has	Max
A	3	9
B	0	–
C	7	7

Free: 0

(d)

	Has	Max
A	3	9
B	0	–
C	0	–

Free: 7

(e)

- In fig. (a) exista o stare in care A are 3 instante ale resursei, dar poate avea nevoie in cele din urma de 9
- B detine 2 instante si ar putea avea nevoie de 4
- C are 2 instante si ar putea avea nevoie de 7
- Exista 10 instante ale resursei, 7 sunt deja ocupate

Stari sigure si nesigure

- Starea din fig. (a) este sigura
- Exista o secventa de alocari care permit tuturor proceselor sa-si termine corect executia
- Este planificat sa ruleze mai intii procesul B
- Cand se termina de executat B (fig. c) se incepe executia procesului C (fig. d)
- In final se executa procesul A (obtine cele 6 instante ale resursei)
- Se termina executia procesului A

Stari sigure si nesigure

- Procesul A cere si primeste inca o resursa
- Se ruleaza B pana se termina si se ajunge la fig. (c)
- S-a ajuns dintr-o stare sigura (fig. a) intr-una nesigura (fig. b)
- Cererea lui A nu ar fi trebuit sa fie aprobata

	Has	Max
A	3	9
B	2	4
C	2	7

Free: 3

(a)

	Has	Max
A	4	9
B	2	4
C	2	7

Free: 2

(b)

	Has	Max
A	4	9
B	4	4
C	2	7

Free: 0

(c)

	Has	Max
A	4	9
B	—	—
C	2	7

Free: 4

(d)

Stari sigure si nesigure

- Stare sigura – sistemul poate garanta ca toate procesele se vor termina
  - Interblocarile sunt evitate
- Stare nesigura – nu se poate da o astfel de garantie
  - Nu inseamna neaparat ca se va ajunge la o interblocare