DISTRIBUIRANI ALGORITMI I SISTEMI

Iz kursa CSCE 668
Proleće 2014

Autor izvorne prezentacije:
Prof. Jennifer Welch

Nemogućnost asinhronog konsenzusa (1/2)

- Pokazati nemogućnost za read/write deljenu memoriju sa *n* procesora i $n - 1$ otkaza
  - dokazati direktno: lako jer ima mnogo otkaza
  - implicira da ne postoji alg. za 2-proc i 1 otkaz
- Pokazati nemogućnost za r/w deljenu memoriju sa *n* proc. i 1 otkazom. Dva pristupa:
  - Redukcija: koristiti hipotetički alg. za *n*-proc i 1 otkaz kao podprogram za projektovanje alg. za 2-proc i 1 otkaz
  - Direktno: Slične ideje kao za slučaj $n-1$ otkaza

Nemogućnost asinhronog konsenzusa (2/2)

- Pokazati nemogućnost za sis. sa slanjem poruka sa *n* procesora i 1 otkazom. Dva pristupa:
  - Redukcija: Koristiti hipotetički alg. sa slanjem poruka za *n* proc. i 1 otkazom kao podprog. za projektovanje alg. sa deljenom memorijom za *n* proc. i 1 otkazom. To vodi do kontradikcije sa predhodnim rezultatom.
  - Direktno: Koristiti slične ideja kao za sl. deljene memorije, pojačane rukovanjem porukama. (Istorijski, ova verzija je prva dokazana.)

Model asinhronog sistema sa otkazima tipa ispada

- Neka je *f* maks. broj procesora u otkazu.
- Za SM i MP: Svi osim *f* procesora moraju izvesti beskonačan br. koraka u prihvatljivom izvršenju.
- Za MP: Takođe zahteva da sve poruke poslate <u>ka</u> ispravnom procesoru moraju biti nekada konačno isporučene, osim onih poslatih <u>od</u> proc. u otkazu, u njegovom zadnjem koraku, koje mogu a ne moraju biti isporučene.

Algoritmi oslobođeni čekanja (Wait-Free, WF)

- Alg. za *n* procesora je **WF** alg. ako toleriše *n − 1* otkaza.
- Intuicija je da ispravan procesor ne čeka na druge procesore da nešto urade: on to ne može, jer on može biti jedini procesor koji je ostao aktivan.
- Prvi rezultat je da ne postoji WF alg. konsenzusa za asinhron model sa r/w deljenom memorijom.

Nemogućnost WF konsenzusa

- Pred. radi kontradikcije da postoji algoritam za n procesora i $n - 1$ otkaza u asinhronom modelu sa read/write deljenom memorijom.
- Dokaz je sličan onom da je potrebno $f + 1$ rundi u sinhronom modelu sa slanjem poruka.

Izmenjene definicije valenci

- U dokazu donje granice za broj rundi, valenca se odnosila na odluke koje su dostupne u prihvatljivim izvršenjima sa *retkim otkazima*.
- U ovom dokazu nas interesuju odluke koje su dostupne u *bilo kom* prihvatljivom izvršenju (za asinhron model sa deljenom memorijom, prihvatljivo je do $n - 1$ otkaza).

Univalentna sličnost

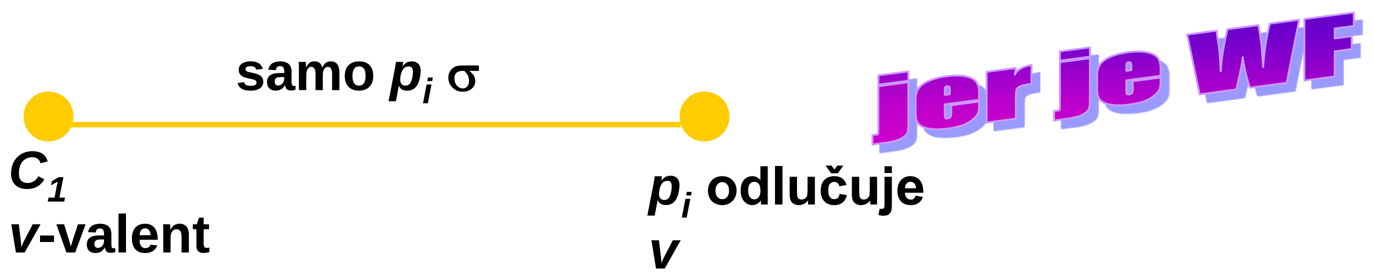
Lema (10.15): Ako su C_1 i C_2 univalentne i *slične za* p_i (stanje deljene memorije je isto, i lokalno stanje od p_i je isto), onda one imaju istu valencu.

Dokaz:

Univalentna sličnost

Lema (10.15): Ako su C_1 i C_2 univalentne i *slične za* p_i (stanje deljene memorije je isto, i lokalno stanje od p_i je isto), onda one imaju istu valencu.

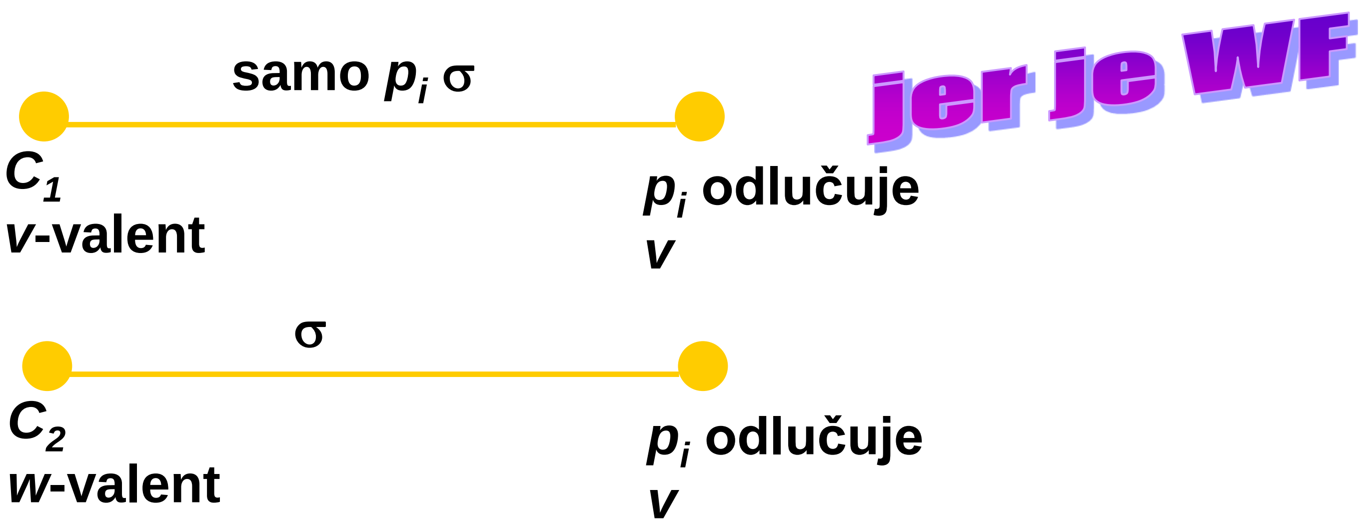
Dokaz:



Univalentna sličnost

Lema (10.15): Ako su C_1 i C_2 univalentne i *slične za* p_i (stanje deljene memorije je isto, i lokalno stanje od p_i je isto), onda one imaju istu valencu.

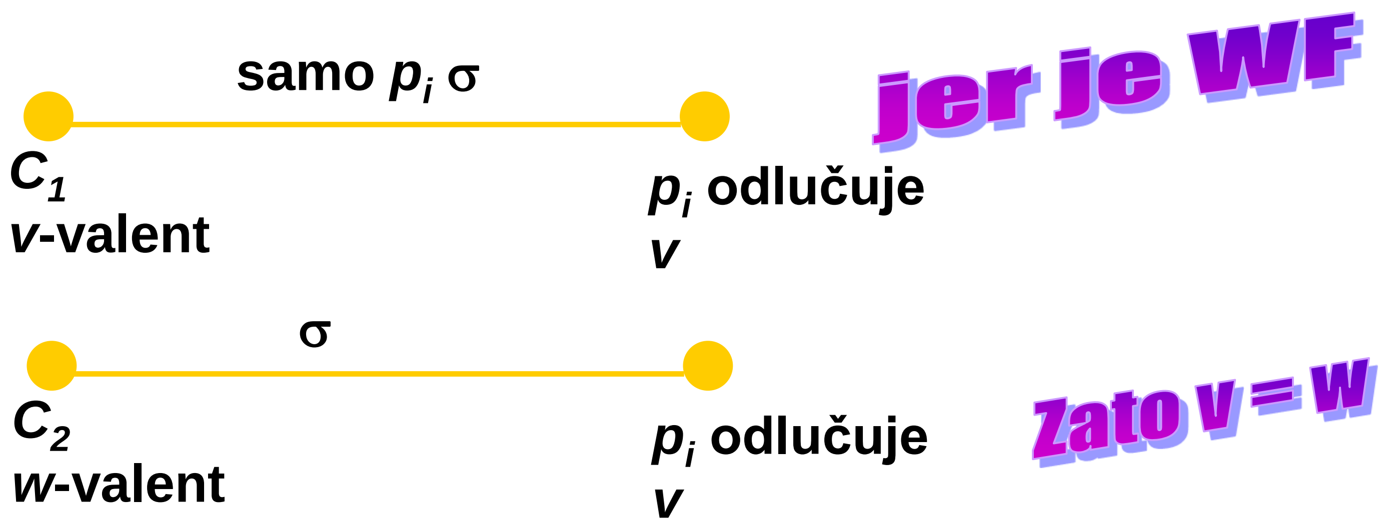
Dokaz:



Univalentna sličnost

Lema (10.15): Ako su C_1 i C_2 univalentne i *slične za* p_i (stanje deljene memorije je isto, i lokalno stanje od p_i je isto), onda one imaju istu valencu.

Dokaz:



Bivalentna početna konfiguracija

Lema (10.16): Postoji bivalentna početna konfiguracija.

Dokaz je sličan dokazu za $f + 1$ donju granicu za broj rundi u sinhronom modelu.

Kritičan procesor (1/4)

Def: Ako je C bivalentna i *i*(C) (rezultat kad p_i izvodi jedan korak) je univalentna, onda je p_i **kritičan** u C.

Lema (10.17): Ako je C bivalentna, onda bar jedan procesor nije kritičan u C, tj., postoji bivalentno proširenje.

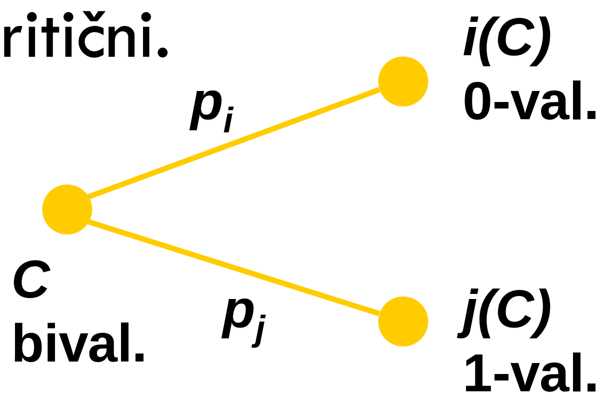
Dokaz: Pred. radi kontradikcije da su svi procesori kritični.

Kritičan procesor (1/4)

Def: Ako je C bivalentna i *i*(C) (rezultat kad p_i izvodi jedan korak) je univalentna, onda je p_i **kritičan** u C.

Lema (10.17): Ako je C bivalentna, onda bar jedan procesor nije kritičan u C, tj., postoji bivalentno proširenje.

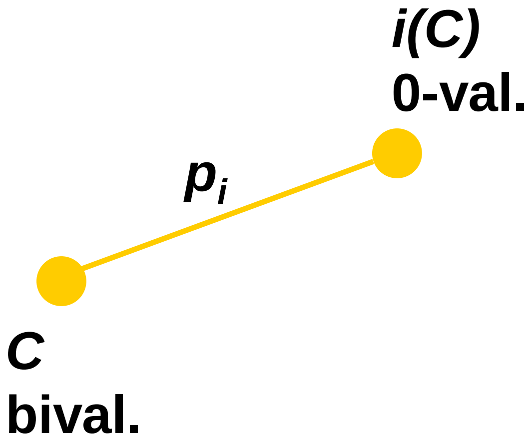
Dokaz: Pred. radi kontradikcije da su svi procesori kritični.



Dalje sledi analiza slučajeva za varijacije dva koraka p_i i p_j

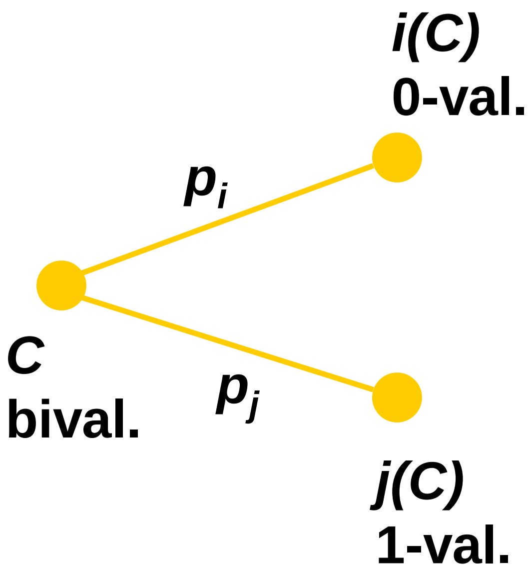
Kritičan procesor (2/4)

Slučaj 1: p_i i p_j pristupaju različitim registrima.



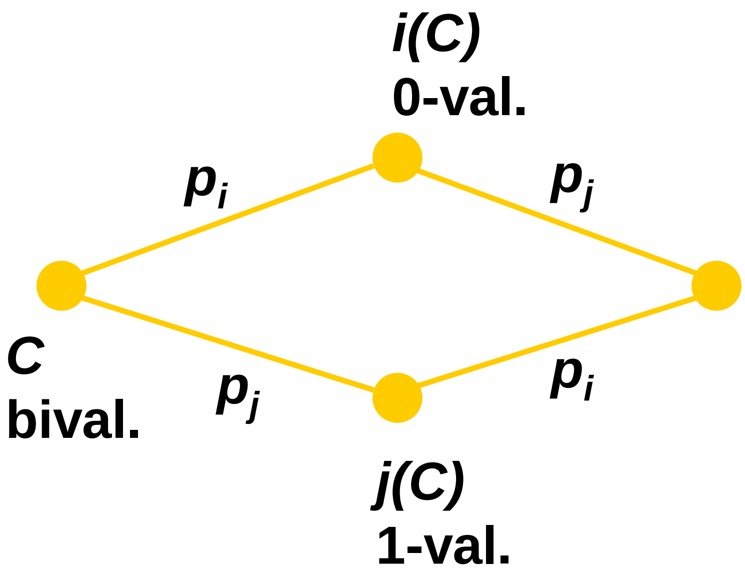
Kritičan procesor (2/4)

Slučaj 1: p_i i p_j pristupaju različitim registrima.



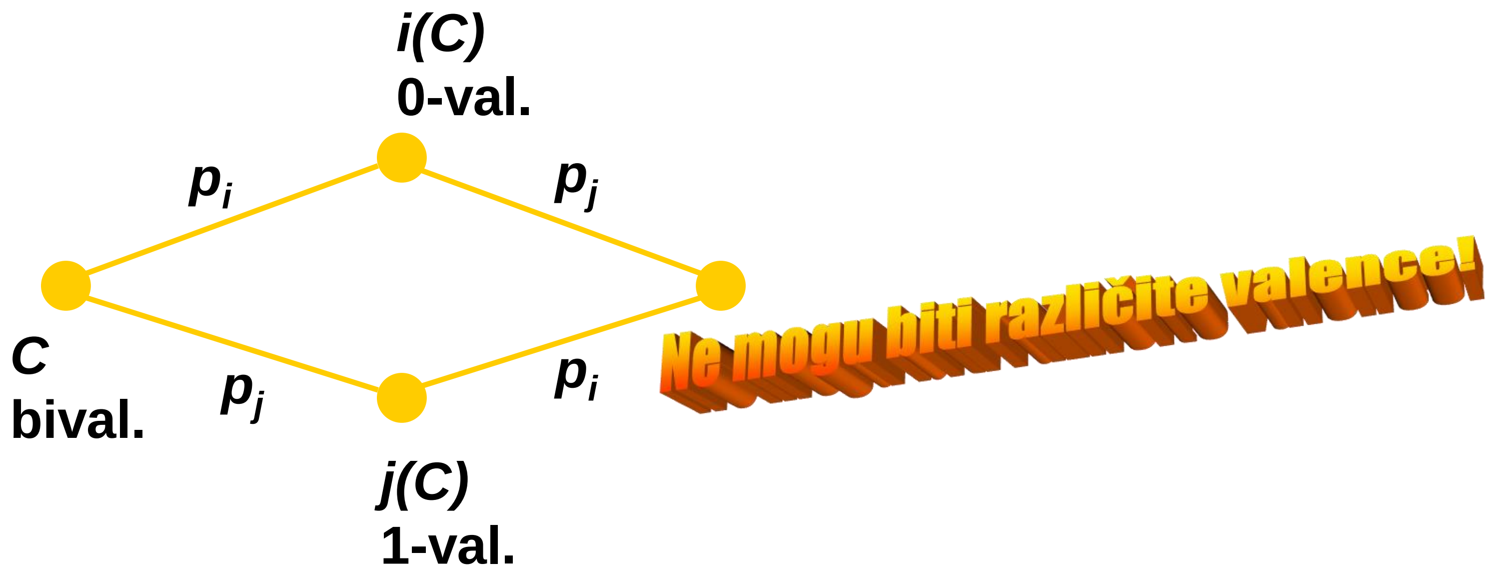
Kritičan procesor (2/4)

Slučaj 1: p_i i p_j pristupaju različitim registrima.



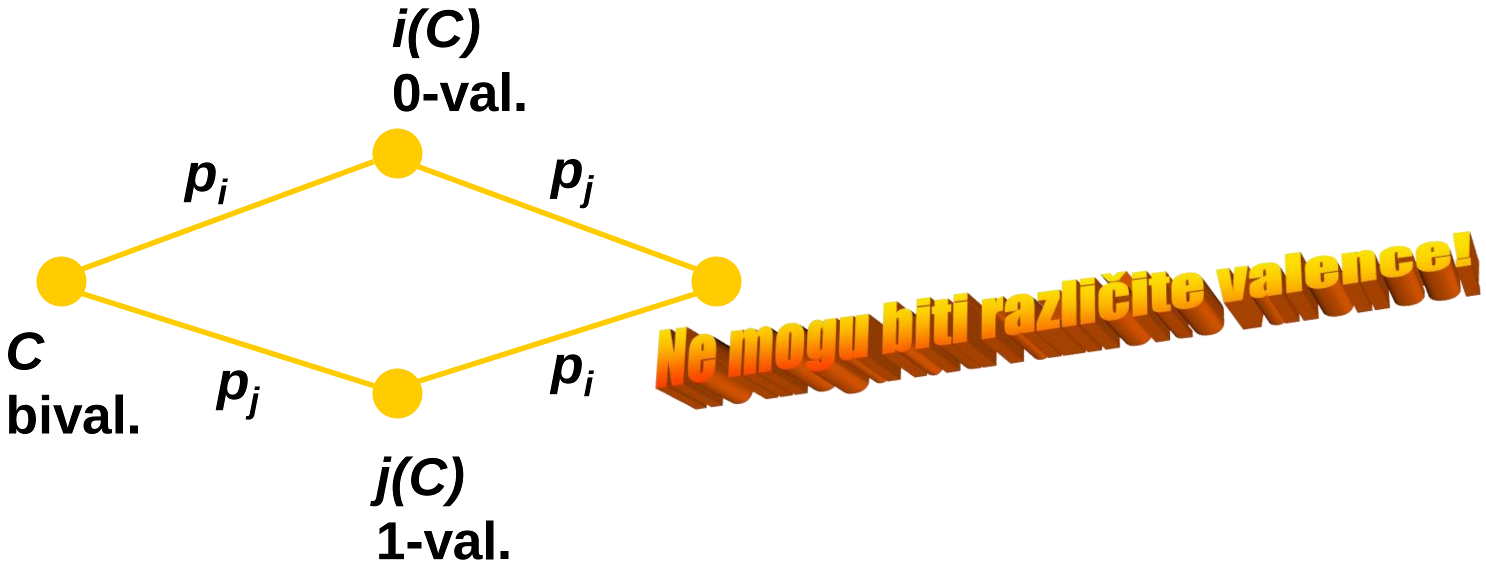
Kritičan procesor (2/4)

Slučaj 1: p_i i p_j pristupaju različitim registrima.



Kritičan procesor (2/4)

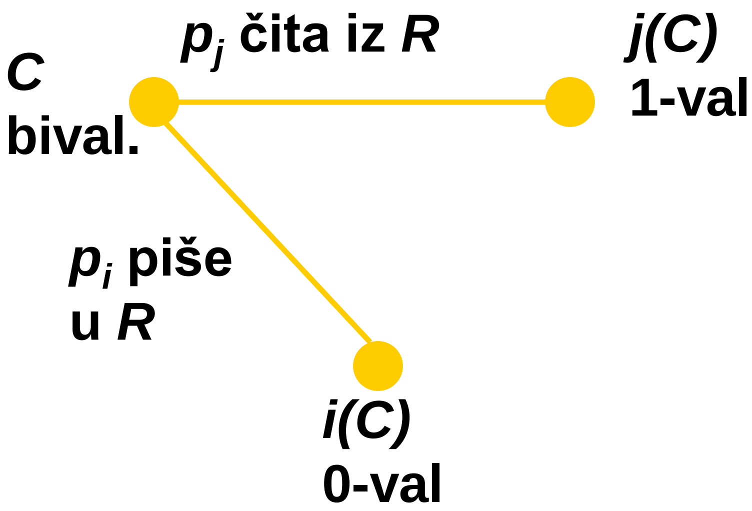
Slučaj 1: p_i i p_j pristupaju različitim registrima.



Slučaj 2: p_i i p_j čitaju isti registar. Isti dokaz.

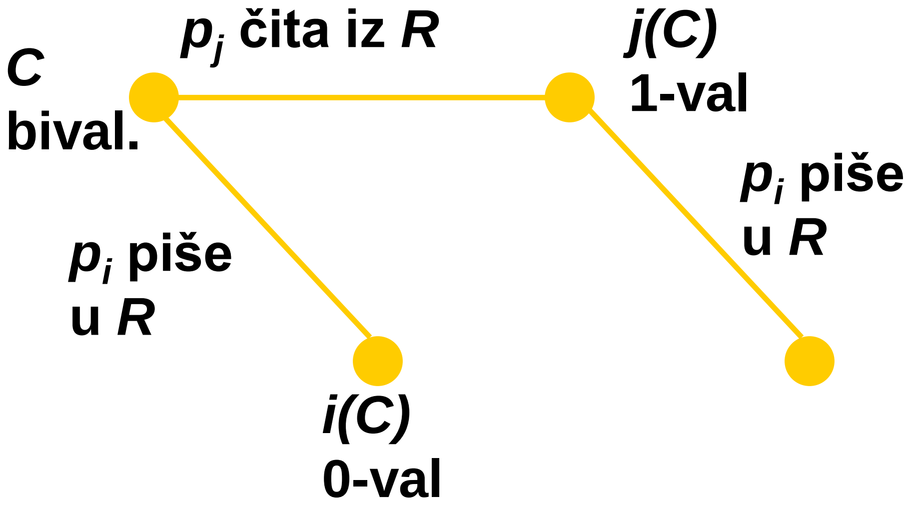
Kritičan procesor (3/4)

Slučaj 3: p_i piše u registar *R* a p_j čita iz *R*.



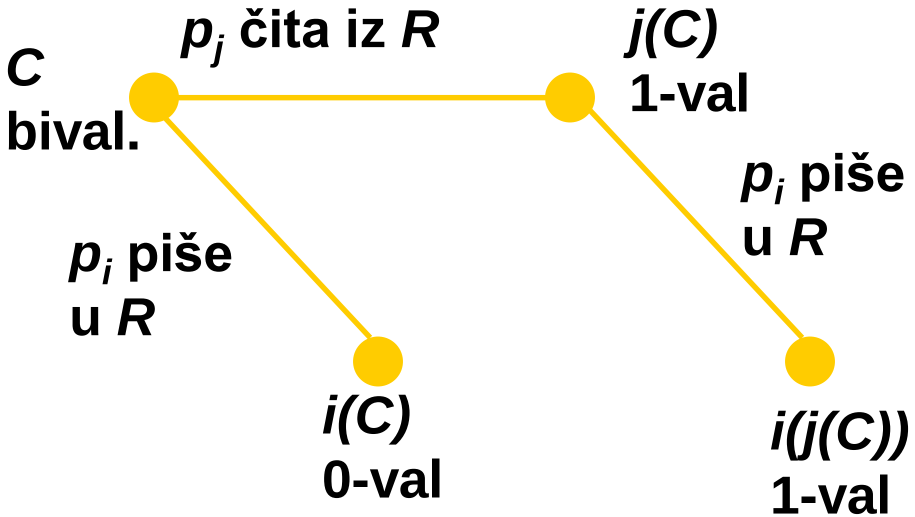
Kritičan procesor (3/4)

Slučaj 3: p_i piše u registar *R* a p_j čita iz *R*.



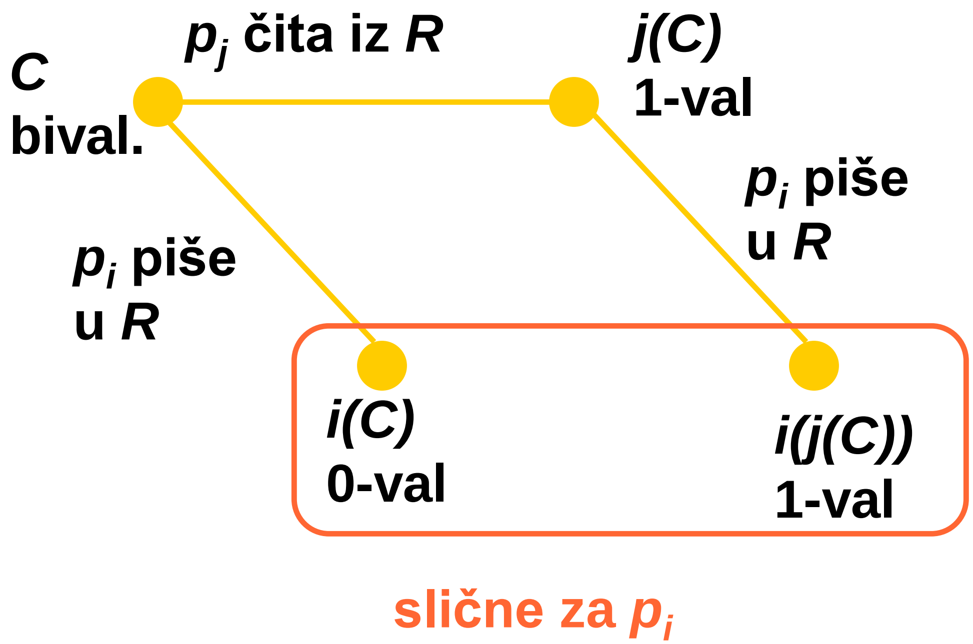
Kritičan procesor (3/4)

Slučaj 3: p_i piše u registar *R* a p_j čita iz *R*.



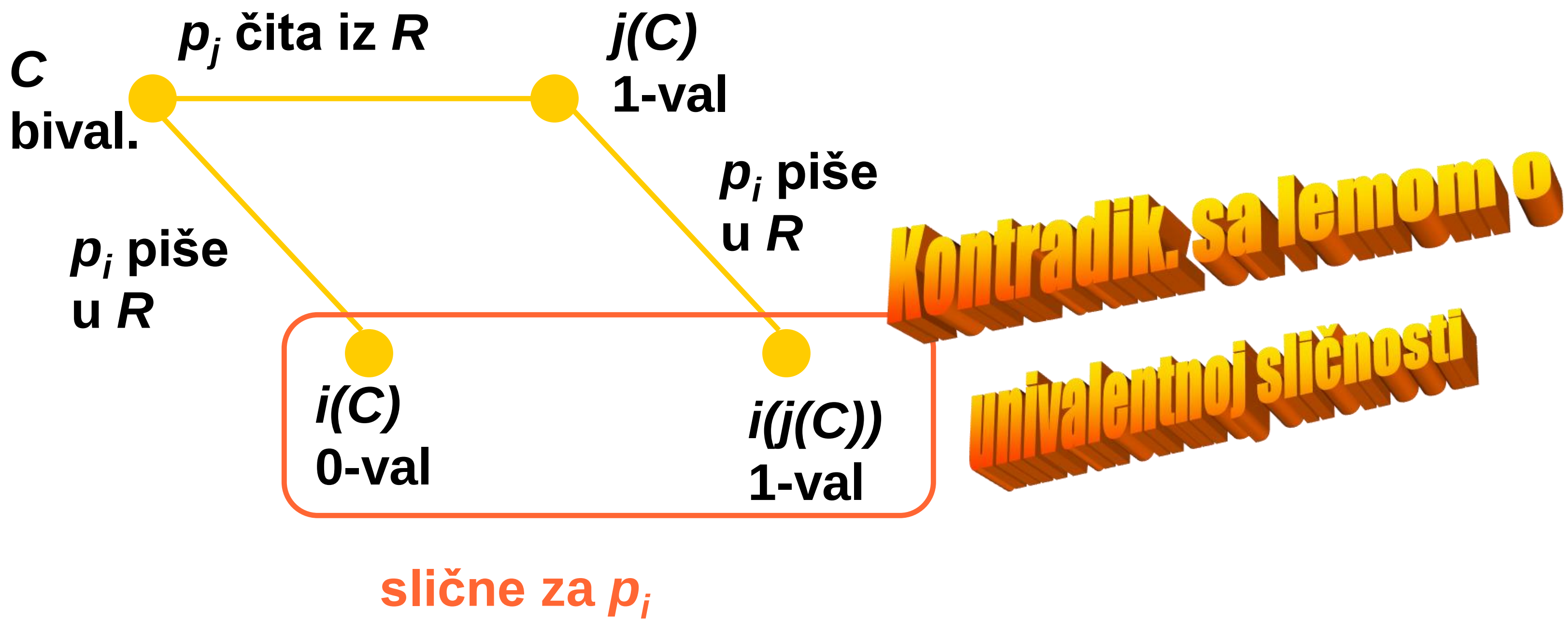
Kritičan procesor (3/4)

Slučaj 3: p_i piše u registar R a p_j čita iz R.



Kritičan procesor (3/4)

Slučaj 3: p_i piše u registar *R* a p_j čita iz *R*.



Kritičan procesor (4/4)

Slučaj 4: Šta ako i p_i i p_j pišu u istu deljenu promenljivu?

- Možemo „uprostiti" problem sa pred. da postoje samo deljene prom. u koje piše samo jedan proc.
- Ili, možemo izvesti dokaz sličan dokazu za slučaj 3.

Završetak dokaza nemogućnosti...

- Napravimo prihvatljivo izvr. $C_0, i_1, C_1, i_2, C_2, \ldots$ u kom su sve konfiguracije bivalentne.
  - dovodi do kontradikcije sa zahtevom za završetak
- Počnimo sa bivalentnom početnom konfig.
- Pred. da postoji bivalentna C_k.

  Da bi dobili bivalentnu C_{k+1}:
  - Neka je p_i procesor koji nije kritičan u C_k.
  - Neka C_{k+1} bude $i_{k+1}(C_k)$.

Nemogućnost 1-elastičnog konsenzusa: Ideja redukcije

Čak i ako broj ispravnih proc. postane dominantan, konsenzus se i dalje ne može rešiti u asinhronom SM (sa read/write registrima).

1. Pred. da postoji algoritam *A* za *n* procesora i 1 otkaz.
2. Uzmimo *A* kao podprogram u projektovanju algoritma *A'* za 2 procesora i 1 otkaz.
3. Ali, upravo smo dokazali da takav *A'* ne postoji.
4. Zato ni *A* ne postoji.

Nemogućnost 1-elastičnog konsenzusa: Ideja direktnog dokaza

- Pred. radi kontradik. da postoji takav algoritam.
- Strategija: Konstruisati prihvatljivo izvršenje (sa najviše 1 otkazom) koje se nikad ne završava:
  - Pokazati da postoji bivalentna početna konfiguracija
  - Pokazati kako ići od jedne bivalentne konfig do druge, zauvek (tako da se nikad ne završava)
- Tehnički teži dokaz, jer pri konstruisanju ovog izvršenja, ne može otkazati više od jednog procesora.

Nemogućnost konsenzusa u modelu sa slanjem poruka: Redukcija

Strategija:

1. Pred. da postoji 1-elastičan alg. konsenzusa *A* za *n*-proc u asinhronom modelu sa slanjem por.
2. Uzmimo *A* kao podprogram u projektovanju 1-elastičnog algoritma konsenzusa *A'* za *n*-proc. u asinhronom modelu sa deljenom memorijom (sa read/write prom.).
3. Ali, već smo dokazali da *A'* ne postoji.
4. Zato ni *A* ne postoji.

Nemogućnost konsenzusa u modelu sa slanjem poruka

Ideja za A':

- Simulirati kanale poruka sa read/write registrima.
- Onda postaviti alg. *A* na vrh ovih simuliranih kanala.

Za simulaciju kanala od p_i do p_j:

- Uzmimo jedan registar za sekvencu poruka poslatih preko ovog kanala
- p_i „šalje" poruku *m* upisom stare vrednosti registra na koju se doda *m*
- p_j „prima" poruku čitanjem registra i proverom novih poruka na kraju pročitane vrednosti