Hijerarhija konsenzusa
Deo 1

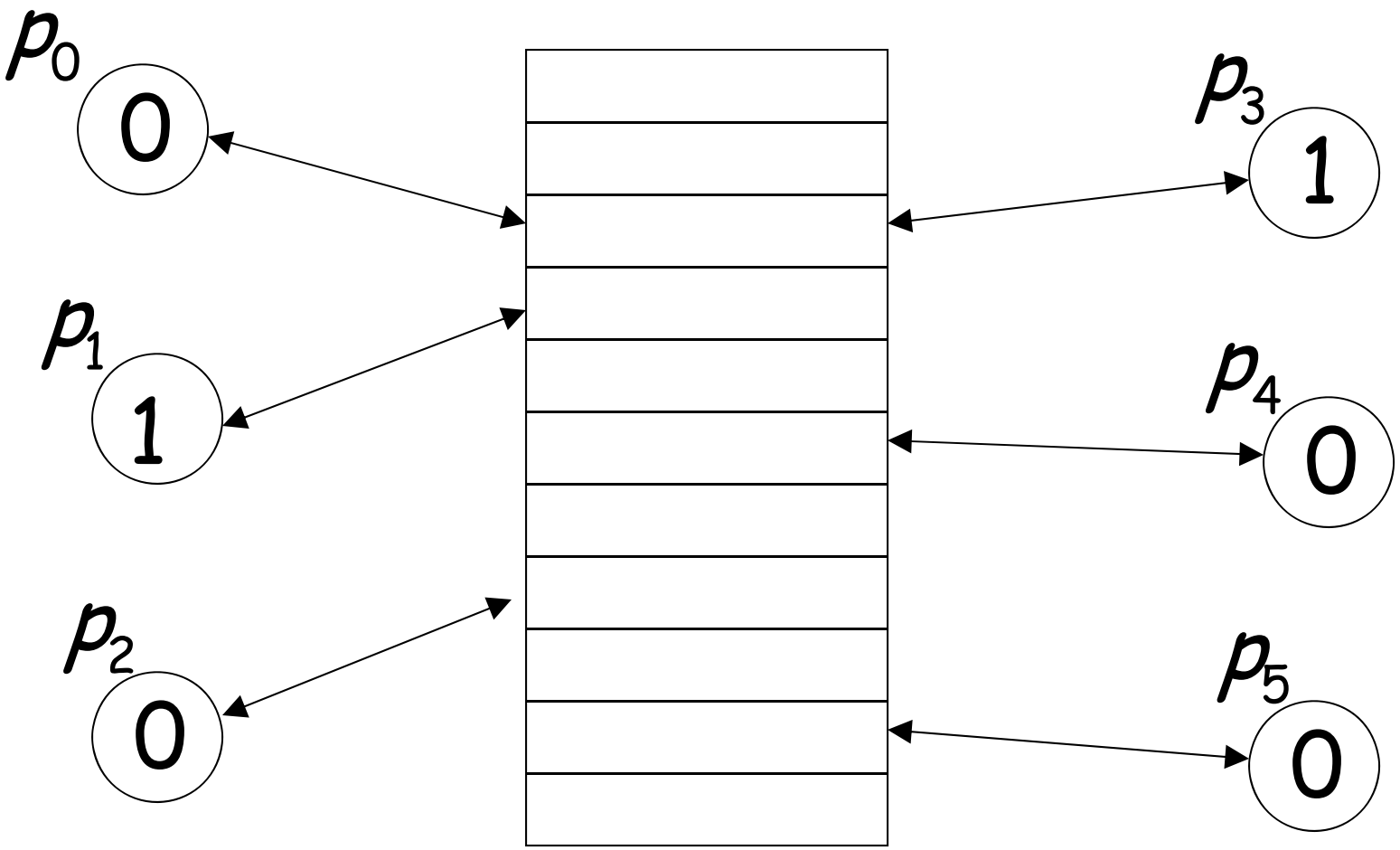
Konsenzus u deljenoj memoriji

Razmotrimo n procesora sa deljenom mem:

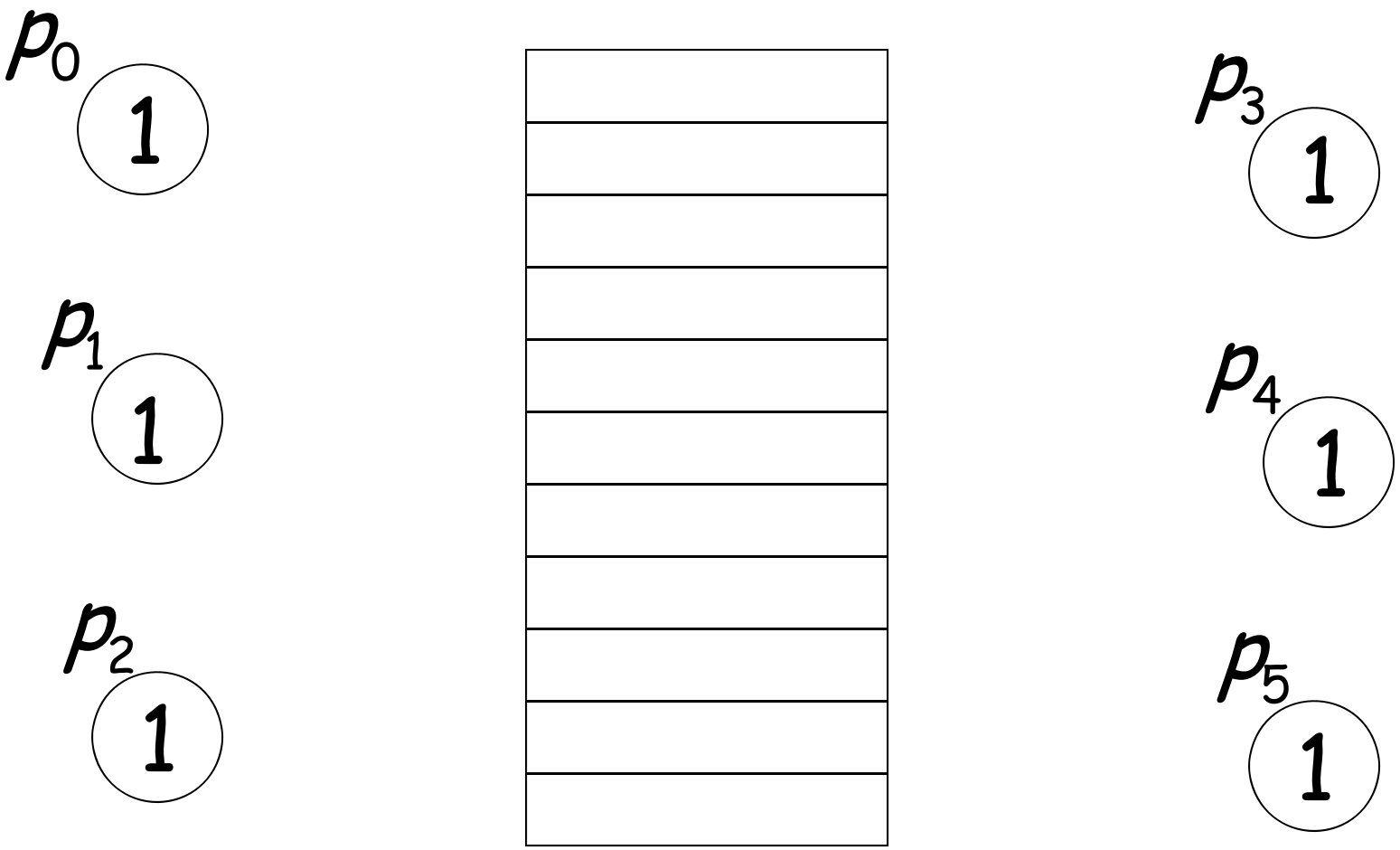
$$p_0, \ldots, p_{n-1}$$

koji pokušavaju da reše problem konsenzusa

Svaki proces počinje sa početnom vrednošću smeštenoj u lokalnoj memoriji (0 ili 1)



komunikacija kroz deljenu memoriju



Na kraju izvršenja, svi procesi su se odlučili za istu vrednost (0 ili 1)

Uslov validnosti:

Ako svaki proces počne sa istom vrednošću,
onda svi procesi treba da se odluče za tu vred

Dodatni uslov:

Odlučena vrednost je jedna od početnih vred

Oslobođen-čekanja (Wait-freedom, WF)

u asinhronim sistemima:

Proces bi trebao da može da
završi izvršenje algoritma
čak i ako svi drugi procesi otkažu

Pojam oslobođen-čekanja obuhvata:

- Asinhrona izvršenja
- Otkaze tipa ispada

Tipovi objekata

Read/Write

FIFO

Test&Set

Upis u n-registara
(n-register assignment)

Compare&Swap

Broj konsenzusa (Consensus Number)

Broj konsenzusa (CN) za dati tip objekta:

Je **maksimalan** broj procesa od kojih objekt može biti korišćen da se reši problem konsenzusa oslobođen-čekanja (zajedno sa read/write objektima)

Tip objekta	Broj konsenzusa
Read/Write	1
FIFO, Test&Set	2
upis u n-registara	2n-2
Compare&Swap	∞ (beskonačan)

```
int Test-and-Set(boolean lock) {
        boolean initial = lock;
        lock = true;
        return initial;
}
```

```
int compare_and_swap ( int* register, int oldval, int newval) {
      int old_reg_val = *register;
      if (old_reg_val == oldval)
         *register = newval;
      return old_reg_val;
}
```

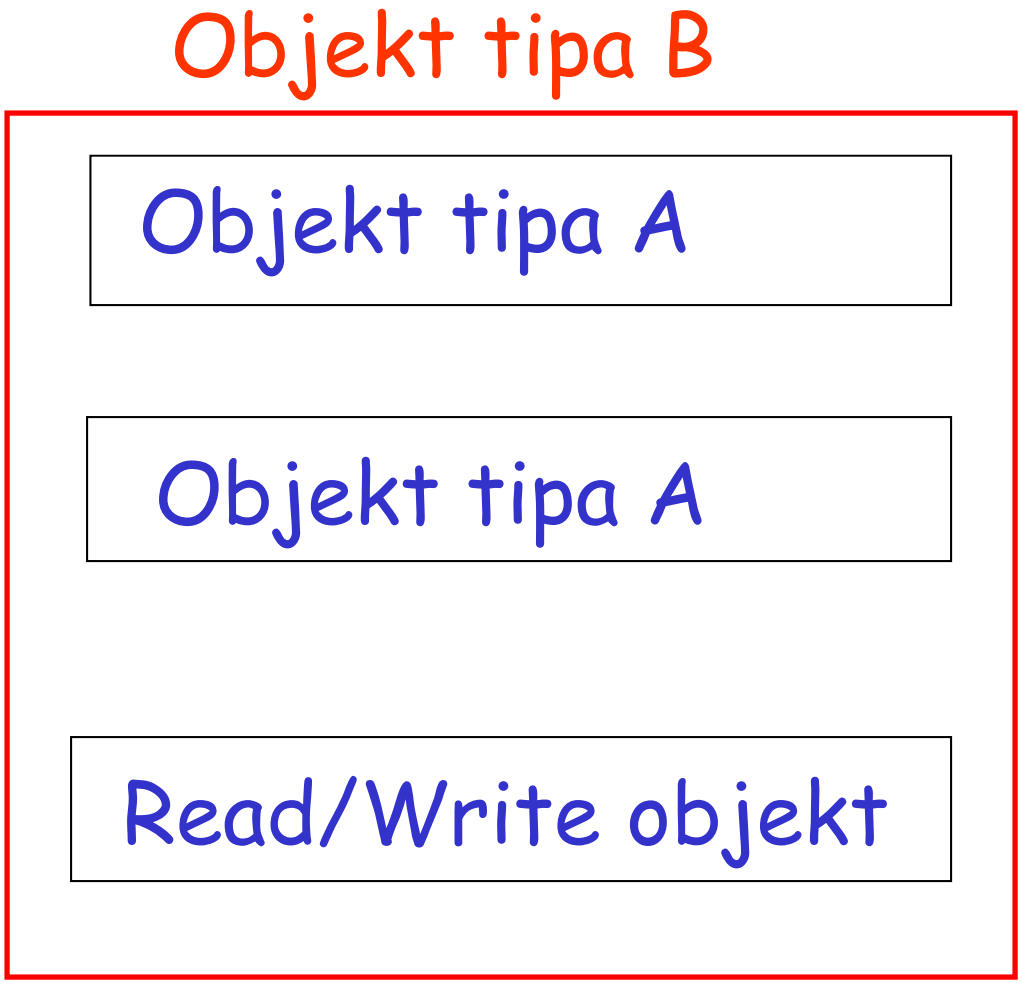
Međusobno isključivanje

```
(Shared ) boolean lock = false;

function Critical_Section() {
    while TestAndSet(lock)
        skip //spin until lock is acquired

    //Critical-section code - only one process can be in this section at a time
    begin {
       …
    }
    //end-of critical section – release lock
    lock = false //release lock when finished with the critical section
}
```

Simulacija:



Objekt tipa A simulira objekt tipa B
(koristeći pomoćne read/write objekte)

Teorema: Objekti tipa *A* sa brojem konsenzusa n ne mogu da na WF način simuliraju drugi objekt tipa B sa brojem konsenzusa $m > n$

Dokaz: Jer bi inače, objekt tipa *A* imao broj konsenzusa m

Kraj dokaza

Univerzalan objekt:

može da simulira bilo koji drugi
objekt na način oslobođen-čekanja

Primer: Compare&Swap

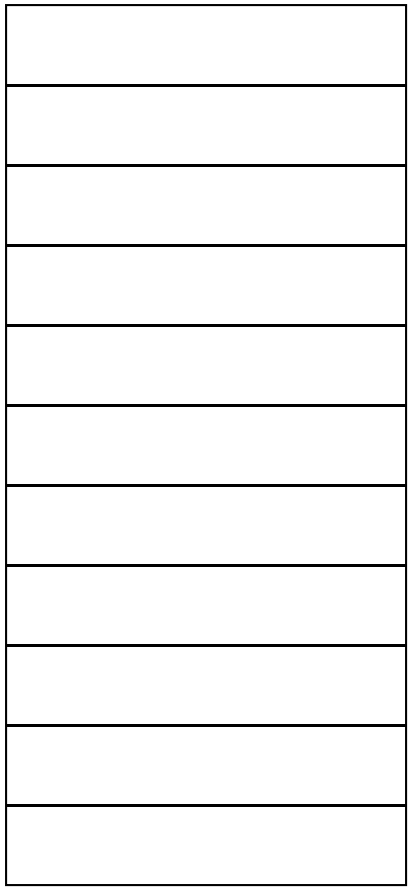
∞ (beskonačan br. konsenzusa)

Možemo da pokažemo da:

Objekti sa brojem konsenzusa *n* mogu simulirati, na WF način, bilo koji drugi objekt za do *n* procesora

Read/Write

Deljena memorija



Predpost. da se deljenoj mem može pristupati samo kroz Read ili Write operacije

Teorema: Broj konsenzusa za Read/Write objekt je 1

Dokaz teoreme:

Trivijalno, bilo koji algoritam konsenzusa sa jednim procesom koji koristi read/write promenljive je oslobođen-čekanja

Ostaje da se pokaže:

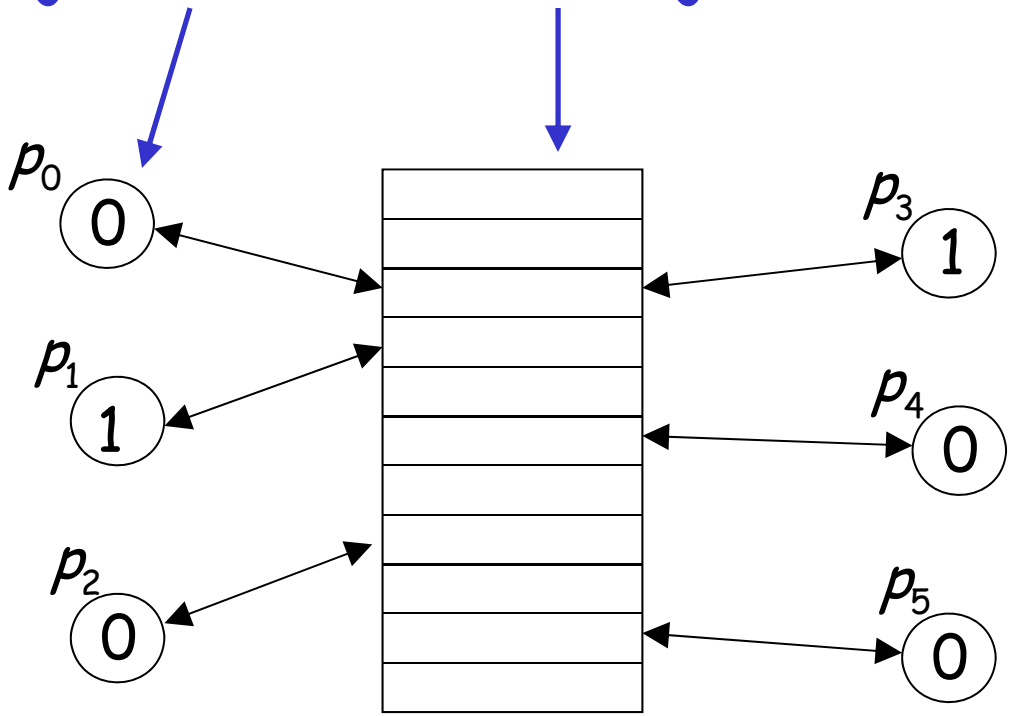
Konsenzus oslobođen-čekanja ne može biti rešen korišćenjem samo read/write objekata za $n \geq 2$ procesora

Pristup:

Pokazaćemo da bilo koji algoritam koji rešava konsenzus oslobođen-čekanja za $n \geq 2$ ima izvršenje koje se nikada ne završava

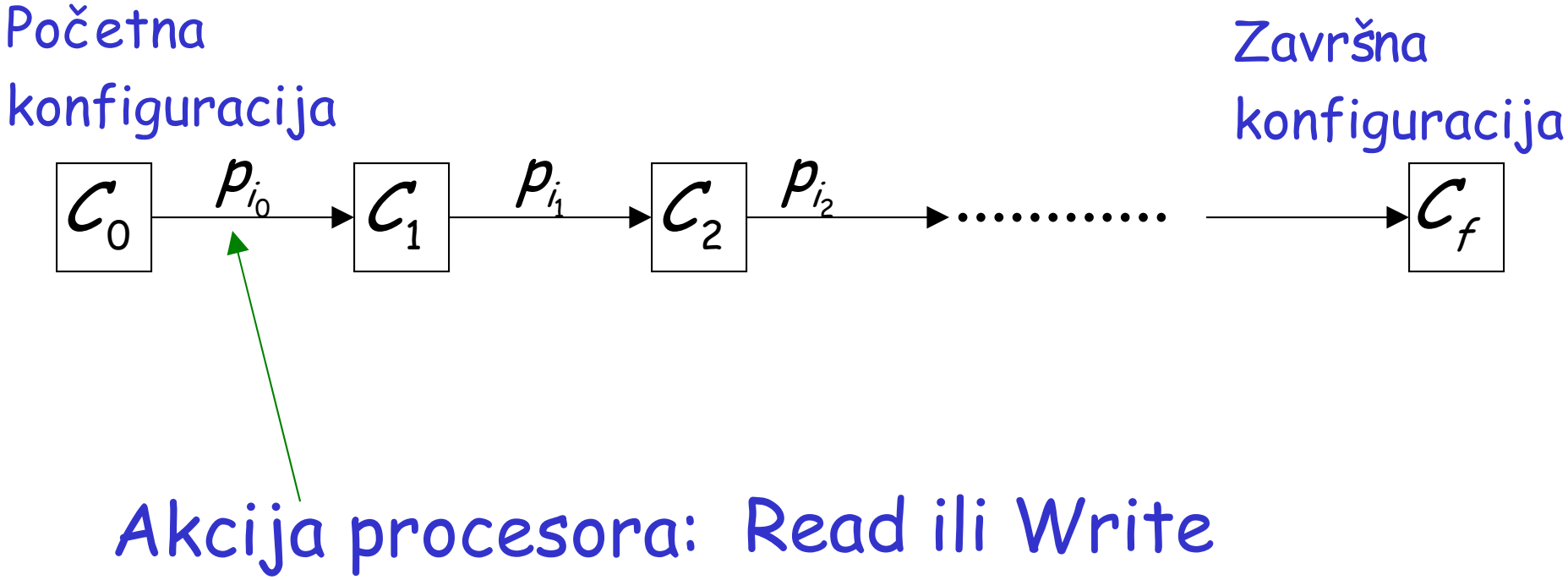
Konfiguracija sistema: C

je skup svih promenljivih u sistemu, uključujući lokalne i deljene

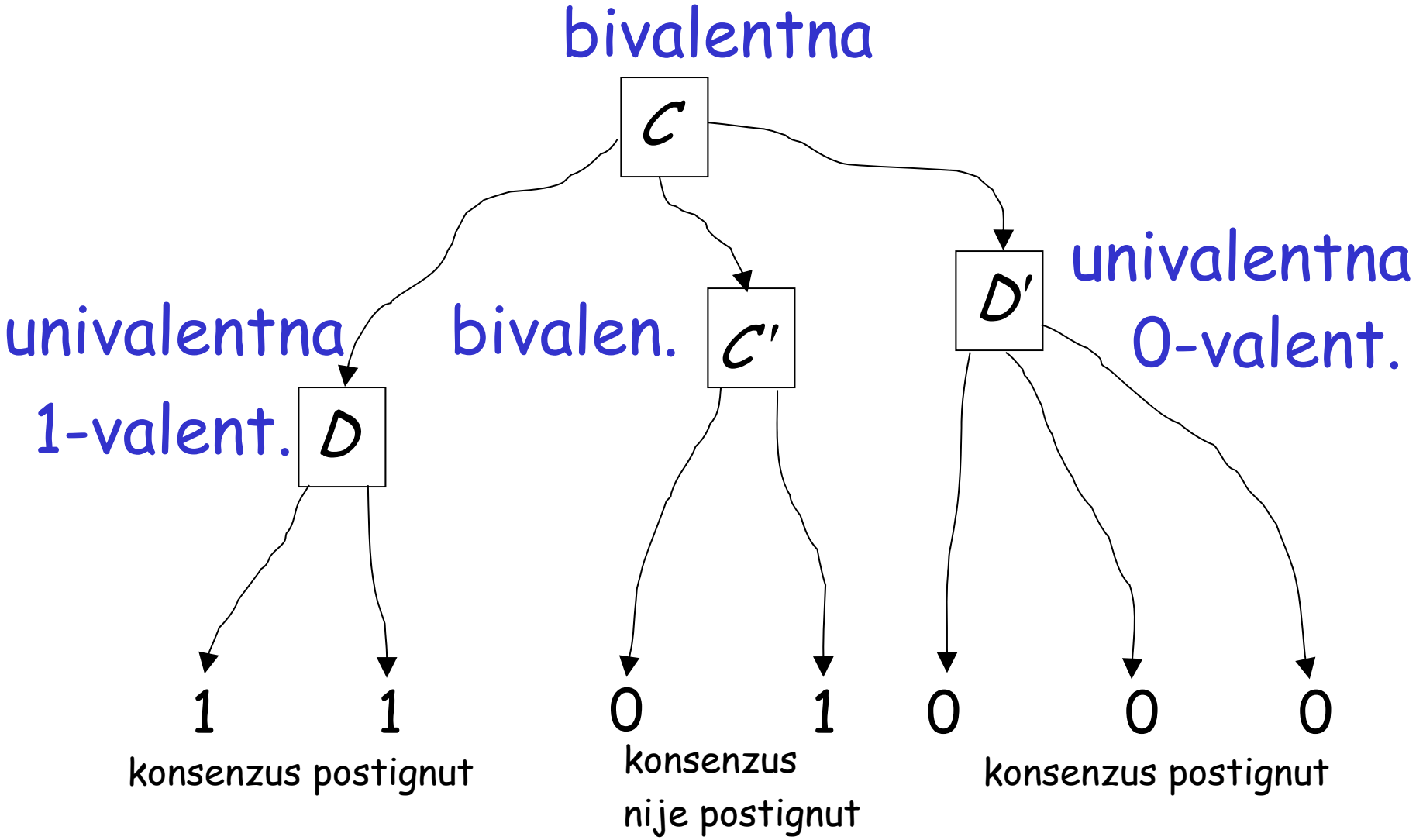


Izvršenje distribuiranog sistema se uvek može posmatrati kao:

sekvenca konfiguracija

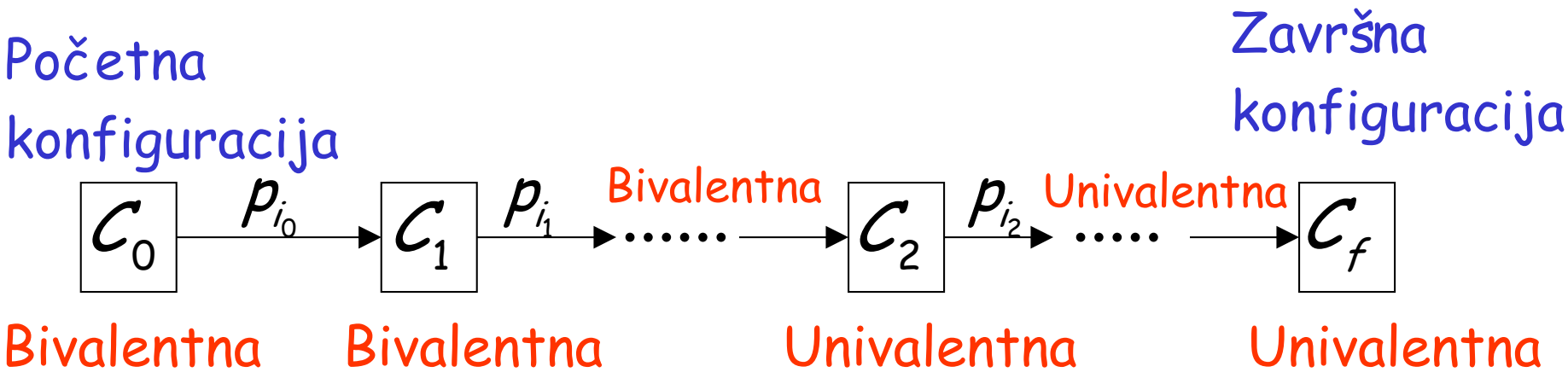


Valenca konfiguracije sistema

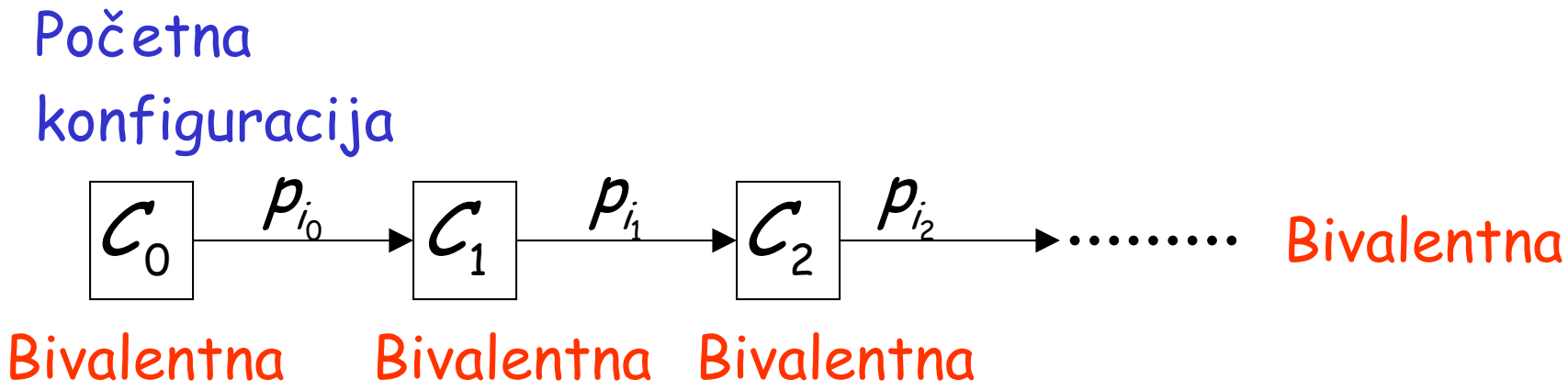


Vred konsenzusa na mogućim putanjama izvr.

Izvršenje koje se završava:



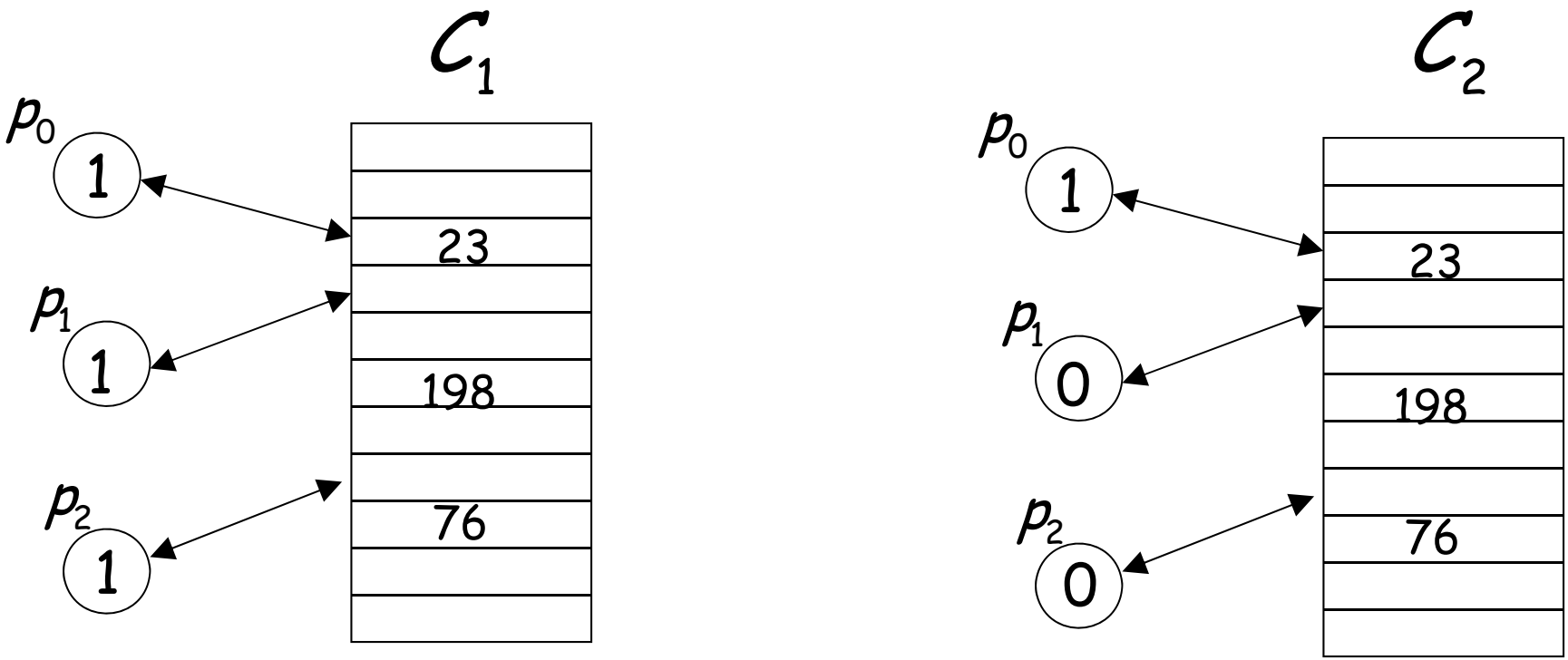
Da bi dokazali teoremu, pokazaćemo da uvek postoji izvršenje u kom je svaka konfiguracija bivalentna



Izvršenje bez završetka

Slične konfiguracije za procesor p_0

$$C_1 \overset{p_0}{\approx} C_2$$



Iste deljene promenljive

Lokalne prom od drugih se mogu razlikovati

Lema: Ako postoje univalentne konfiguracije C_1 i C_2 takve da $C_1 \overset{p_i}{\approx} C_2$

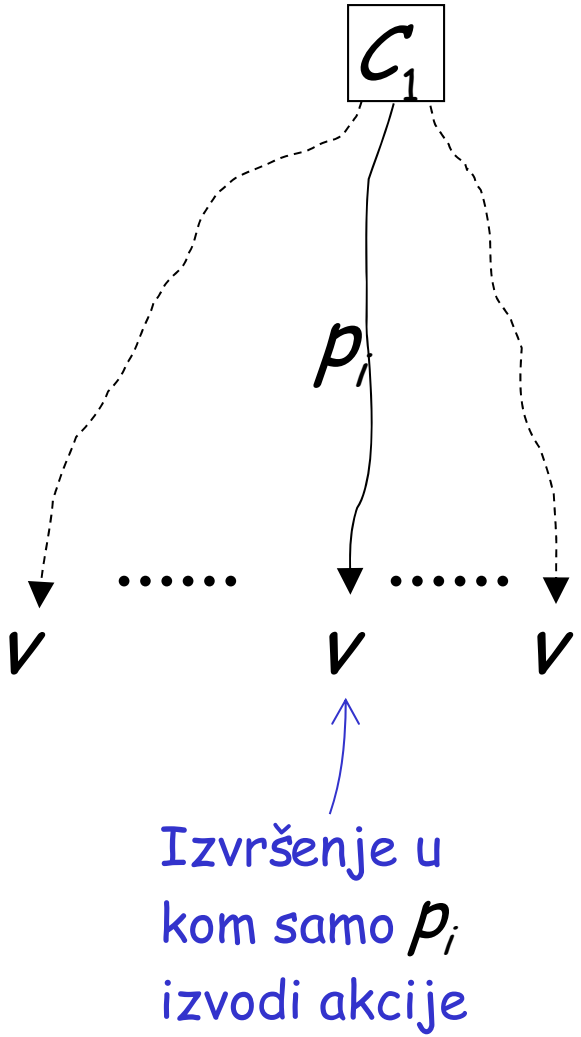
onda ako je C_1 v-valentna

onda je i C_2 v-valentna

$(v = 0 \text{ ili } 1)$

Dokaz leme:

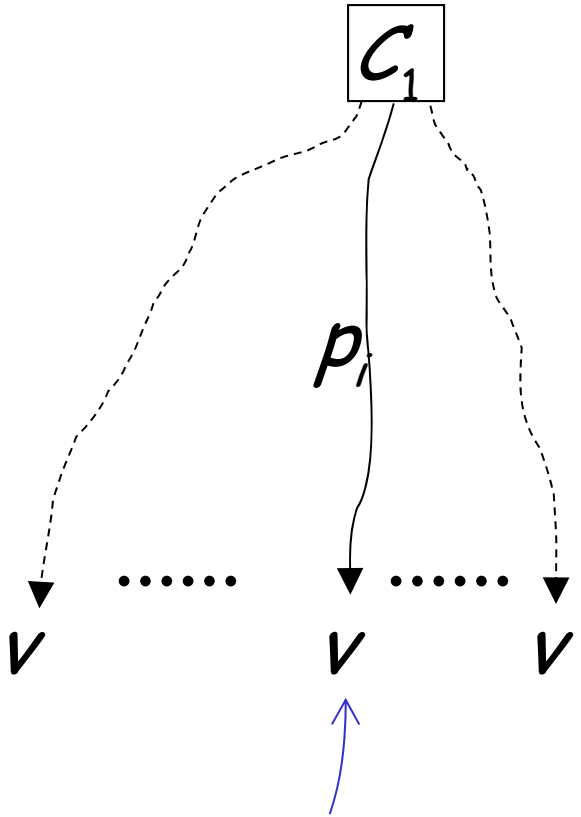
Univalentna



Sva moguća izvršenja iz C_1

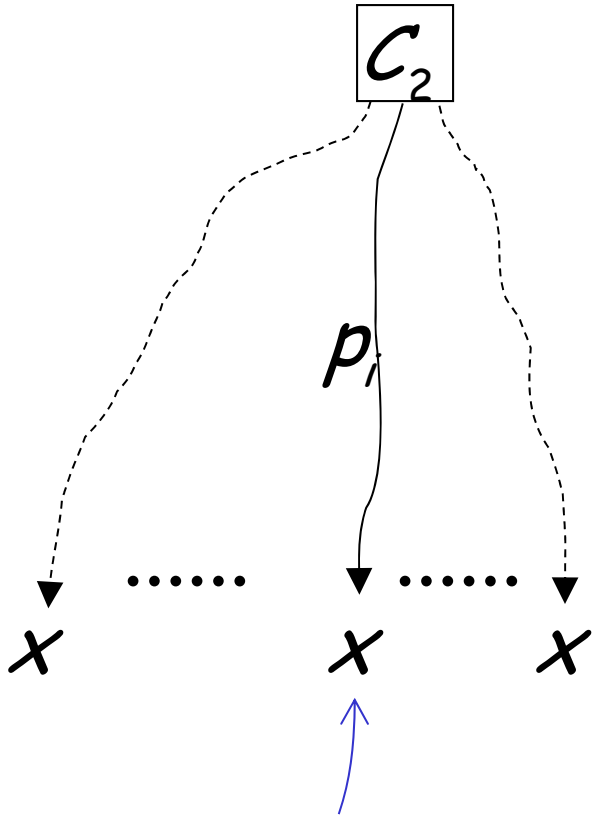
konačna odluka za svako moguće izvršenje

Univalentna



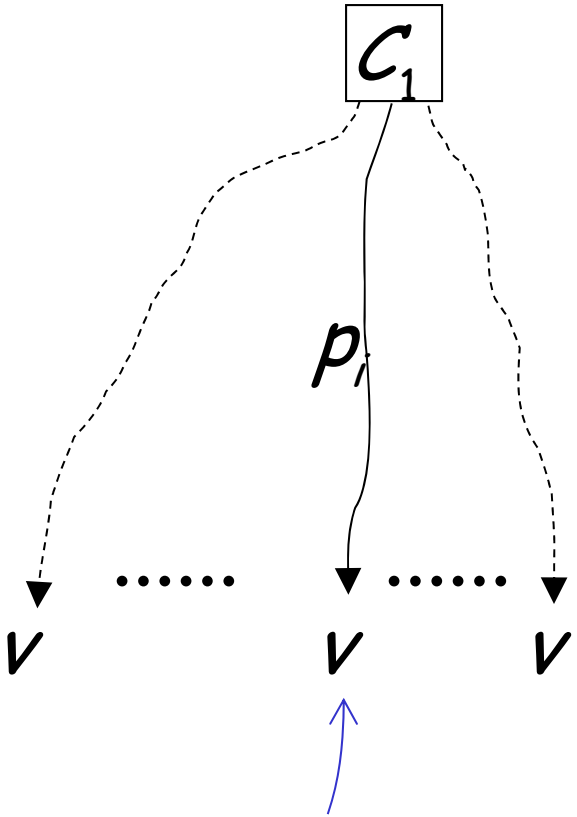
Izvršenje u kom samo p_i izvodi akcije

Univalentna



Izvršenje u kom samo p_i izvodi akcije

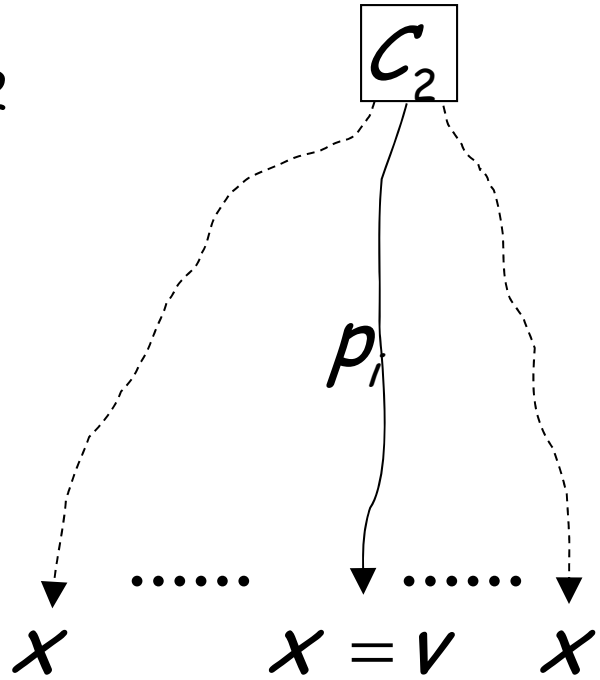
Univalentna



Izvršenje u kom samo p_i izvodi akcije

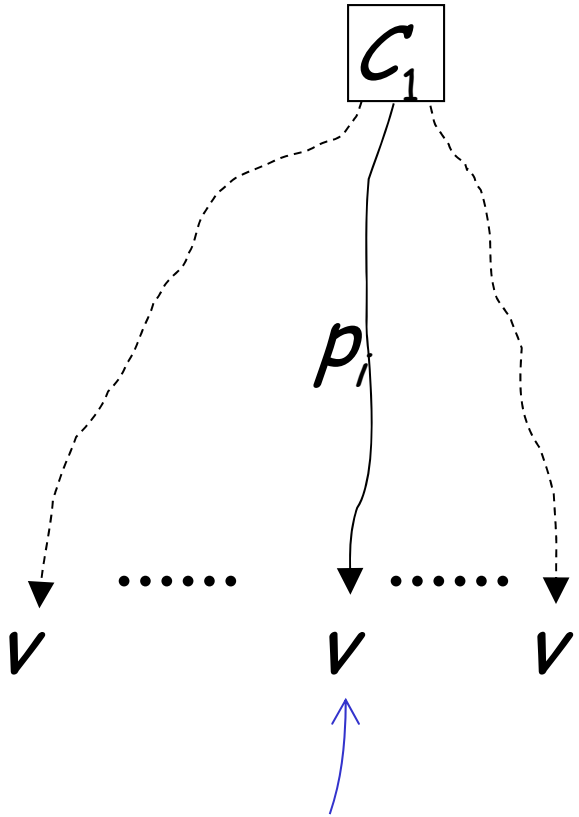
$$C_1 \overset{p_i}{\approx} C_2$$

Univalentna



Izvršenje u kom samo p_i izvodi akcije

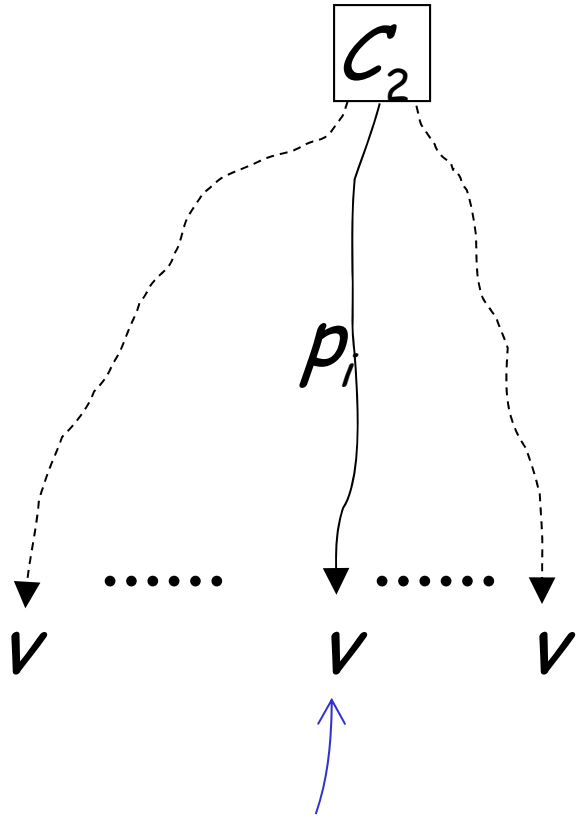
Univalentna



Izvršenje u kom samo p_i izvodi akcije

$C_1 \overset{p_i}{\approx} C_2$

Univalentna



Izvršenje u kom samo p_i izvodi akcije

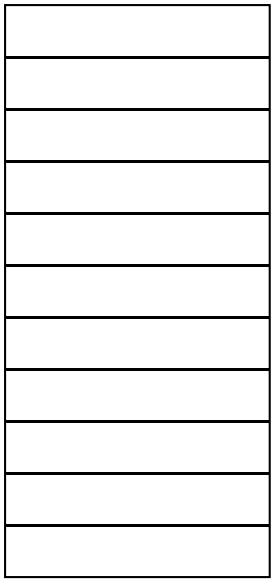
Kraj dokaza leme

Lema: Postoji bivalentna početna konfiguracija

Dokaz leme:

Moguće početne konfiguracije

Deljena memorija



Prazna

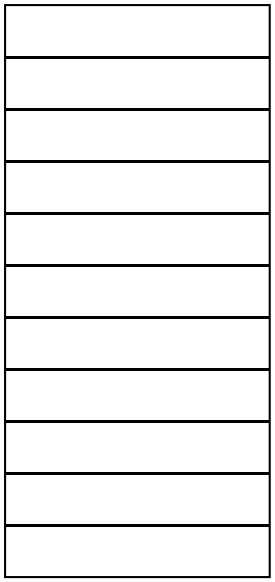
Lokalna memorija

Početna konfiguracija

	I_0	I_{01}	I_1
p_0	0	0	1
p_1	0	1	1
	⋮	⋮	⋮
p_{n-1}	0	1	1

Moguće početne konfiguracije

Deljena memorija



Prazna

Lokalna memorija

Početna konfiguracija

	I_0	I_{01}	I_1
p_0	0	0	1
p_1	0	1	1
⋮	⋮	⋮	⋮
p_{n-1}	0	1	1
	0-valentna	?	1-valentna

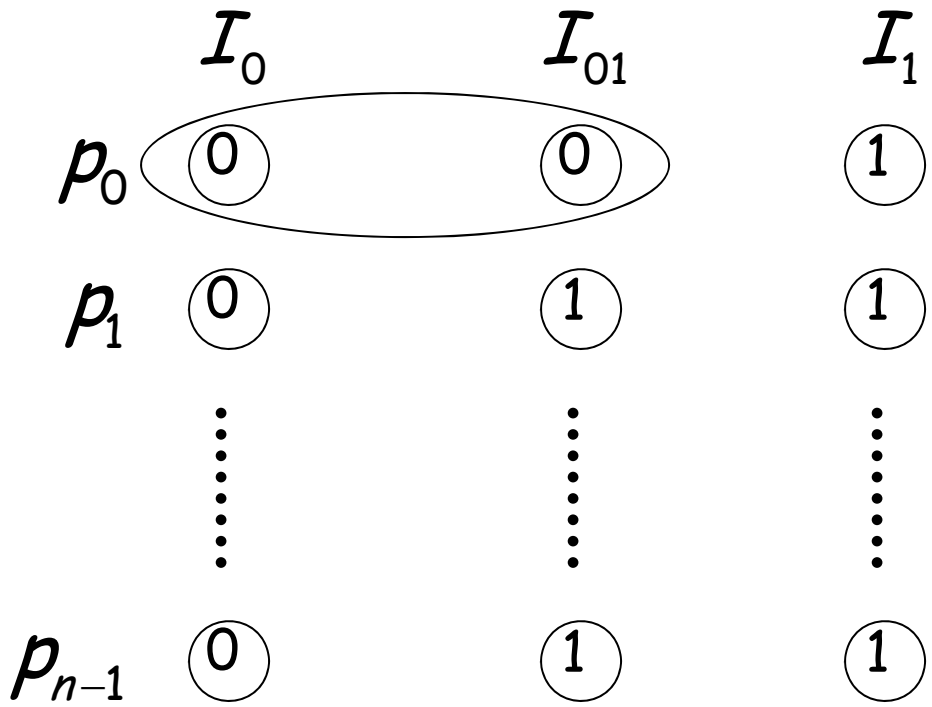
Moguće početne konfiguracije

Deljena memorija

Lokalna memorija

Prazna

Početna konfiguracija



0-valentna 1-valentna? 1-valentna

Ne, pošto je $I_0 \overset{p_0}{\approx} I_{01}$

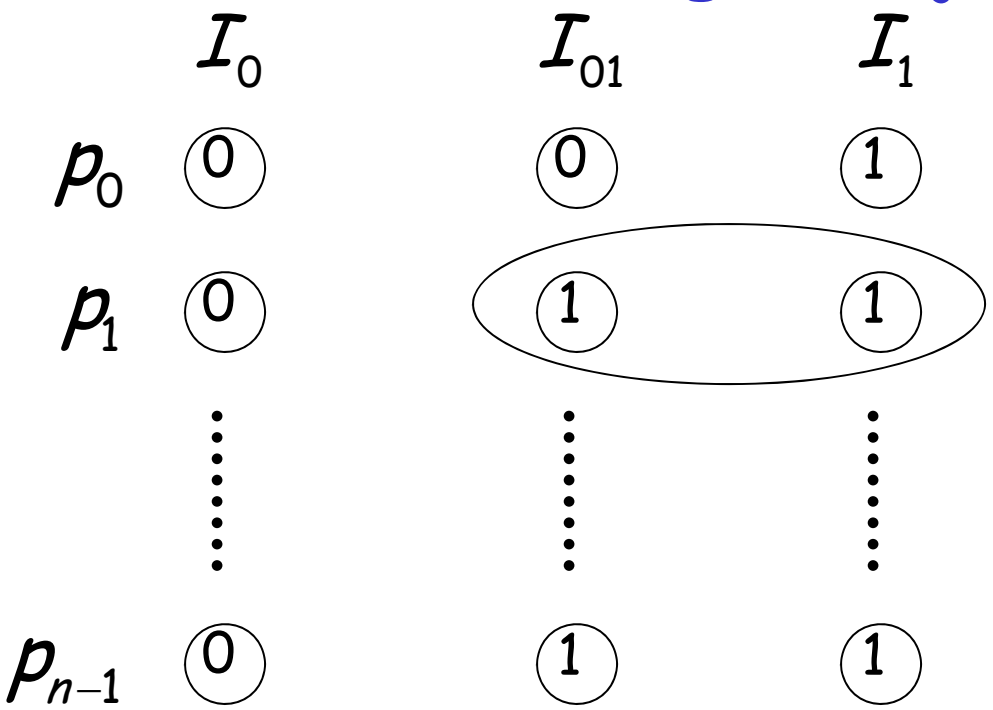
Moguće početne konfiguracije

Deljena memorija

Lokalna memorija

Prazna

Početna konfiguracija

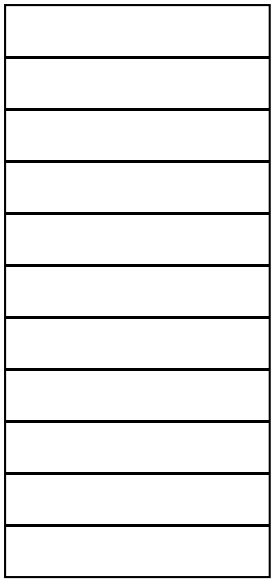


0-valentna 0-valentna? 1-valentna

Ne, pošto je $I_{01} \overset{p_1}{\approx} I_1$

Moguće početne konfiguracije

Deljena memorija



Prazna

Lokalna memorija

Početna konfiguracija

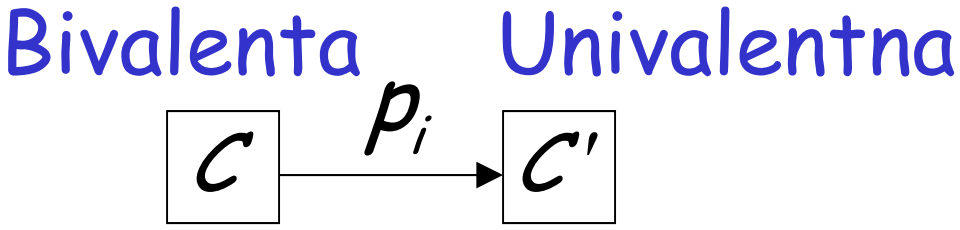
	I_0	I_{01}	I_1
p_0	0	0	1
p_1	0	1	1
	⋮	⋮	⋮
p_{n-1}	0	1	1

0-valentna bivalentna 1-valentna

Kraj dokaza leme

Kritičan procesor za datu konfiguraciju:

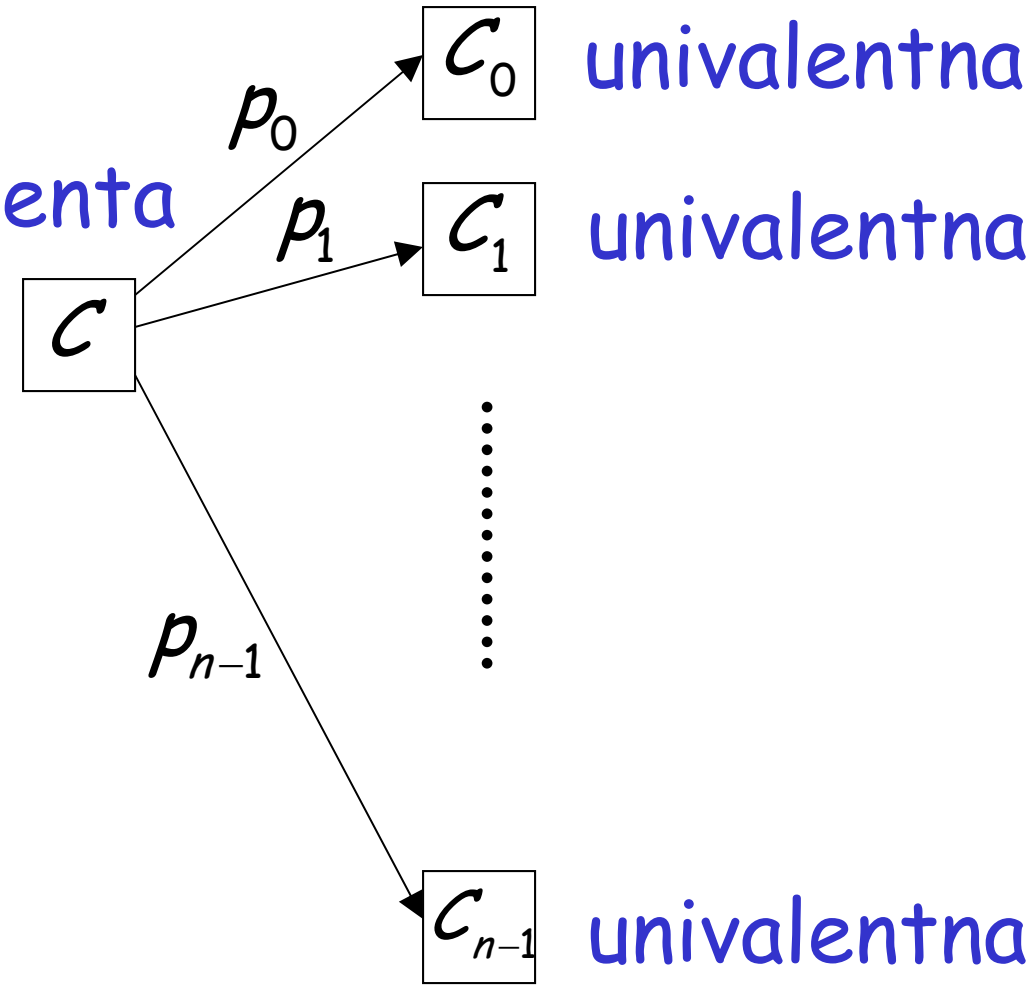
ta konfiguracija je bivalentna,
i nakon što procesor izvede korak,
konfiguracija postaje univalentna



Lema: Ako je C bivalentna konfiguracija, onda postoji bar jedan procesor koji nije kritičan

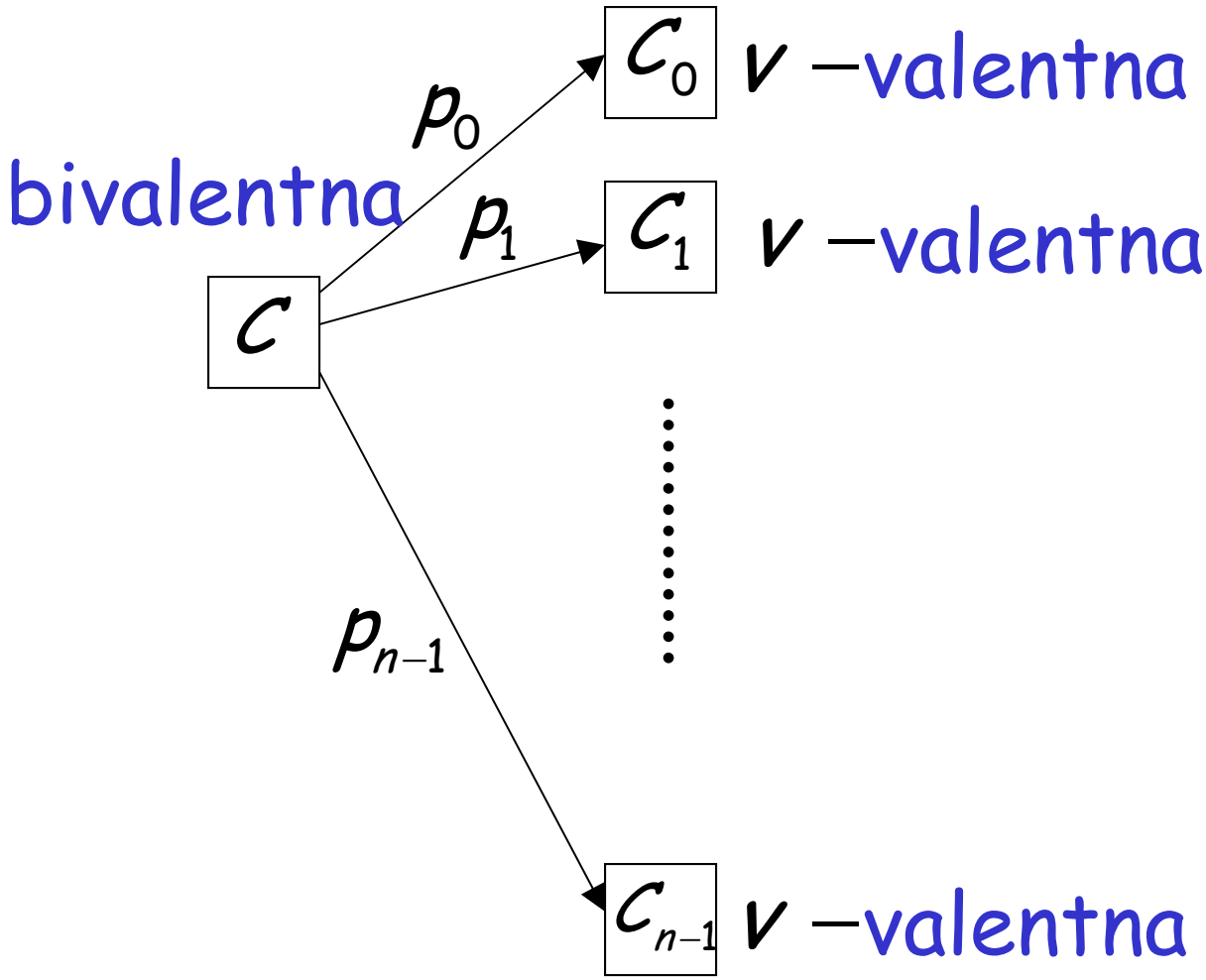
Dokaz leme:

Predpost. radi kontradikcije da su svi procesori kritični



Ne može biti da svi imaju istu valencu

$(v = 0 \text{ ili } 1)$

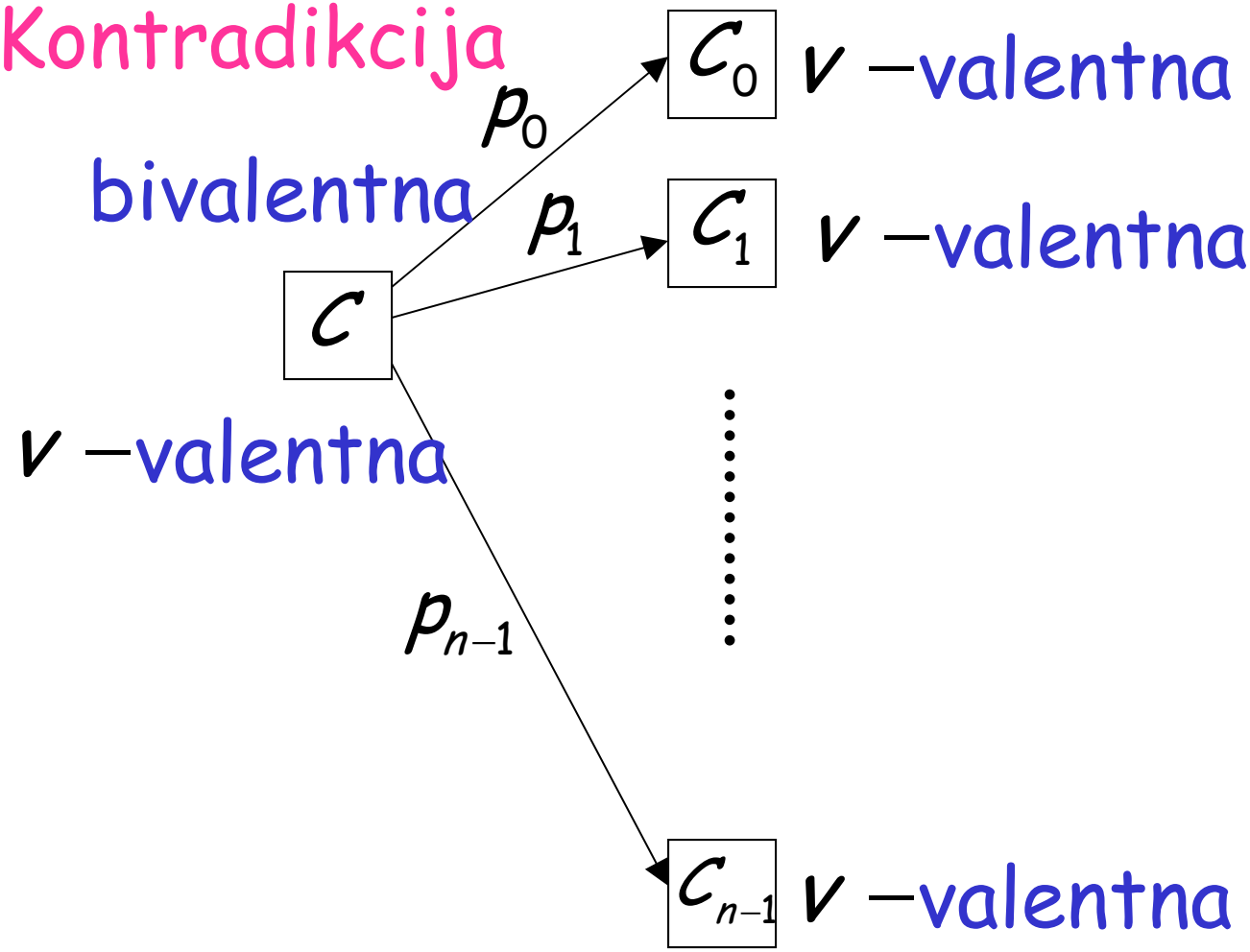


Ne može biti da svi imaju istu valencu

$(v = 0 \text{ ili } 1)$

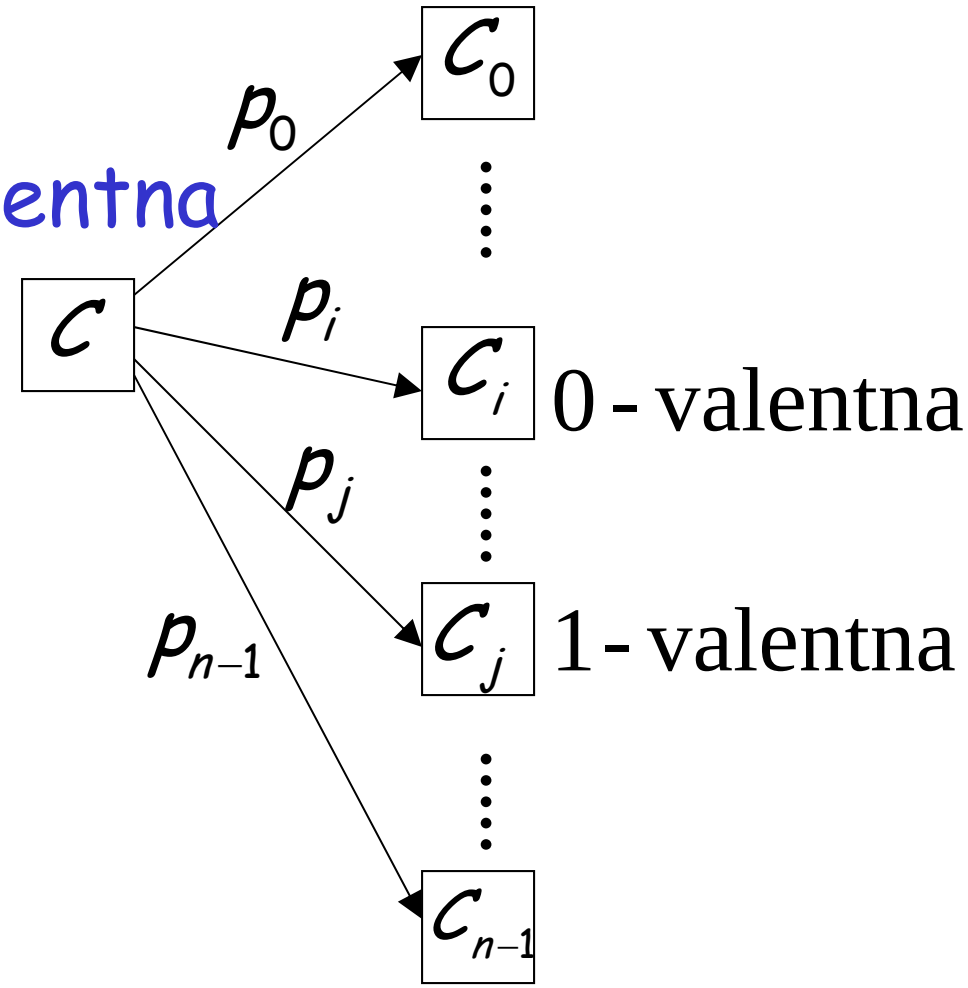
Kontradikcija

bivalentna

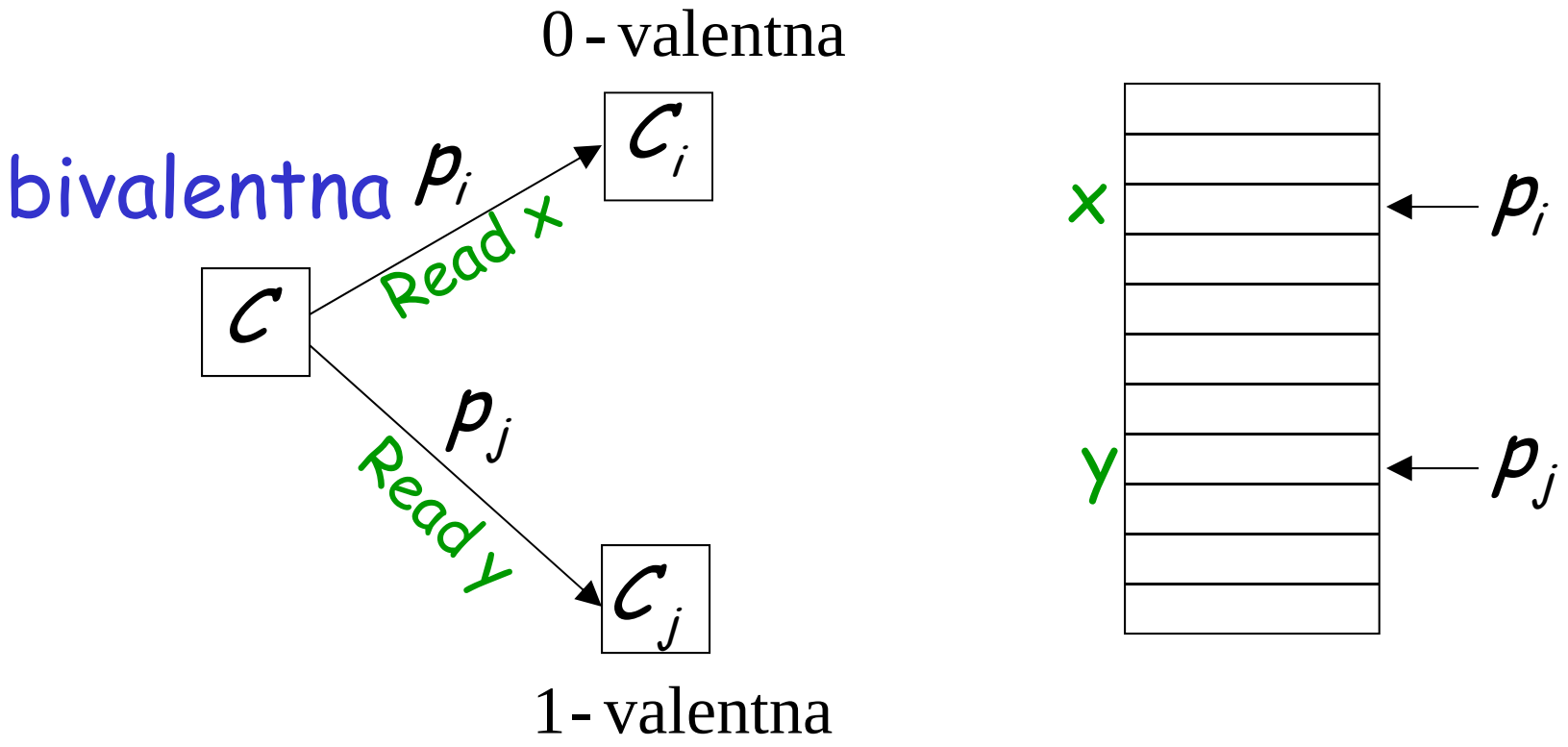


Moraju postojati dva procesora sa različitim valencama

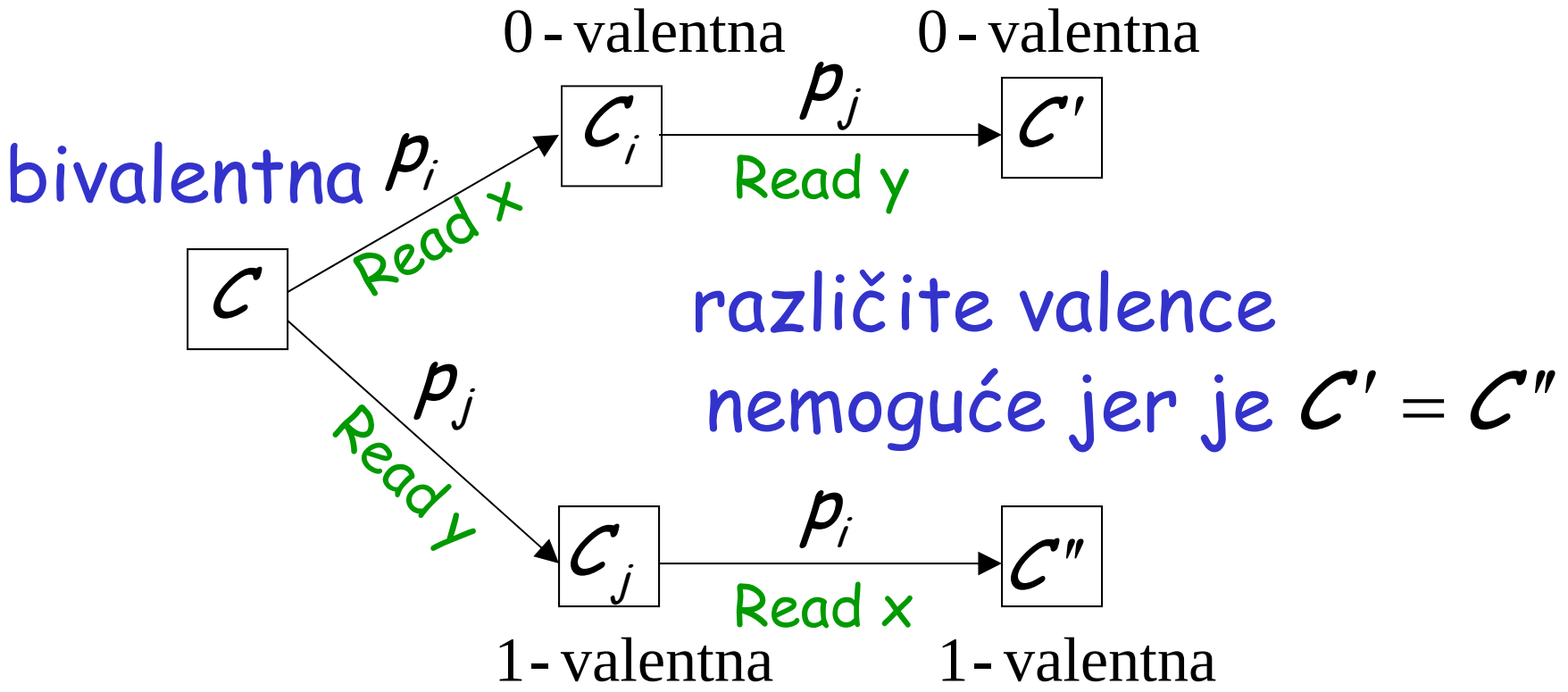
bivalentna



Sluč. 1: predpost. da oni pristupaju različitim deljenim promenljivama



dva moguća izvršenja



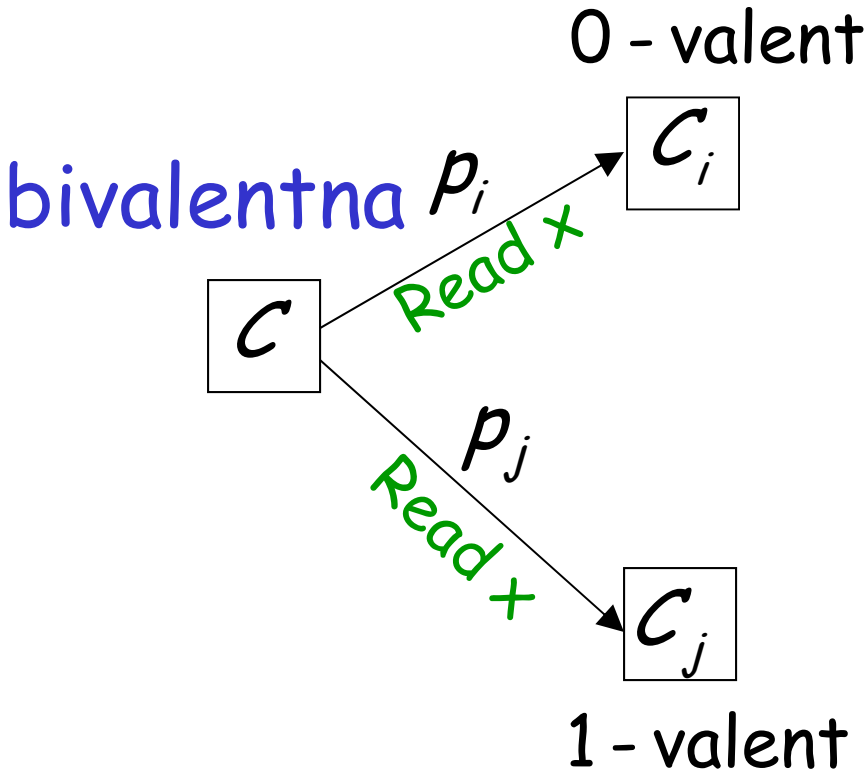
različite valence

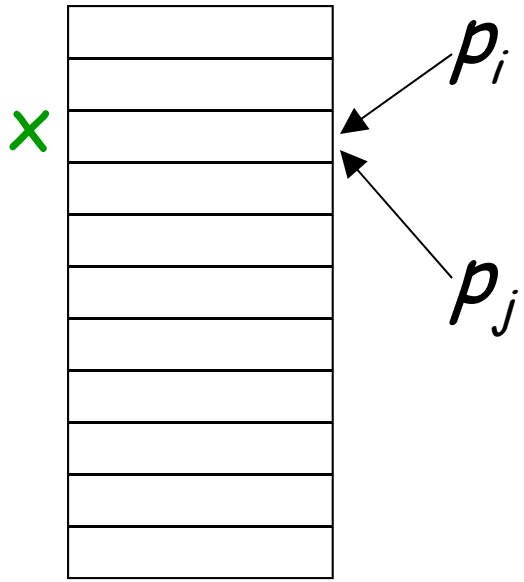
nemoguće jer je $C' = C''$

isti rezultat važi za bilo koju vrstu
operacija (Read ili Write)
koje procesori primenjuju na x i y

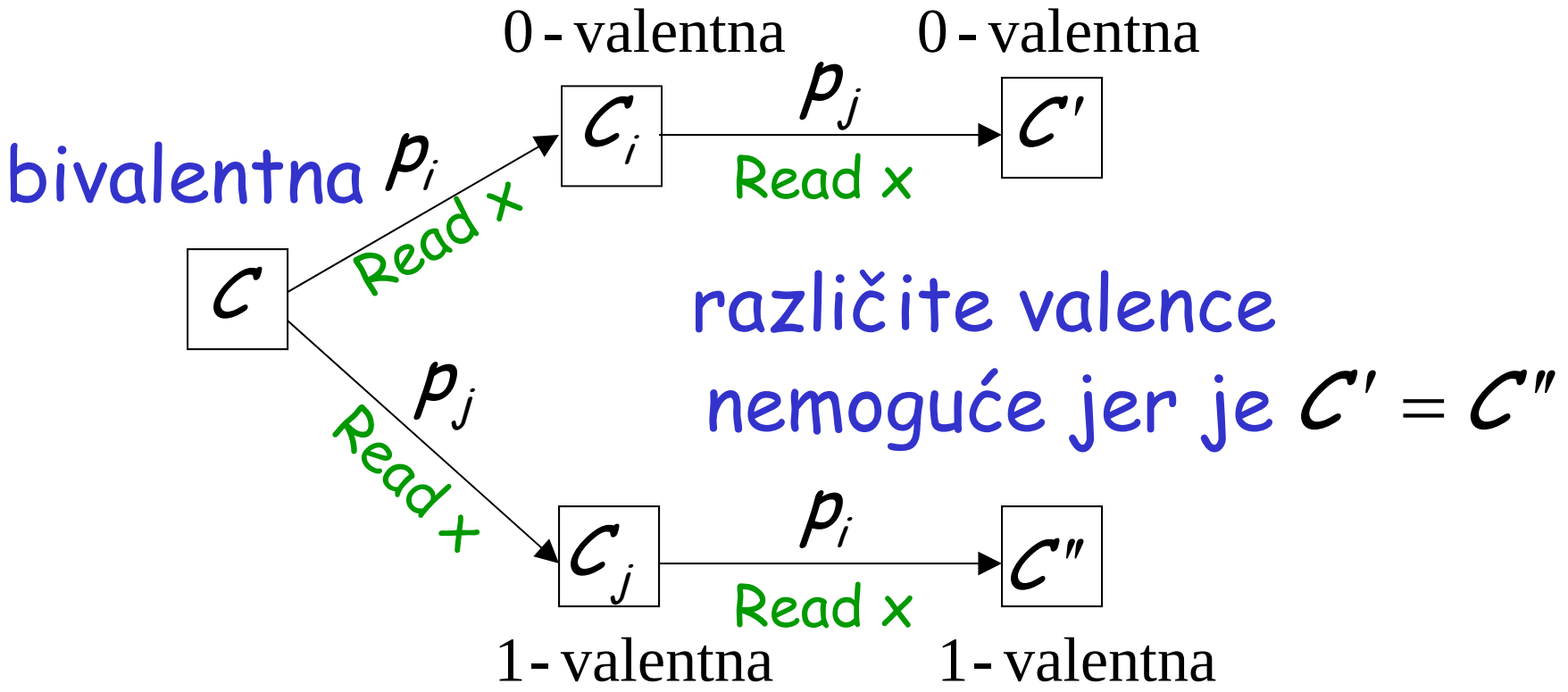
Sluč. 2: predost. da oni pristupaju istoj deljenoj promenljivoj

podslučaj: read/read





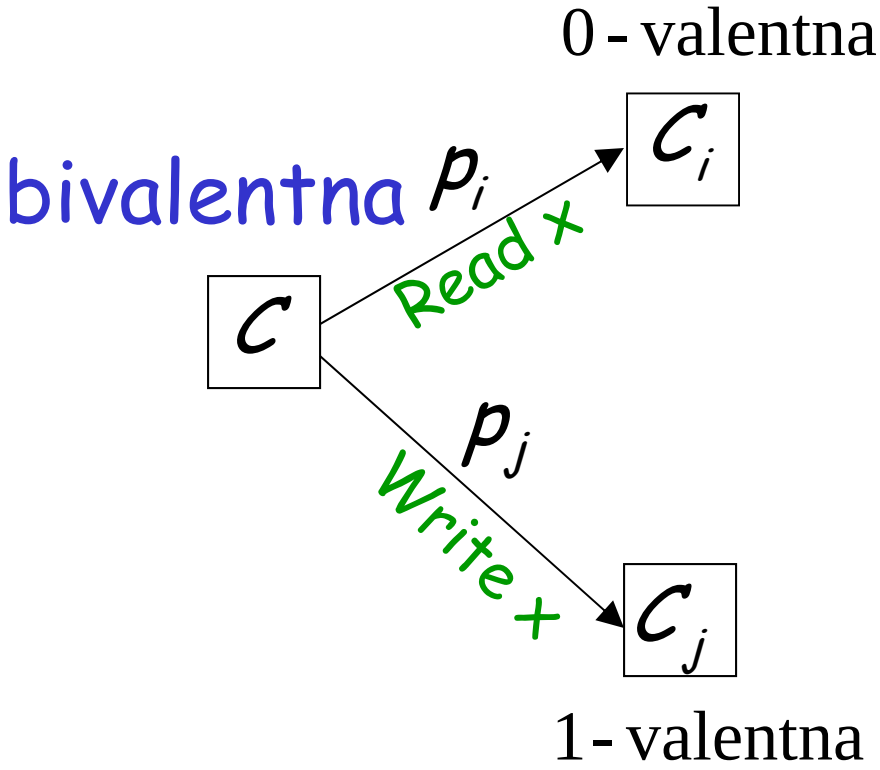
dva moguća izvršenja



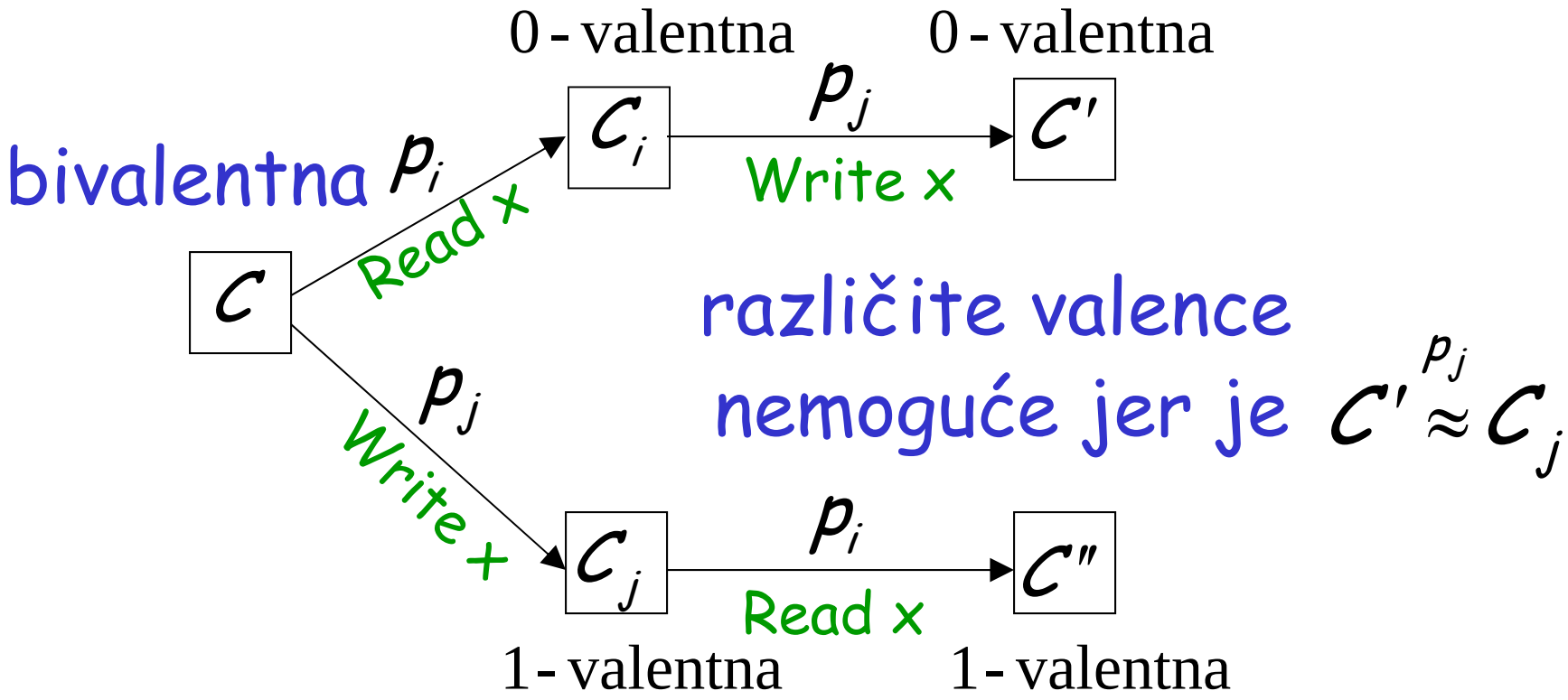
različite valence

nemoguće jer je $C' = C''$

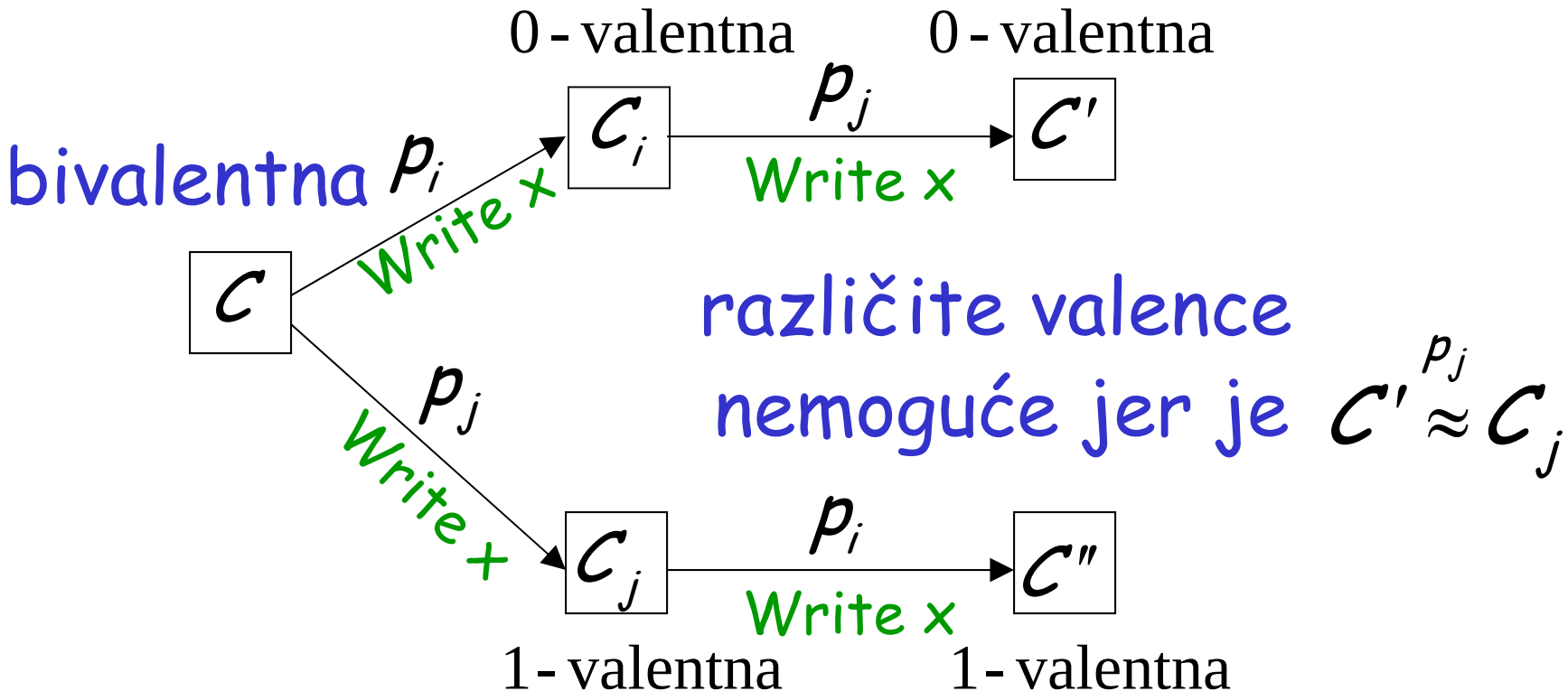
podslučaj: read/write



dva moguća izvršenja



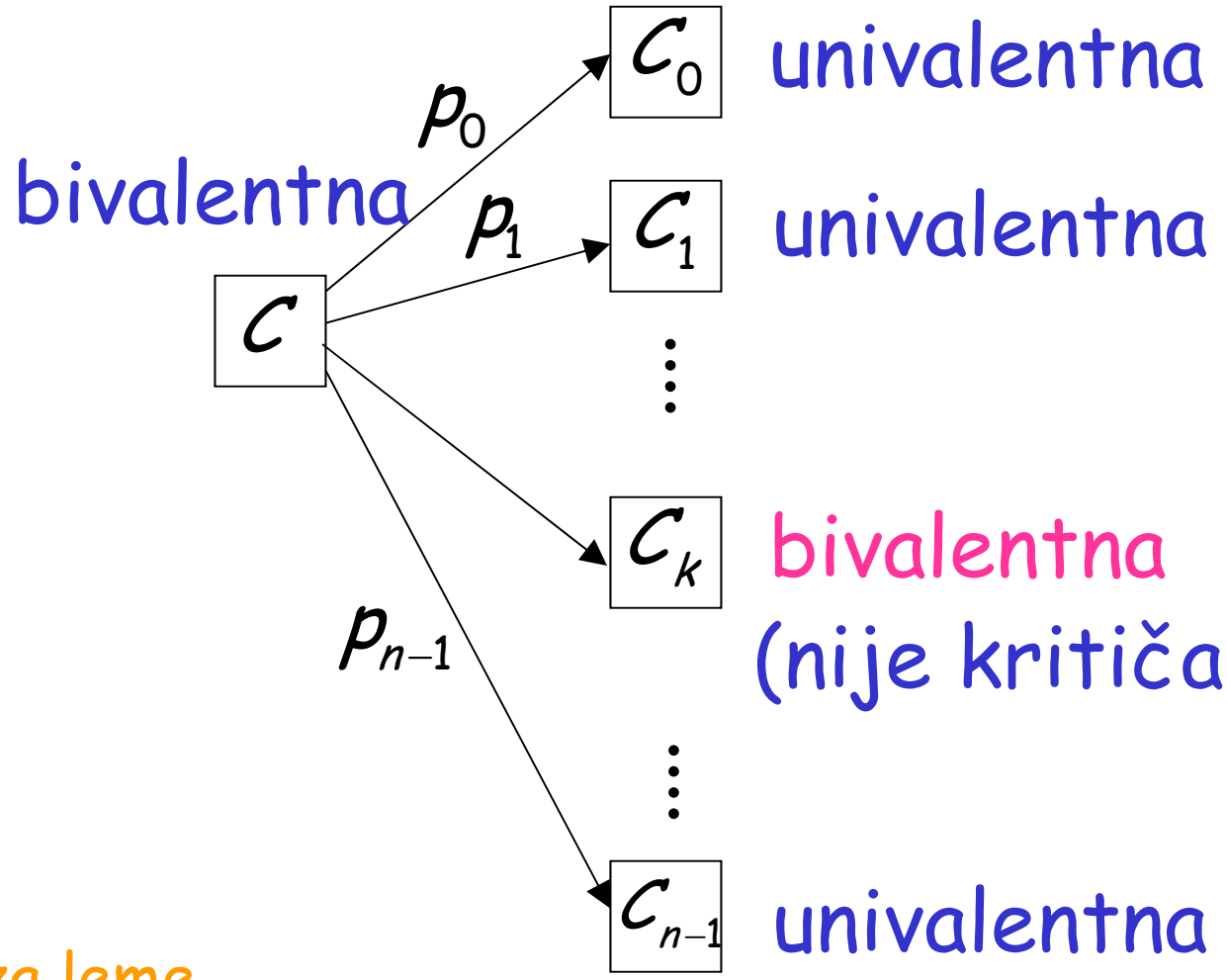
podslučaj: write/write



različite valence

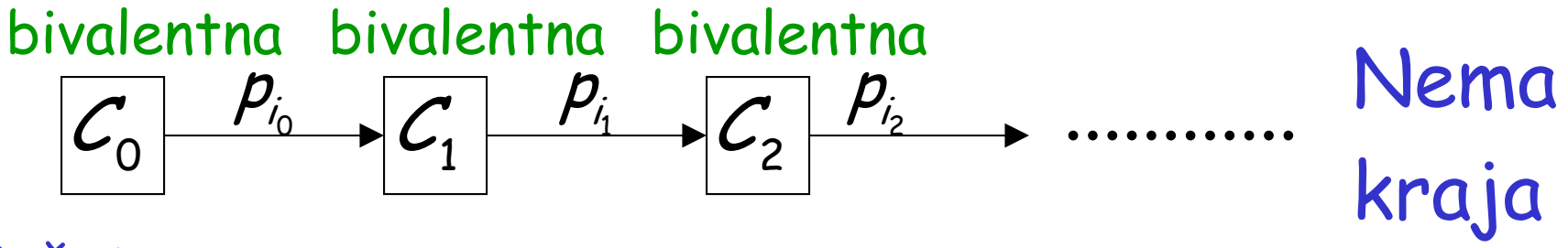
nemoguće jer je $C' \overset{p_j}{\approx} C_j$

U svim slučajima smo dobili kontradikciju
Zbog toga, postoji neki procesor
koji nije kritičan



Kraj dokaza leme

Prema tome, možemo konstruisati neko izvršenje



Konsenzus se nikada ne može postići

Kraj dokaza teoreme