

DISTRIBUIRANI ALGORITMI I SISTEMI

Read/Write deljene promenljive

2

- U jednom atomskom koraku procesor može
 - ▣ da očita vrednost promenljive ili
 - ▣ da upiše novu vrednost u promenljivu
 - ▣ ali ne može da izvrši obe operacije istovremeno!

Algoritam sa ceduljicama (Bakery Algorithm)

3

- To je algoritam za
 - ▣ međusobno isključivanje
 - ▣ bez trajnog zaključavanja
- koristi $2n$ deljenih read/write promenljivih
 - ▣ Bulove prom. `Choosing[i]` : inicijalno false, u koje upisuje p_i a čitaju ih drugi
 - ▣ Celi br. `Number[i]` : inicijalno 0, u koje upisuje p_i a čitaju ih drugi

Algoritam sa ceduljicama

4

Kod za ulaznu sekciju:

```
Choosing[i] := true
Number[i] := max{Number[0], ..., Number[n-1]} + 1
Choosing[i] := false
for j := 0 to n-1 (except i) do
    wait until Choosing[j] = false
    wait until Number[j] = 0 or
        (Number[j], j) > (Number[i], i)
endfor
```

Kod za izlaznu sekciju:

```
Number[i] := 0
```

Algoritam sa ceduljicama obezbeđuje međusobno isključivanje

5

Lema (3.5): Ako je p_i u kritičnoj sekciji i
 $\text{Number}[k] \neq 0$ ($k \neq i$), onda
 $(\text{Number}[k], k) > (\text{Number}[i], i)$.

Dokaz: Razmotrimo dva slučaja:

p_i u KS i
 $\text{Number}[k] \neq 0$

Algoritam sa ceduljicama obezbeđuje međusobno isključivanje

5

Lema (3.5): Ako je p_i u kritičnoj sekciji i

$\text{Number}[k] \neq 0$ ($k \neq i$), onda

$(\text{Number}[k], k) > (\text{Number}[i], i)$.

Dokaz: Razmotrimo dva slučaja:

p_i -jevo najnovije
čitanje $\text{Number}[k]$;

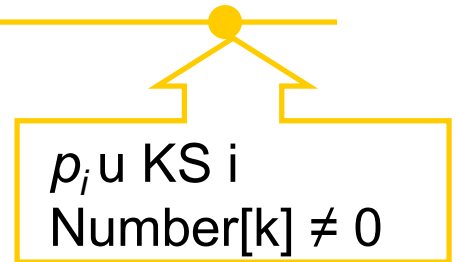
Slučaj 1: vraća 0

Slučaj 2: $(\text{Number}[k], k) > (\text{Number}[i], i)$

p_i u KS i
 $\text{Number}[k] \neq 0$

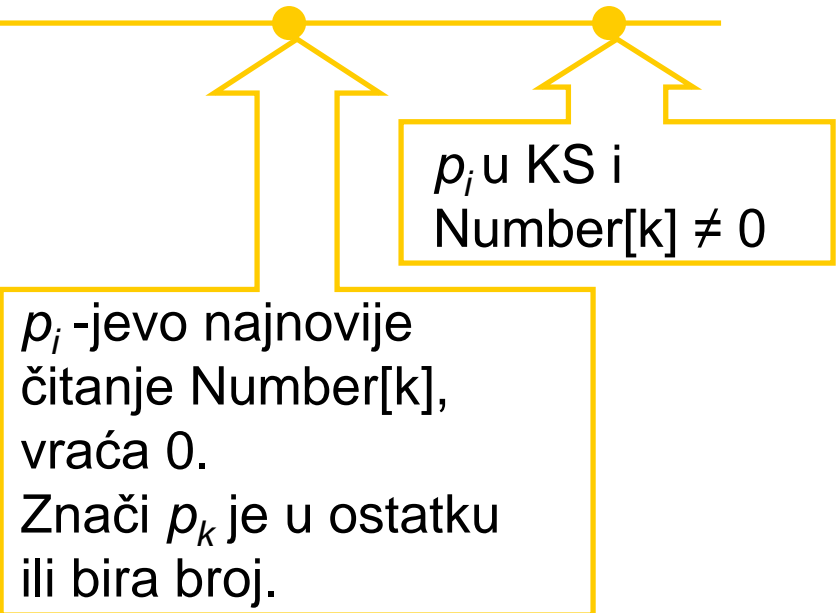
Međusobno isključivanje: Slučaj 1

6



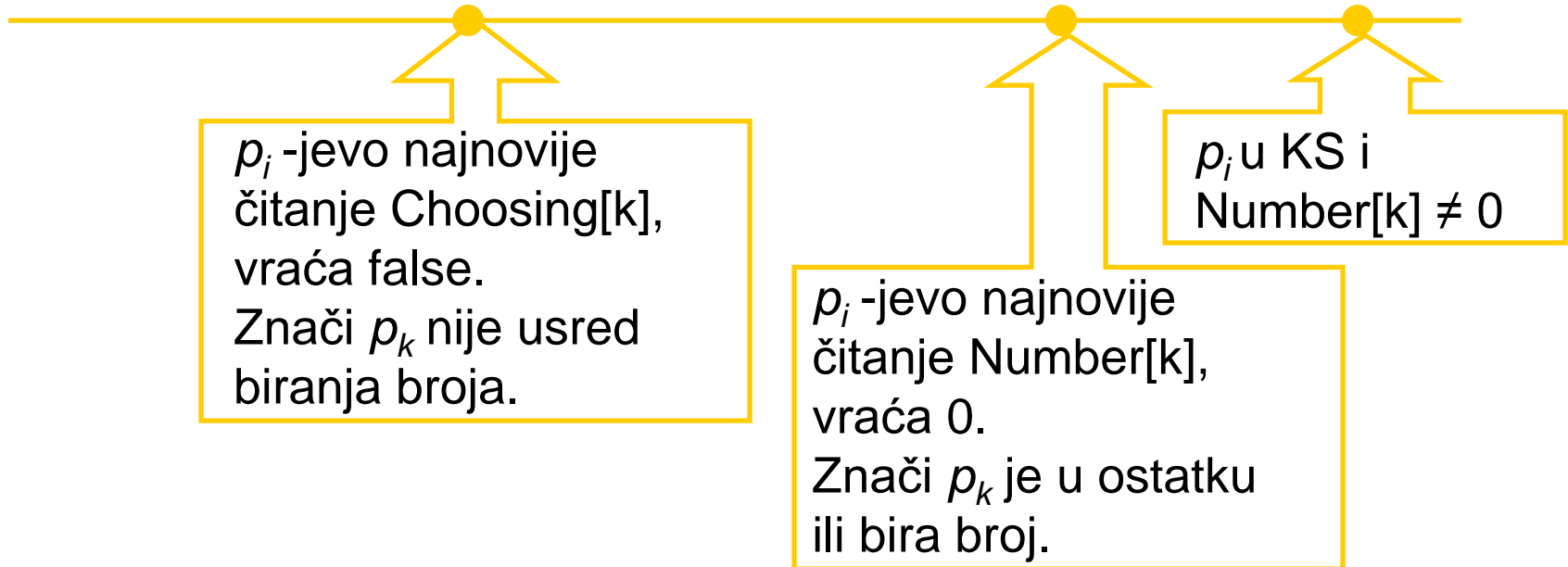
Međusobno isključivanje: Slučaj 1

6



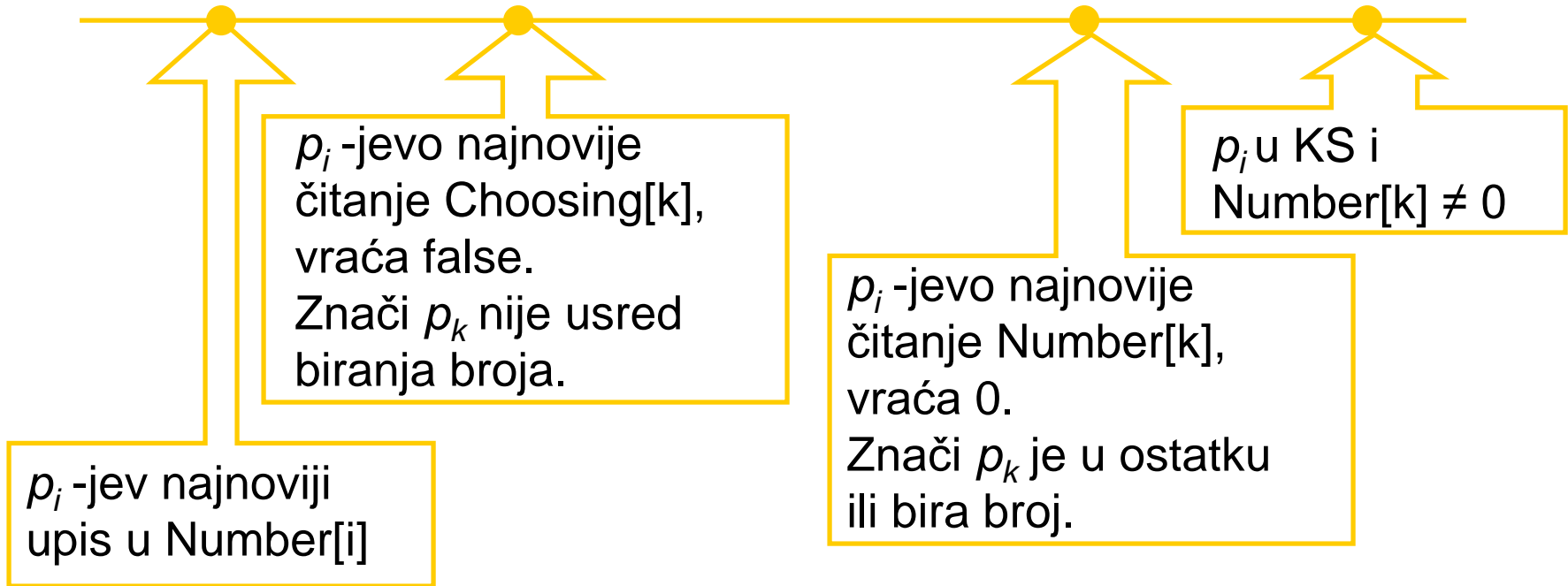
Međusobno isključivanje: Slučaj 1

6



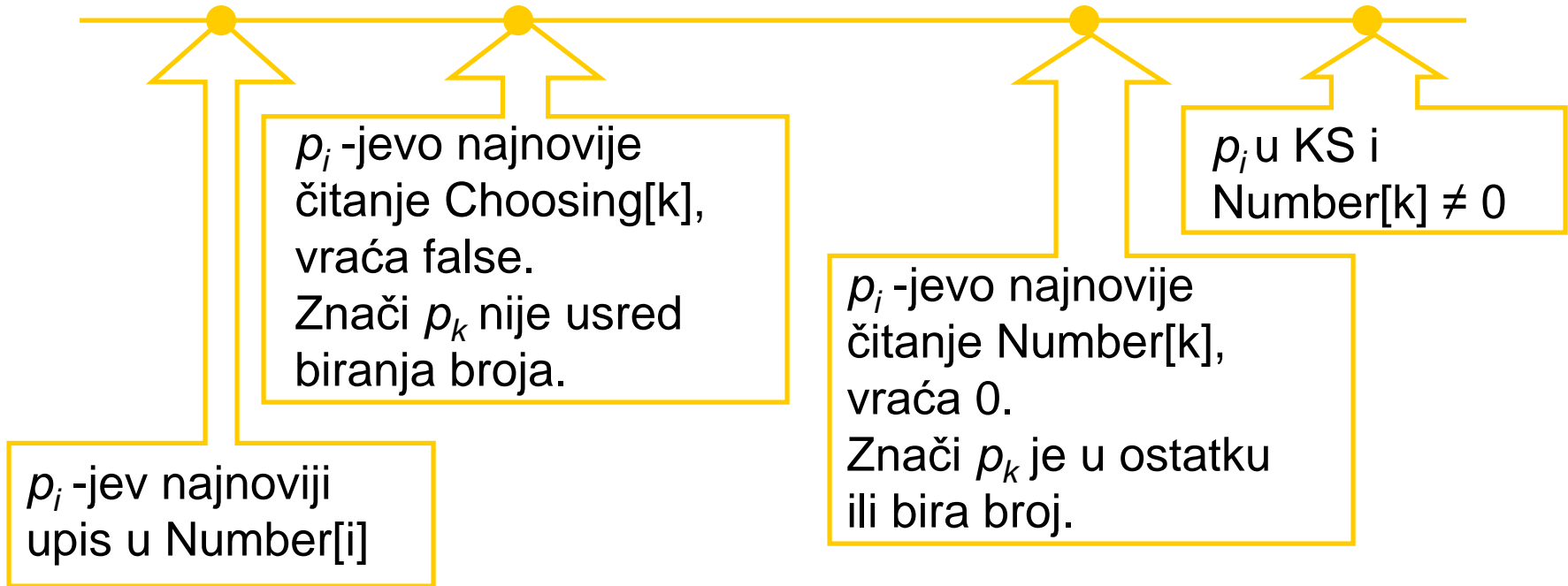
Međusobno isključivanje: Slučaj 1

6



Međusobno isključivanje: Slučaj 1

6



Znači p_k bira broj u ovom intervalu, pogleda p_i -jev broj, i izabira veći broj.

Međusobno isključivanje: Slučaj 2

7

- Se dokazuje korišćenjem argumenata sličnih onima za Slučaj 1.

Međusobno isključivanje za algoritam sa ceduljicama

8

- **Lema (3.6):** Ako je p_i u kritičnoj sekciji, onda $\text{Number}[i] > 0$.
 - Dokaz je na osnovu neposredne indukcije.
- **Međusobno isključivanje:** Predpos. da su p_i i p_k istovremeno u KS.
 - Po Lemi 3.6, oba imaju $\text{Number} > 0$.
 - Po Lemi 3.5,
 - $(\text{Number}[k], k) > (\text{Number}[i], i)$ i
 - $(\text{Number}[i], i) > (\text{Number}[k], k)$

Međusobno isključivanje za algoritam sa ceduljicama

8

- **Lema (3.6):** Ako je p_i u kritičnoj sekciji, onda $\text{Number}[i] > 0$.
 - Dokaz je na osnovu neposredne indukcije.
- **Međusobno isključivanje:** Predpos. da su p_i i p_k istovremeno u KS.
 - Po Lemi 3.6, oba imaju $\text{Number} > 0$.
 - Po Lemi 3.5,
 - $(\text{Number}[k],k) > (\text{Number}[i],i)$ i
 - $(\text{Number}[i],i) > (\text{Number}[k],k)$

Kontradikcija!

Algoritam sa ceduljicama:

Nema trajnog zaključavanja

9

- Pred. radi kontradikcije da postoji blokiran procesor.
- Blokirani procesori čekaju u Liniji 5 ili 6 (iskazi čekanja), *ne čekaju* dok biraju broj.
- Neka je p_i blokiran procesor sa najmanjim parom (`Number[i],i`).
- Bilo koji procesor koji ulazi u krit. sekc. nako što je p_i izabrao svoj broj, bira veći broj.
- Svaki procesor sa manjim brojem konačno ulazi u KS (ne ostaje blokiran) i izlazi iz nje.
- Zato p_i ne može da čeka u Liniji 5 ili 6.

Algoritam sa ceduljicama:

Nema trajnog zaključavanja

9

- Pred. radi kontradikcije da postoji blokiran procesor.
- Blokirani procesori čekaju u Liniji 5 ili 6 (iskazi čekanja), *ne čekaju* dok biraju broj.
- Neka je p_i blokiran procesor sa najmanjim parom (`Number[i],i`).
- Bilo koji procesor koji ulazi u krit. sekc. nako što je p_i izabrao svoj broj, bira veći broj.
- Svaki procesor sa manjim brojem konačno ulazi u KS (ne ostaje blokiran) i izlazi iz nje.
- Zato p_i ne može da čeka u Liniji 5 ili 6.

Kontradikcija!

Prostorna složenost algoritma sa cedunjicama

10

- Broj deljenih promenljivih je $2n$
- Choosing su Bulove promenljive
- Number promenljive su neograničene
- Da li je moguće da algoritam koristi manje deljenog prostora?

ME algoritam: Ograničen prostor, 2- procesora

11

Koristi 3 binarne read/write deljene promenljive:

- $W[0]$: inicijalno 0, u nju upisuje p_0 a čita je p_1
- $W[1]$: inicijalno 0, u nju upisuje p_1 a čita je p_0
- `Priority` : inicijalno 0, oba procesora je čitaju i pišu

ME algoritam bez međusobnog blokiranja (ND = No Deadlock)

12

- Krećemo sa ograničenim algoritmom za 2 procesora i ND uslovom, zatim proširujemo na NL uslov, i na kraju proširujemo na n procesora.
- Neke ideje iz algoritma sa 2 procesora:
 - ▣ svaki procesor ima deljenu Bul. prom. $W[i]$ koja ukazuje da li on želi da uđe u KS
 - ▣ p_0 uvek ima prednost u odnosu na p_1 ; asimetričan kod

ME algoritam bez međusobnog blokiranja

13

Kod za ulaznu sekciju p_0 :

```
1   .  
2   .  
3   W[0] := 1  
4   .  
5   .  
6   wait until W[1] = 0
```

Kod za izlaznu sekciju p_0 :

```
7   .  
8   W[0] := 0
```

ME algoritam bez međusobnog blokiranja

14

Kod za ulaznu sekciju p_1 :

```
1  W[1] := 0
2  wait until W[0] = 0
3  W[1] := 1
4  .
5      if (W[0] = 1) then goto Line 1
6  .
```

Kod za izlaznu sekciju p_1 :

```
7  .
8  W[1] := 0
```

Diskusija ME algoritam bez međusobnog blokiranja

15

- Obezbeđuje međusobno isključivanje: procesori koriste W promenljive da bi ovo obezbedili
- Obezbeđuje da nema međusob. blokiranja (vežba)
- Ali nije fer (moguće trajno zaključavanje)
- Ispravka ovog nedostatka: procesori naizmenično dobijaju veći prioritet:
 - ▣ deljena promenljiva `Priority`, oba procesora je čitaju i pišu

ME algoritam bez međusobnog blok. i bez trajnog zaključavanja

16

Kod za ulaznu sekciju:

```
1  W[i] := 0
2  wait until W[1-i] = 0 or Priority = i
3  W[i] := 1
4  if (Priority = 1-i) then
5      if (W[1-i] = 1) then goto Line 1
6  else wait until (W[1-i] = 0)
```

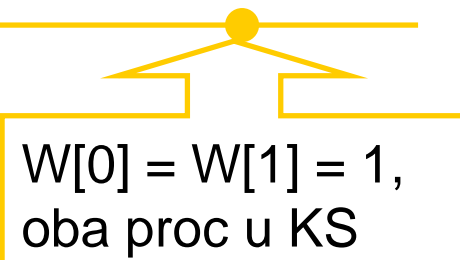
Kod za izlaznu sekciju:

```
7  Priority := 1-i
8  W[i] := 0
```

Analiza ME algoritma: međusobno isključivanje

17

- **Međusobno isključivanje:** Pret. radi kontradikcije da su p_0 i p_1 istovremeno u KS.



$W[0] = W[1] = 1,$
oba proc u KS

Analiza ME algoritma: međusobno isključivanje

17

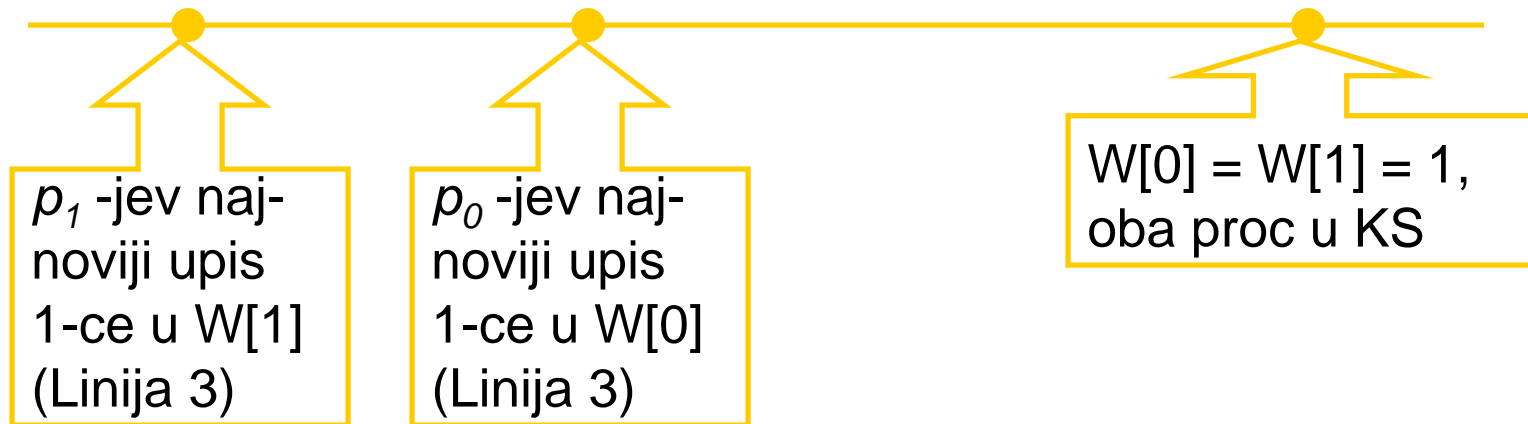
- **Međusobno isključivanje:** Pret. radi kontradikcije da su p_0 i p_1 istovremeno u KS.



Analiza ME algoritma: međusobno isključivanje

17

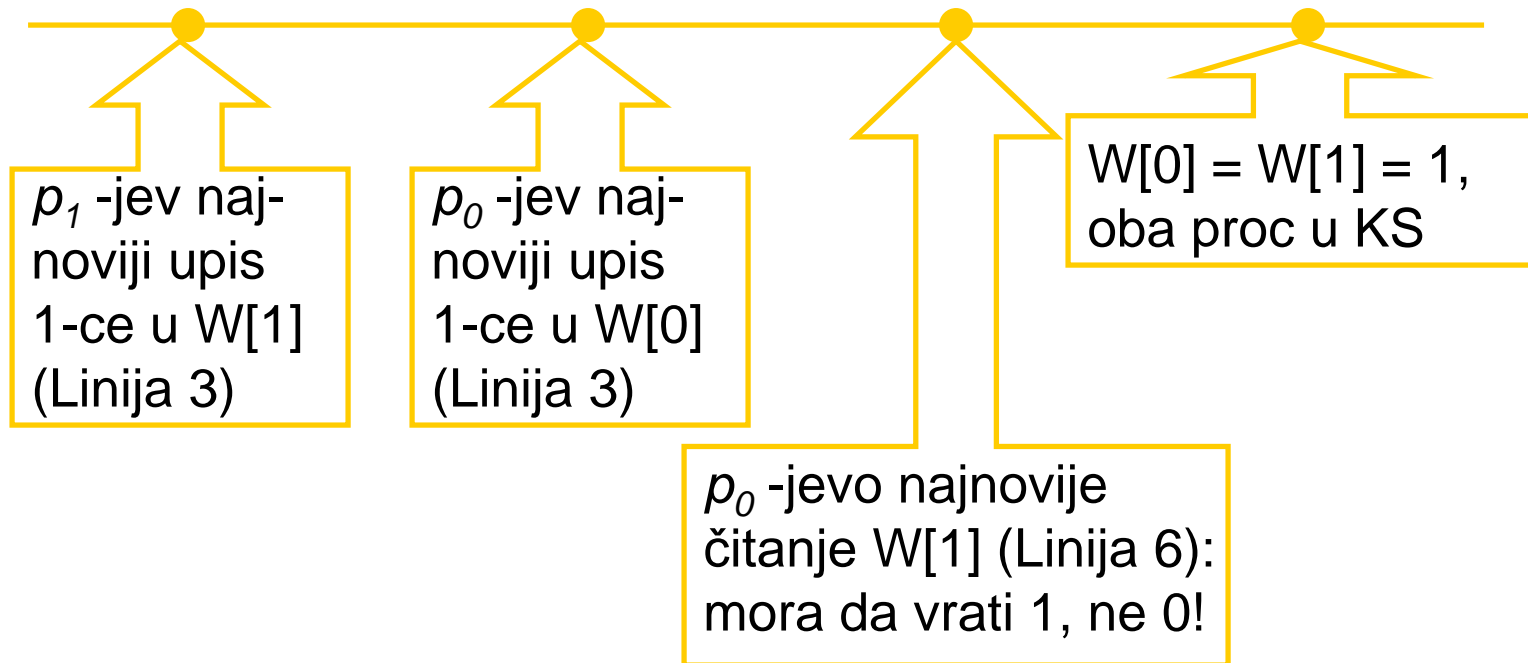
- **Međusobno isključivanje:** Pret. radi kontradikcije da su p_0 i p_1 istovremeno u KS.



Analiza ME algoritma: međusobno isključivanje

17

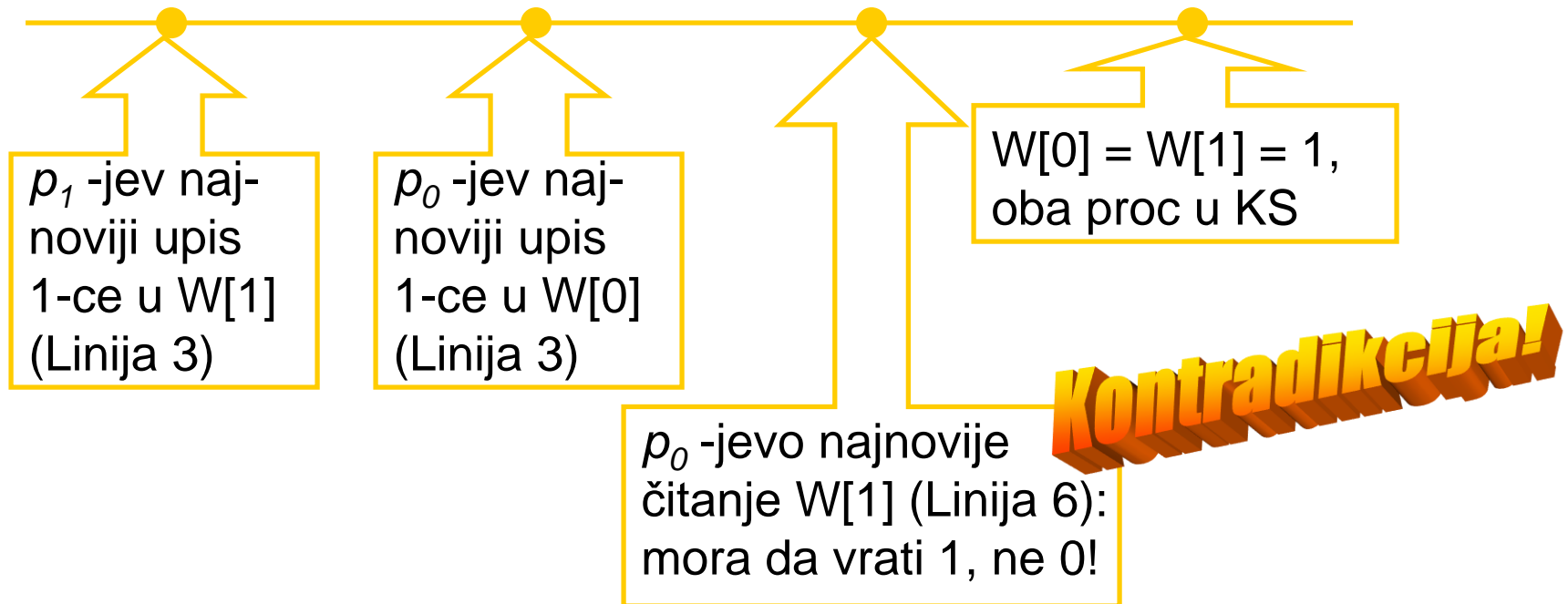
- **Međusobno isključivanje:** Pret. radi kontradikcije da su p_0 i p_1 istovremeno u KS.



Analiza ME algoritma: međusobno isključivanje

17

- **Međusobno isključivanje:** Pret. radi kontradikcije da su p_0 i p_1 istovremeno u KS.



Analiza ME algoritma: nema međusobnog blokiranja

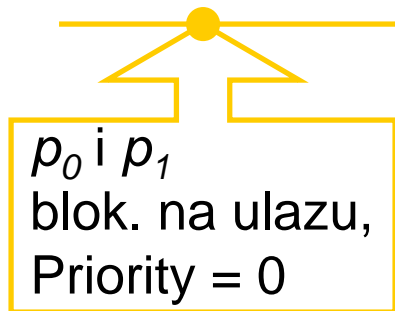
18

- Korisno za pokazivanje da nema trajnog zaključa.
- Ako jedan proc. ikad uđe u ostatak za uvek, drugi proc. ne može biti trajno zaključan (starved).
 - ▣ Npr: Ako p_1 uđe u ostatak za uvek, onda će p_0 uvek videti $W[1] = 0$.
- Znači, bilo koje međusobno blokiranje bi blokiralo oba proc.

Analiza ME algoritma: nema međusobnog blokiranja

19

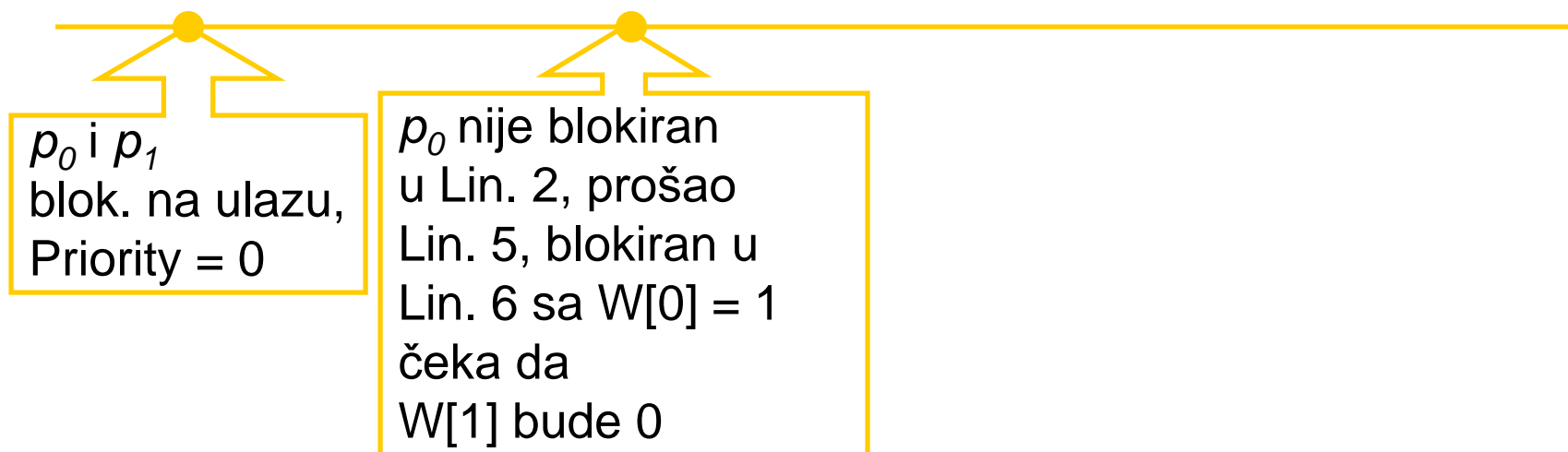
- Pred. radi kontradikcije da je došlo do među. blok.
- Pret. da `Priority` ostaje na 0 nakon što su se oba proc. blokirali u svojim ulaznim sekcijama.



Analiza ME algoritma: nema međusobnog blokiranja

19

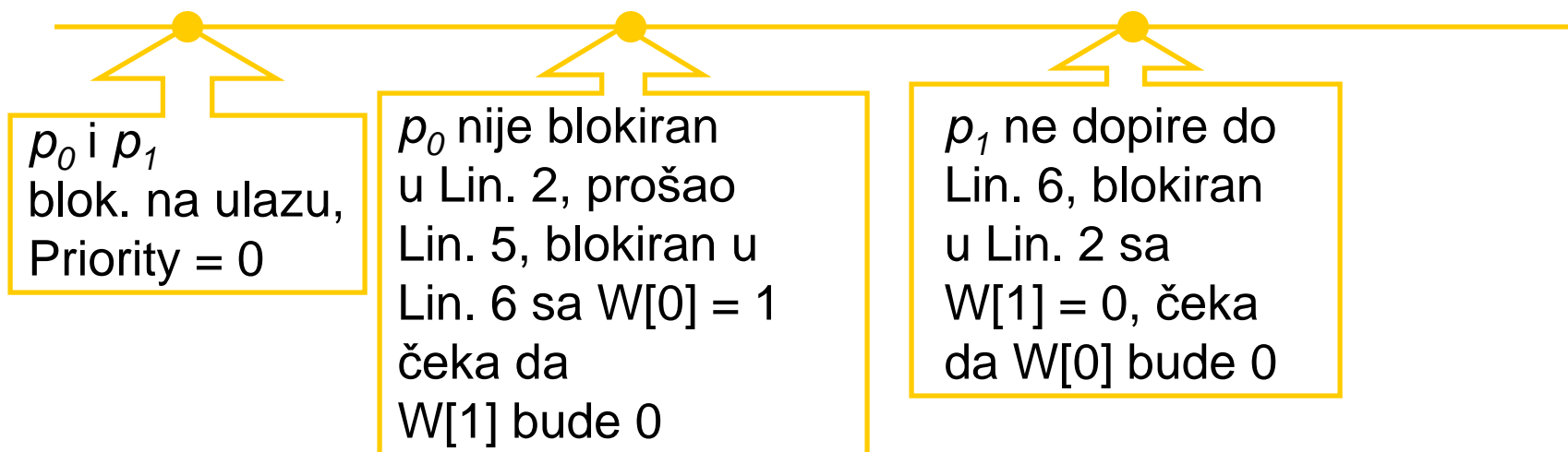
- Pred. radi kontradikcije da je došlo do među. blok.
- Pret. da `Priority` ostaje na 0 nakon što su se oba proc. blokirali u svojim ulaznim sekcijama.



Analiza ME algoritma: nema međusobnog blokiranja

19

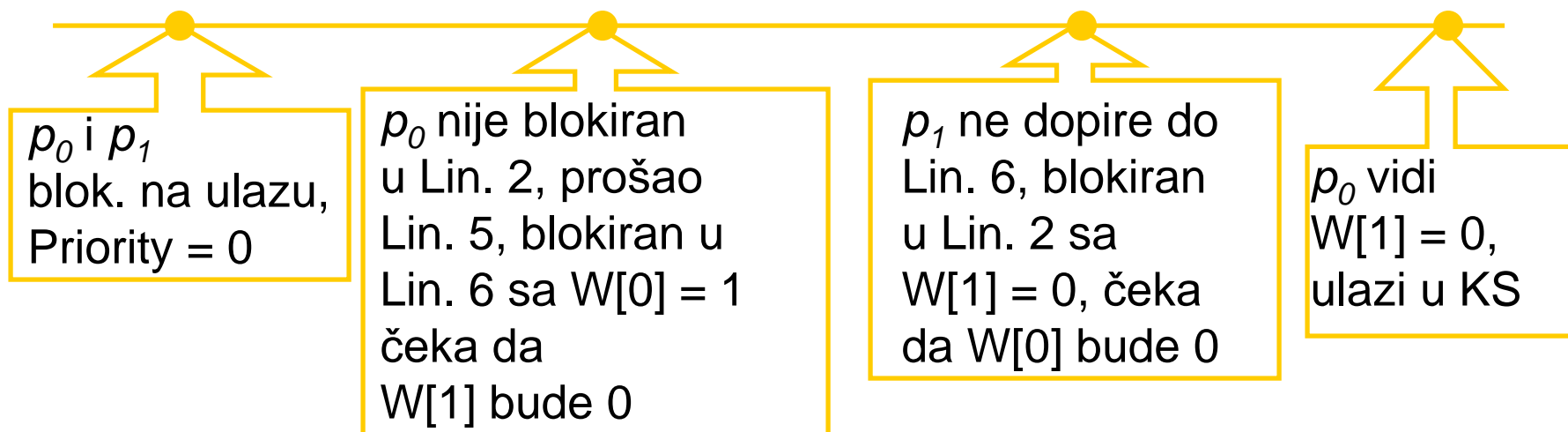
- Pred. radi kontradikcije da je došlo do među. blok.
- Pret. da `Priority` ostaje na 0 nakon što su se oba proc. blokirali u svojim ulaznim sekcijama.



Analiza ME algoritma: nema međusobnog blokiranja

19

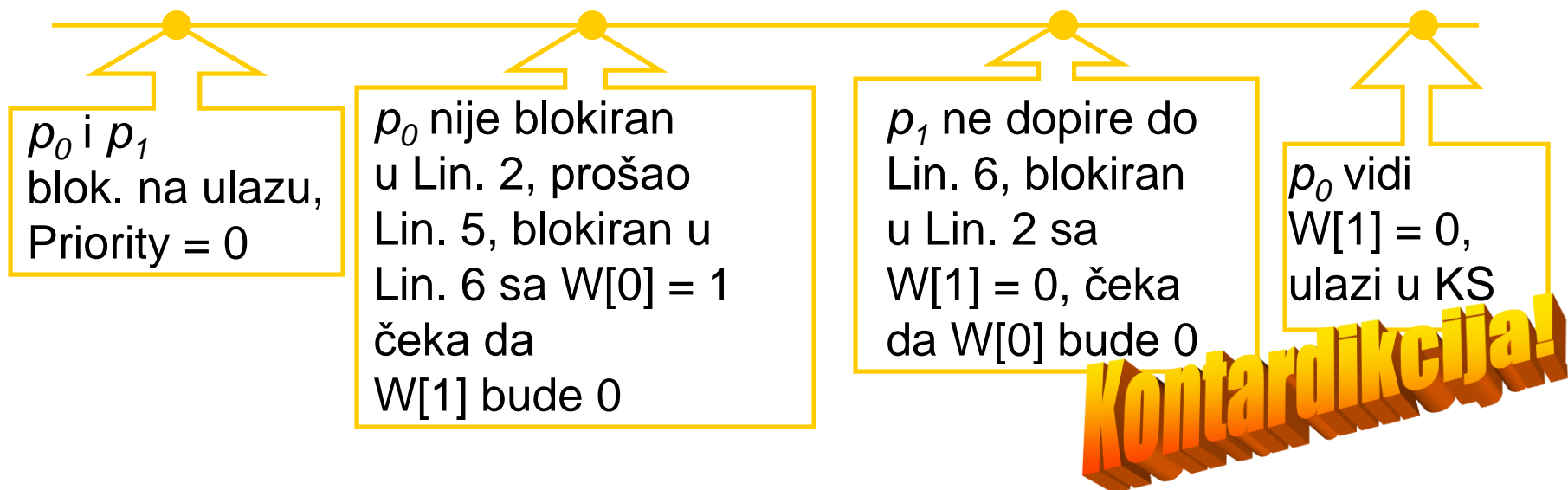
- Pred. radi kontradikcije da je došlo do među. blok.
- Pret. da `Priority` ostaje na 0 nakon što su se oba proc. blokirali u svojim ulaznim sekcijama.



Analiza ME algoritma: nema međusobnog blokiranja

19

- Pred. radi kontradikcije da je došlo do među. blok.
- Pret. da `Priority` ostaje na 0 nakon što su se oba proc. blokirali u svojim ulaznim sekcijama.



Analiza ME algoritma: nema trajnog zaključavanja

20

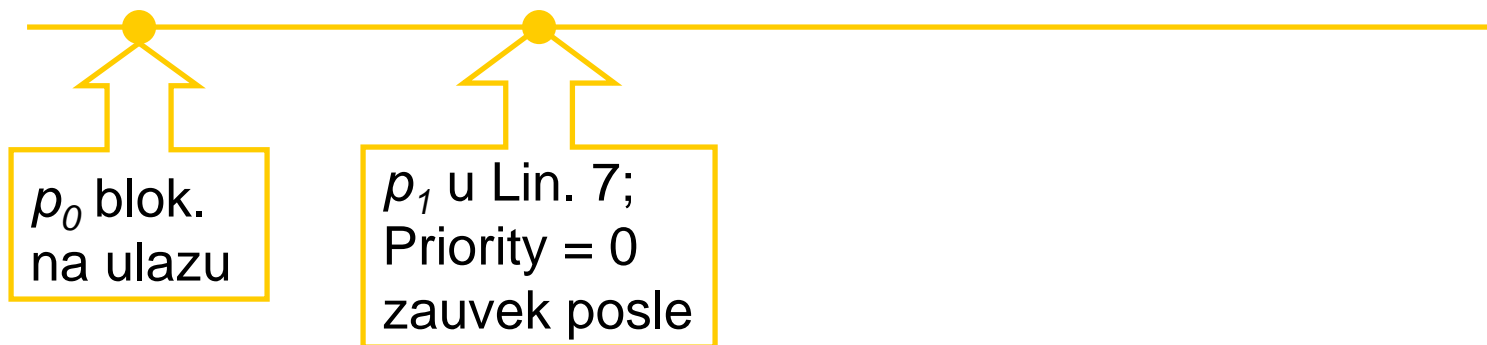
- Pred. radi kontradikcije da je p_0 trajno zaključan.
- Pošto nema među. blokiranja, p_1 ulazi u KS beskonačno često.
- Prvi put kad p_1 izvrši Lin. 7 u izlaznoj sekciji nakon što je p_0 blok. na ulazu, `Priority` se blokira na 0.



Analiza ME algoritma: nema trajnog zaključavanja

20

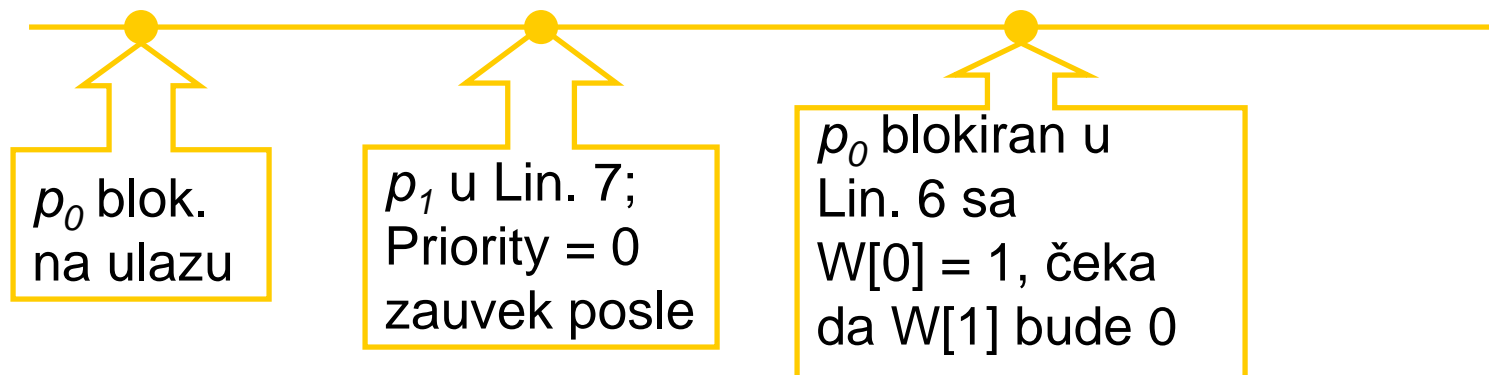
- Pred. radi kontradikcije da je p_0 trajno blokiran.
- Pošto nema među. blokiranja, p_1 ulazi u KS beskonačno često.
- Prvi put kad p_1 izvrši Lin. 7 u izlaznoj sekciji nakon što je p_0 blok. na ulazu, `Priority` se blokira na 0.



Analiza ME algoritma: nema trajnog zaključavanja

20

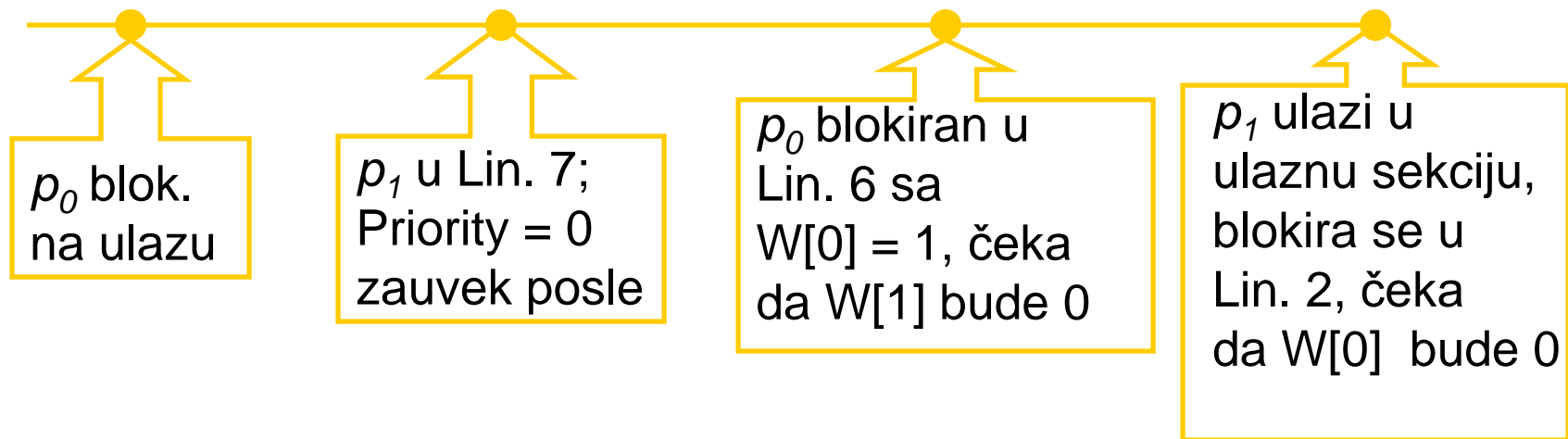
- Pred. radi kontradikcije da je p_0 trajno blokiran.
- Pošto nema među. blokiranja, p_1 ulazi u KS beskonačno često.
- Prvi put kad p_1 izvrši Lin. 7 u izlaznoj sekciji nakon što je p_0 blok. na ulazu, `Priority` se blokira na 0.



Analiza ME algoritma: nema trajnog zaključavanja

20

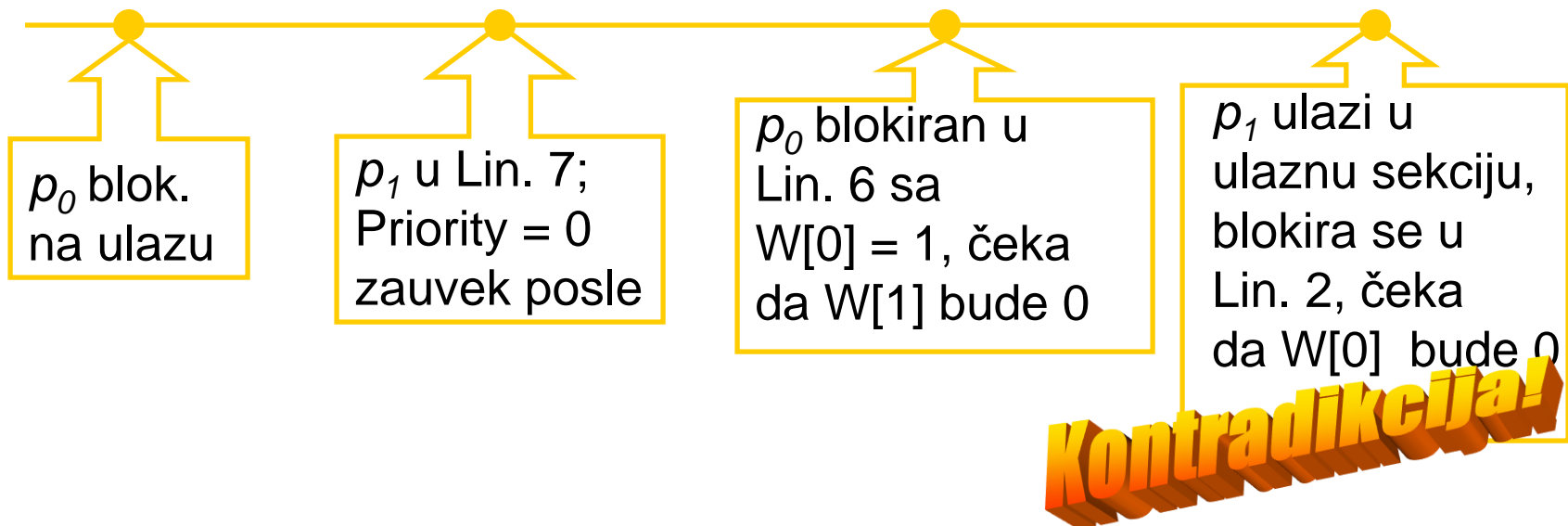
- Pred. radi kontradikcije da je p_0 trajno blokiran.
- Pošto nema među. blokiranja, p_1 ulazi u KS beskonačno često.
- Prvi put kad p_1 izvrši Lin. 7 u izlaznoj sekciji nakon što je p_0 blok. na ulazu, `Priority` se blokira na 0.



Analiza ME algoritma: nema trajnog zaključavanja

20

- Pred. radi kontradikcije da je p_0 trajno blokiran.
- Pošto nema među. blokiranja, p_1 ulazi u KS beskonačno često.
- Prvi put kad p_1 izvrši Lin. 7 u izlaznoj sekciji nakon što je p_0 blok. na ulazu, `Priority` se blokira na 0.



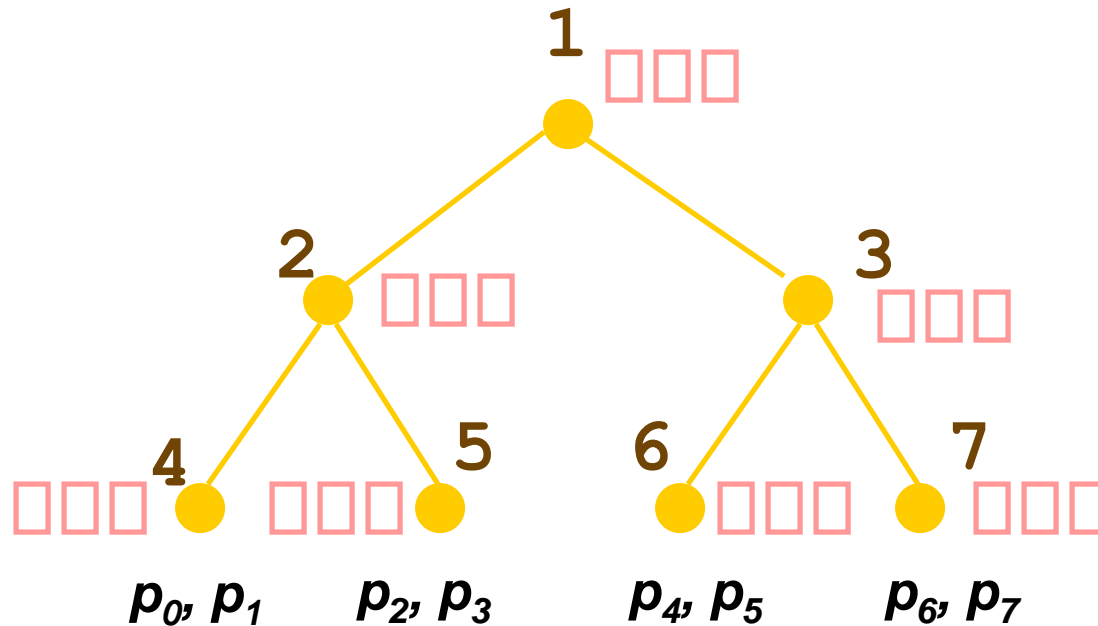
ME algoritam za n procesora

21

- Da li postoji mutex algoritam sa ograničenim prostorom za n veće od 2 procesora?
- Da!
- Na osnovu pojma stabla za turnir: zamislimo potpuno binarno stablo sa $n-1$ čvorova
 - ▣ stablo je samo ideja! ono ne predstavlja kanale za slanje poruka
- Svakom čvoru stabla se pridružuje kopija algoritma za 2 procesora
 - ▣ uključuje zasebne kopije 3 deljene promenljive

Stablo za turnir (Tournament Tree)

22



ME algoritam sa stablom za turnir

23

- Svaki proc. počinje ulaznu sekciju u određenom listu (dva proc. po listu)
- Proc. ide na sledeći nivo stabla posle pobeđe u utakmici 2-proc. za tekući čvor stabla:
 - ▣ na levoj strani, igra ulogu p_0
 - ▣ na desnoj strani, igra ulogu p_1
- Kada proc. pobedi u utakmici za koren stabla, on ulazi u KS.

Još o Alg. sa stablom za turnir

24

- Kod u knjizi je rekurzivan.
- p_i počinje u čvoru $2^k + \lfloor i/2 \rfloor$,
igra ulogu $p_{i \bmod 2}$, gde je $k = \lceil \log n \rceil - 1$.
- Posle pobeđe u čvoru v , „KS“ za čvor v je
 - ▣ ulazni kod za sve čvorove na putanji od predka čvora, a to je $\lfloor v/2 \rfloor$, do korena
 - ▣ prava kritična sekcija
 - ▣ izlazni kod za sve čvorove na putanji od korena do $\lfloor v/2 \rfloor$

Analiza Alg. sa stablom za turnir

25

- **Korektnost:** zasniva se na korekt. algoritma za 2 procesora i strukturi turnira:
 - Projekcija prihvatljivog izvršenja alg. sa turnirom na određen čvor stabla daje prihvatljivo izvršenje alg. za 2 proc.
 - ME uslov alg. sa turnirom sledi iz ME usova alg. za 2 proc. u korenu stabla.
 - NL uslov za alg. sa turnirom sledi iz NL uslova za algoritme za 2 proc. u svim čvorovima stabla
- **Prostorna složenost:** $3n$ Bulovih read/write deljenih promenljivih.