Lygiagrečiojo programavimo esmė

Nuoseklusis programavimas

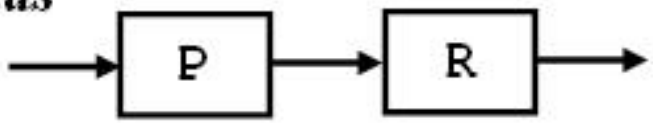
- išraiškų abstrakcija;
- duomenų abstrakcija;
- bet: užrašymo tvarka nustato vykdymo tvarką:
  - $\forall v : P \to R$ arba $R \to P$ (P, R – sakiniai)
  - $\forall v : x \to y$ arba $y \to x$ (x, y – komandos)
- tie patys duomenys – visada tie patys rezultatai;
- tvarkos nurodymo pertekliškumas.

Lygiagretusis (*concurrent*) programavimas

- tvarkos abstrakcija;
- užrašymo tvarka nenurodo vykdymo tvarkos:
  - $P \parallel R \leftrightarrow \exists v$: ne $P \rightarrow R$ ir ne $R \rightarrow P$ (P, R – sakiniai)
  - $P \parallel R \leftrightarrow \exists v$: ne $x \rightarrow y$ ir ne $y \rightarrow x$ (x, y – komandos)
- tie patys duomenys – galimi skirtingi rezultatai;
- vartotoją tenkina tik tam tikri rezultatai.

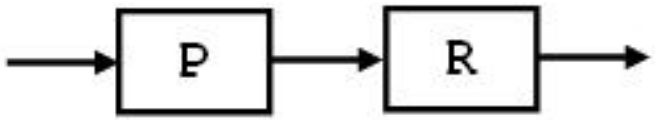
Nuoseklus ir lygiagretus vykdymas

nuoseklus vykdymas

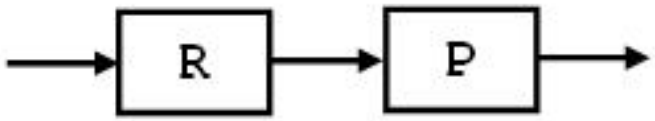


lygiagretus vykdymas

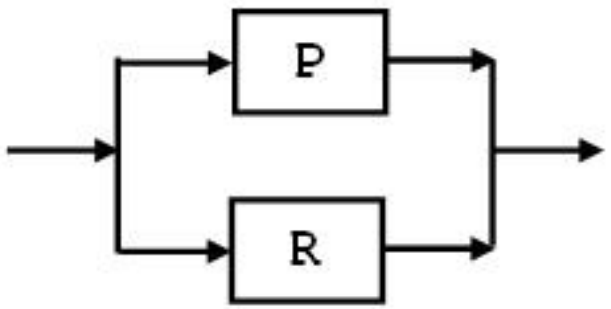
a)



b)



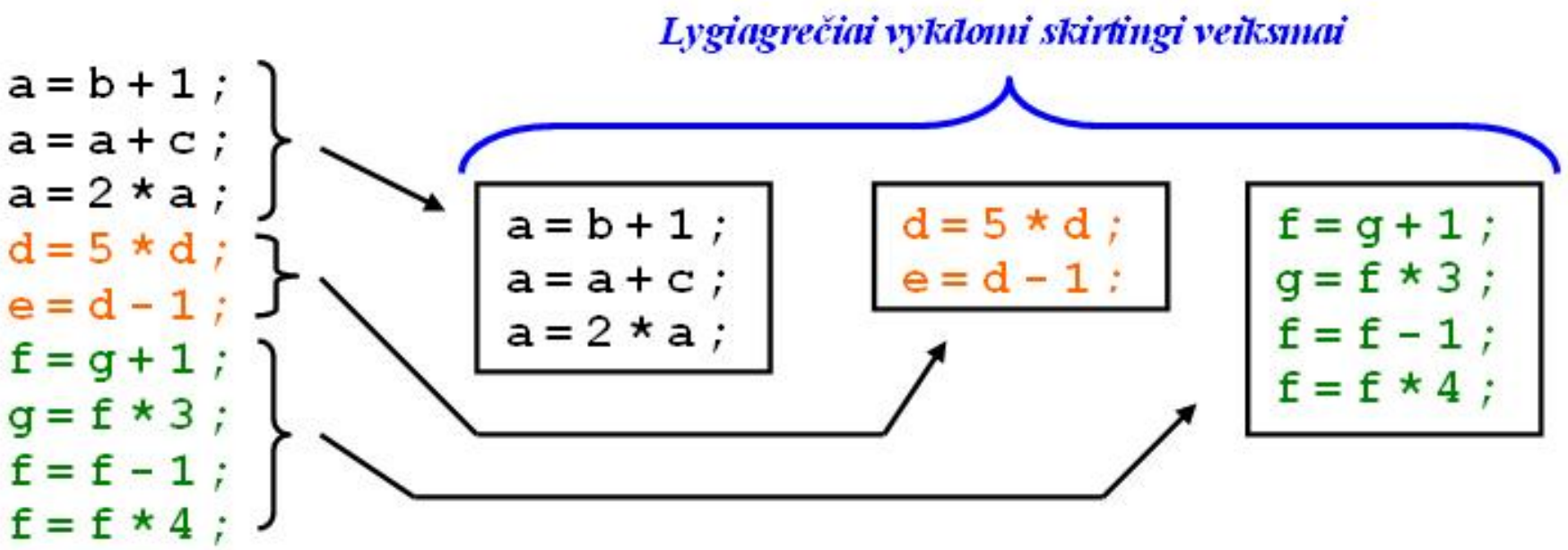
c)



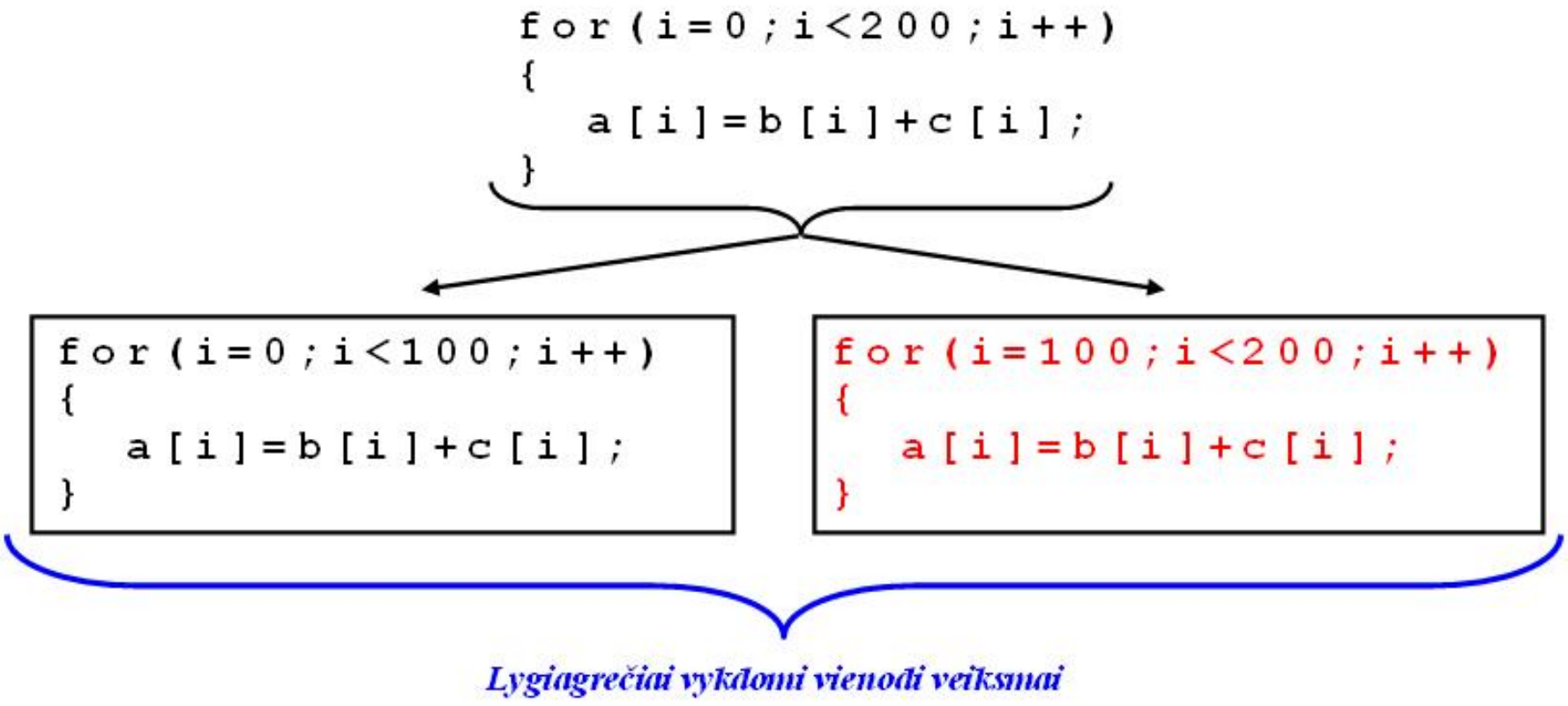
Lygiagretumo variantai

- funkcinis lygiagretumas;
- duomenų lygiagretumas.

Funkcinis lygiagretumas



Duomenų lygiagretumas



LP nauda

- programinis modelis geriau atitinka realųjį pasaulį;
- daugiaprocesorinėje sistemoje galima padidinti vykdymo greitį;
- vienprocesorinėje sistemoje viena programa praktiškai neišnaudoja viso procesoriaus laiko;
- iš anksto nustatyta veiksmų tvarka kai kuriems taikymams yra nepriimtina.

LP taikymo sritys

Gamtos ir inžinerijos mokslų sudėtingų problemų sprendimas:

- atmosferos reiškinių prognozavimas;
- branduolinė fizika, matematika;
- kosminių aparatų ir ginklų projektavimas;
- ir t. t.

Pramonė ir komercija:

- naftos ir dujų gavyba;
- farmacija;
- finansai ir ekonominis modeliavimas;
- ir t. t.

LP problemos

- skirtingi rezultatai gali būti gauti vykdant programą net ir su tais pačiais duomenimis;
- vartotoją tenkina tik tam tikri rezultatai, o tuo pačiu ir tik tam tikros veiksmų atlikimo sekos;
- reikia uždrausti nenaudingas vykdymo sekas, t. y. sinchronizuoti procesus;
- procesai turi keistis informacija.

Procesoriai ir procesai

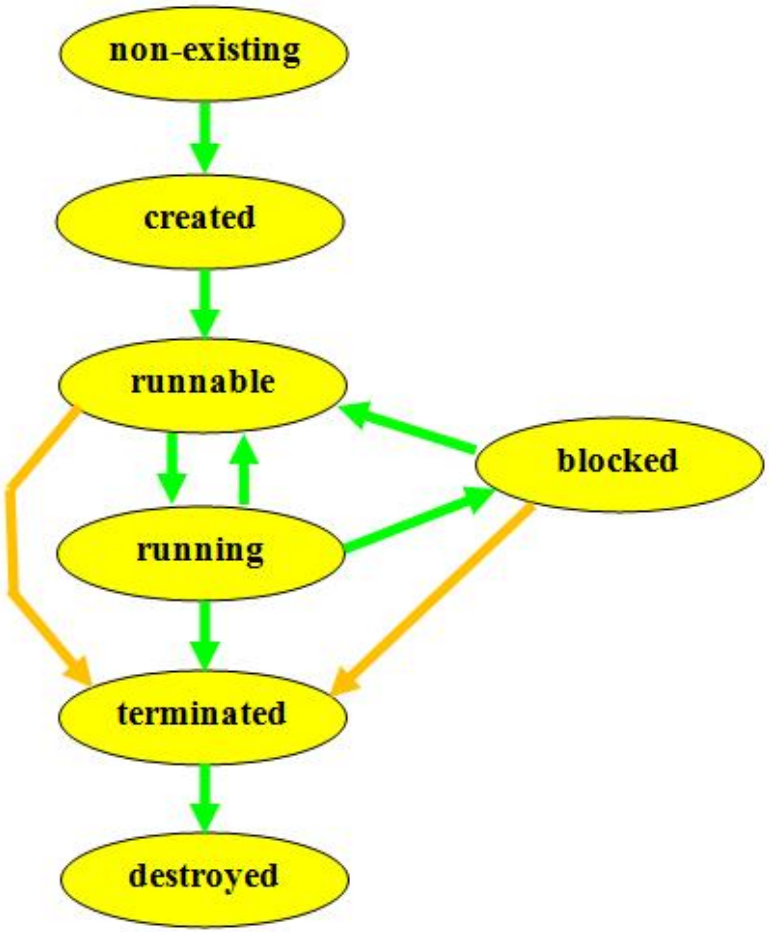
- procesorius – tai techninis įtaisas, vykdantis programos kodą;
- procesas – tai programos kodo fragmentas, nuosekliai vykdomas viename procesoriuje;
- LP: procesų sk. $\geq$ fizinių procesorių sk.,
  procesų sk. = loginių procesorių sk.

Procesų įvairovė

- statiniai ir dinaminiai procesai
- vieno ir daugelio lygių procesai
- procesų inicializavimo ir užbaigimo skirtumai

Procesų būsenos

- neegzistuojąs (*non-existing*)
- sukurtas (*created*)
- pasiruošęs (*ready, runnable*)
- vykstąs (*running*)
- blokuotas (*blocked*)
- nutrauktas (*terminated, dead*)
- sunaikintas (*destroyed*)



Lygiagretumo operacijos savybės

- komutatyvumas: $P \parallel R \Rightarrow R \parallel P$
- asociatyvumas: $P \parallel (R \parallel Q) \Rightarrow (P \parallel R) \parallel Q$
- tranzityvumas: $P \parallel R$ ir $R \parallel Q \Rightarrow P \parallel Q \Rightarrow P \parallel R \parallel Q$

Procesai ir gijos

- OS:
  - 1 programa $\Rightarrow$ 1 procesas
  - 1 procesas $\Rightarrow$ daug ($\geq$ 1) gijų
  - procesai neturi bendros atminties
  - gijų atmintis bendra
- LP:
  - 1 programa $\Rightarrow$ daug ($\geq$ 1) procesų
  - 1 procesas $\equiv$ 1 gija

Procesai, gijos ir atminties naudojimas

- procesai (Java, OpenMP gijos, MPI procesai) gali turėti savo lokalius kintamuosius;
- Java, OpenMP gijos gali turėti bendrus kintamuosius;
- MPI procesai negali turėti bendrų kintamųjų;
- nuo to, ar procesai turi bendrą atmintį, priklauso procesų komunikavimo ir sinchronizavimo priemonės.

Bendros ir atskiros atminties modeliai

- Bendra procesų atmintis:
  - procesai komunikuoja per bendrus kintamuosius;
  - procesai gali būti sinchronizuojami naudojant bendrus kintamuosius;
  - problema: bendrų kintamųjų apsauga.
- Atskira procesų atmintis:
  - procesai komunikuoja siųsdami ir priimdami pranešimus;
  - procesai sinchronizuojami siunčiant sinchronizavimo signalus;
  - problema: priimti pranešimus iš kelių procesų.
- Atminties modelis ir kompiuterio architektūra:
  - programos architektūros modelis (bendra ar atskira atmintis) turi būti nepriklausomas nuo kompiuterio architektūros;
  - programa vykdoma efektyviai, jei programos modelis sutampa su kompiuterio architektūros modeliu.

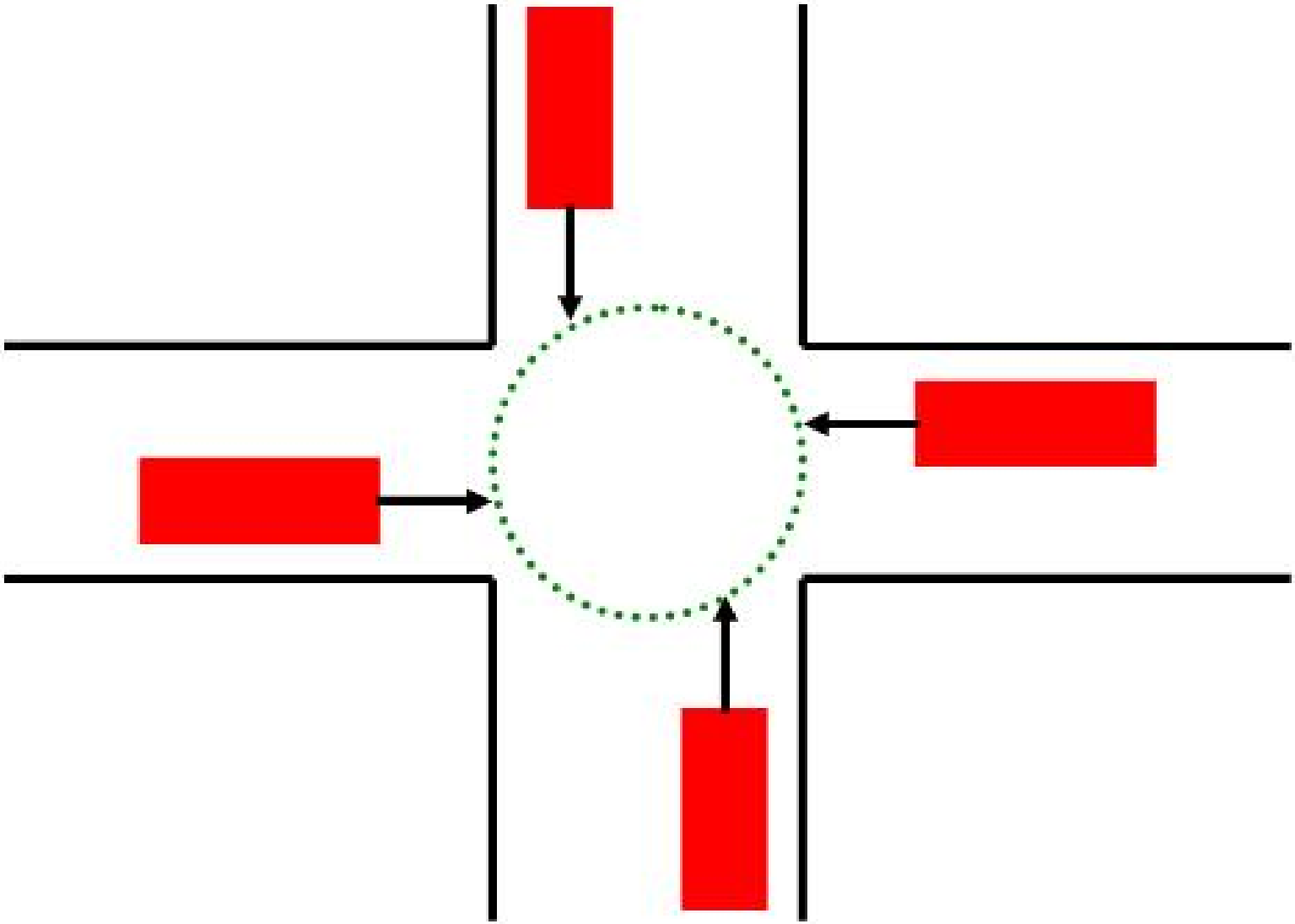
Pagrindinės sąvokos (1)

- atominis veiksmas (*atomic action*) – veiksmas, kuriam vykstant nevyksta kitų procesų veiksmai;
- kritinė sekcija (*critical section*) - programos kodo gabalas, kuris kitų procesų atžvilgiu turi būti atominiu veiksmu;
- tarpusavio išskyrimas (*mutual exclusion*) - priemonės, neleidžiančios dviems procesams vienu metu vykdyti savo veiksmus (naudotis tuo pačiu resursu);
- dviejų procesų KS turi būti vykdomos su tarpusavio išskyrimu (*mutual exclusion*);
- varžymosi būsena (*race condition*) – tai būsena, kai procesai vienu metu varžosi dėl kokių tai resursų;

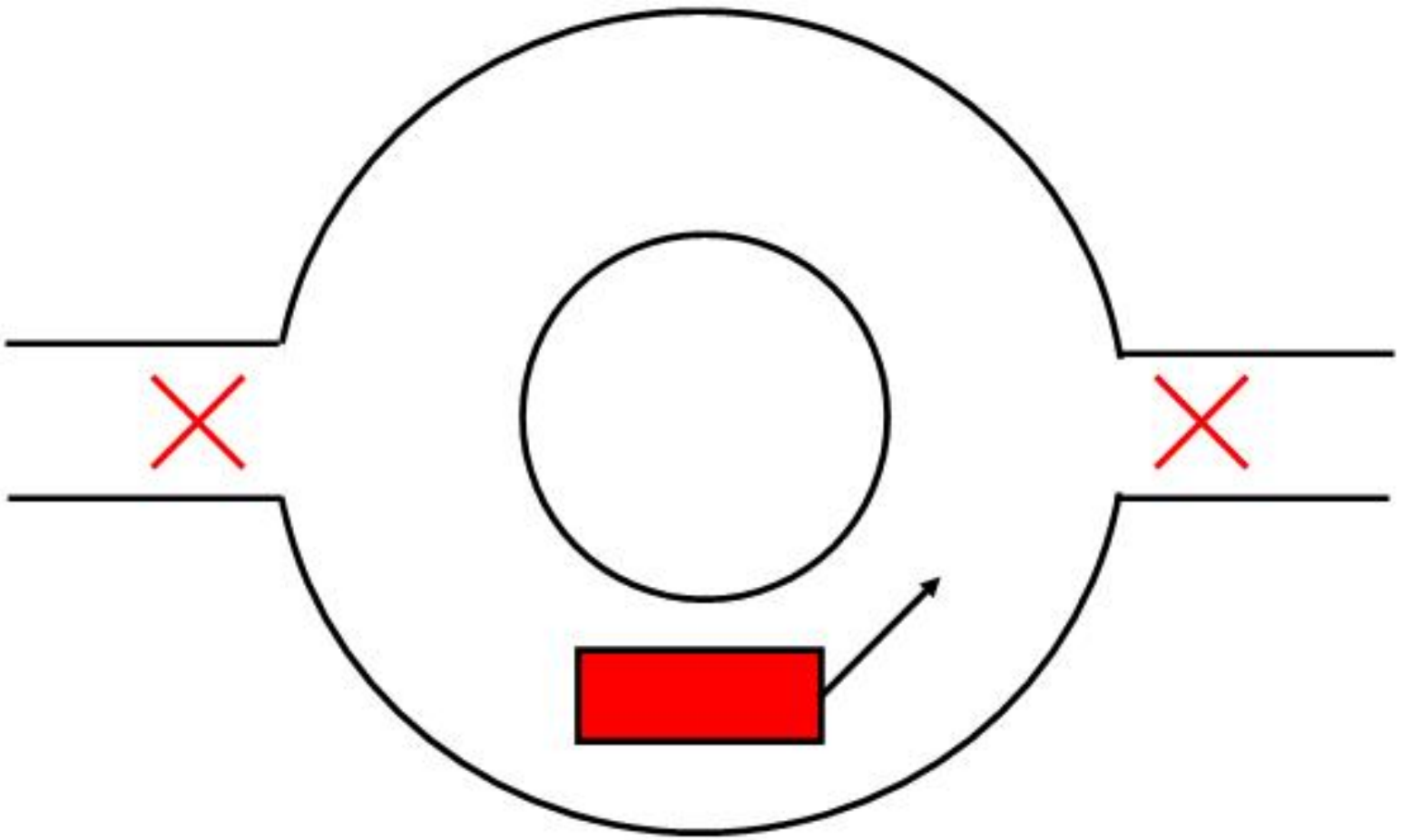
Pagrindinės sąvokos (2)

- užimtas laukimas (*busy wait*) – būsena, kai procesas laukia kokio tai resurso pastoviai tikrindamas jo užimtumą;
- badavimas (*starvation*) – tai būsena, kai procesui pastoviai neleidžiama naudotis bendrais resursais.
- aklavietė (*deadlock*) – tai būsena, kai procesas laukia to, kas niekada neįvyks;
- begalinis ciklas (*livelock*) – tai būsena, kai procesas cikliškai vykdo tuos pačius veiksmus be jokio progreso;

Aklavietės pavyzdys



Begalinio ciklo pavyzdys



LP problemas iliustruojantys uždaviniai

- Dekoratyvinis sodas (*Ornamental Gardens*)
- Gamintojas – Vartotojas (*Producer – Consumer*)
- Skaitantieji – Rašantieji (*Readers – Writers*)
- Pietaujantys filosofai (*Dining Philosophers*)

Bendra atmintis. Nenuspėjamas rezultatas

```
procesas P:                          procesas R:
  ...                                  ...
  int i, k;                            int i, k;
  for(i=0;i<10;i++){                   for(i=0;i<10;i++){
    k=c; k=k+1; c=k;                     k=c; k=k+1; c=k;
  }                                    }
  ...                                  ...
  // ------------------------------------
     int c=0;
     P; R;                  // Vykdomi lygiagrečiai
     Spausdinti (c); // Kokia reikšmė?
```

Nėra bendros atminties. Nenuspėjamas rezultatas

```
procesas Q:                          procesas R:
  int k = 5;                           int k = 10;
  Siųsti k procesui P;                 Siųsti k procesui P;

procesas P:
  int a, k = 0;
  Priimti k iš Q arba R;
  a = k;
  Spausdinti (a);              // Kokia reikšmė?
// ------------------------------------
    P; Q; R;                   // Vykdomi lygiagrečiai
```

Klausimai pakartojimui

1. Kuo ypatinga tvarkos abstrakcija?
2. Kuo pasižymi funkcinis lygiagretumas?
3. Kuo pasižymi duomenų lygiagretumas?
4. Kokia esminė lygiagrečiojo programavimo nauda?
5. Kokios yra pagrindinės lygiagrečiojo programavimo problemos?
6. Paaiškinkite sąvokas "fizinis procesorius", "loginis procesorius".
7. Kokie yra procesų komunikavimo ir sinchronizavimo variantai?