Strukturno PP, I deo:

- Uvod i podela
- Šabloni kompozicije i kontrole toka

Klasično strukturno programiranje

- Klasično strukturno programiranje (KSP):
  - E.W. Dijkstra, A.P. Hoare i N. Wirth
  - Logičko organizovanje programa, korektnost
  - Posebno popularan bio jezik Pascal
- Principi KSP:
  - Programiranje od opšteg ka posebnom, tj. od-gore-na-dole (eng. top-down)
  - Modularnost
  - Ograničen broj upravljačkih struktura (do, while...)

Strukturno paralelno programiranje

- Autori:
  - Michael McCool, Arch D. Robison i James Reinders
- Cilj: Reafirmacija principa KSP u okviru OOP
- Rezultat:
  - Strukturno paralelno programiranje (SPP)
  - Konačan br. upravljačkih struktura
  - SPP je zasnovano na šablonima (eng. structured pattern-based parallel programming)
  - ŠABLONI ALGORITAMSKE STRATEGIJE (ŠAS) na srednjem nivou apstrakcije

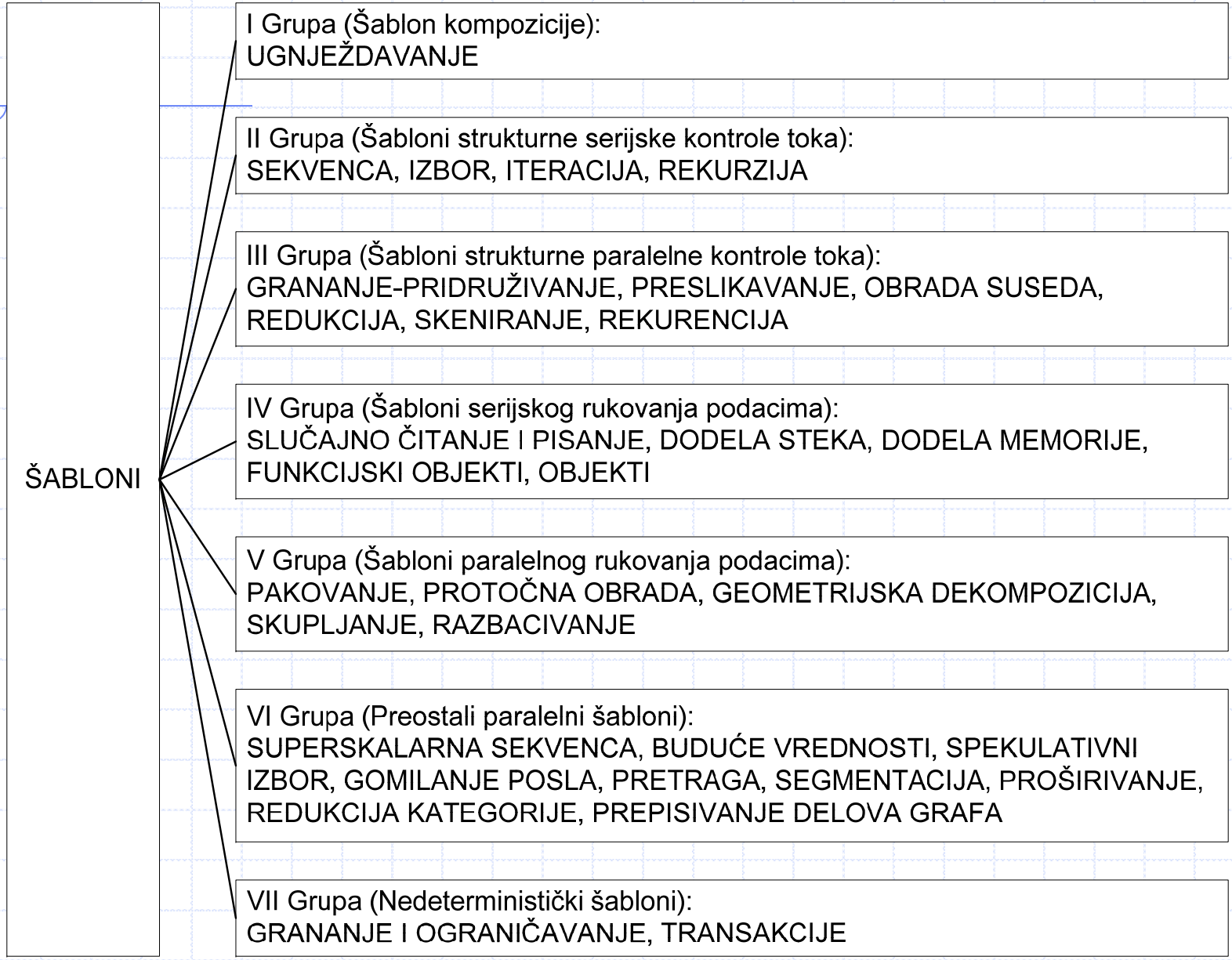
ŠAS (1/2)

- ŠAS su na nivou između:
  - PROJEKTANTSKIH ŠABLONA (eng. design patterns) i
  - IMPLEMENTACIONIH ŠABLONA (eng. impl. patterns)
- ŠAS = ALGORITAMSKI SKELET
- Svaki ŠAS (ili kratko šablon) ima dva dela:
  - Semantika
  - Implementacija
- Semantika opisuje kako se šablon koristi kao gradivni blok algoritma
  - aranžmana zadataka i zavisnosti podataka

ŠAS (2/2)

- Semantika: apstrakcija koja skriva neke detalje
  - Npr. režim izvršenja zadataka
  - Zaista paralelno izvršenje ili ne
- Implementacija mora biti efikasna
- Važni implementacioni problem su:
  - KONTROLA GRANULARNOSTI (eng. granularity)
  - Dobra upotreba SKRIVENE MEMORIJE (eng. cache)
- Različite implementacije šablona mogu dovesti do različite performanse
  - ali ne i do različite semantike

Klasifikacija šablona



Najvažniji šabloni (1/2)

- 3 šablona zaslužuju posebnu pažnju:
  - Ugnježdavanje
  - Preslikavanje
  - Grananje-Pridruživanje
- Ugnježdavanje – ključna ideja:
  - Specificirati NEOBAVEZNI PARALELIZAM umesto OBAVEZNOG PARALELIZMA
- Preslikavanje - regularna paralelizacija:
  - Deli problem na uniformne podprobleme
  - Omogućava efikasnu paralelizaciju i vektorizaciju

Najvažniji šabloni (2/2)

- Grananje-Pridruživanje:
  - Rekurzivno deli problem na podprobleme
  - Kako za regularnu tako i za iregularnu paralelizaciju
  - Strategija podeli-i-zavladaj
- PARALELIZAM PODATAKA:
  - Paralelizam operacija nad podacima - skalabilnosti
  - Broj podproblema raste sa ukupnom veličinom problema

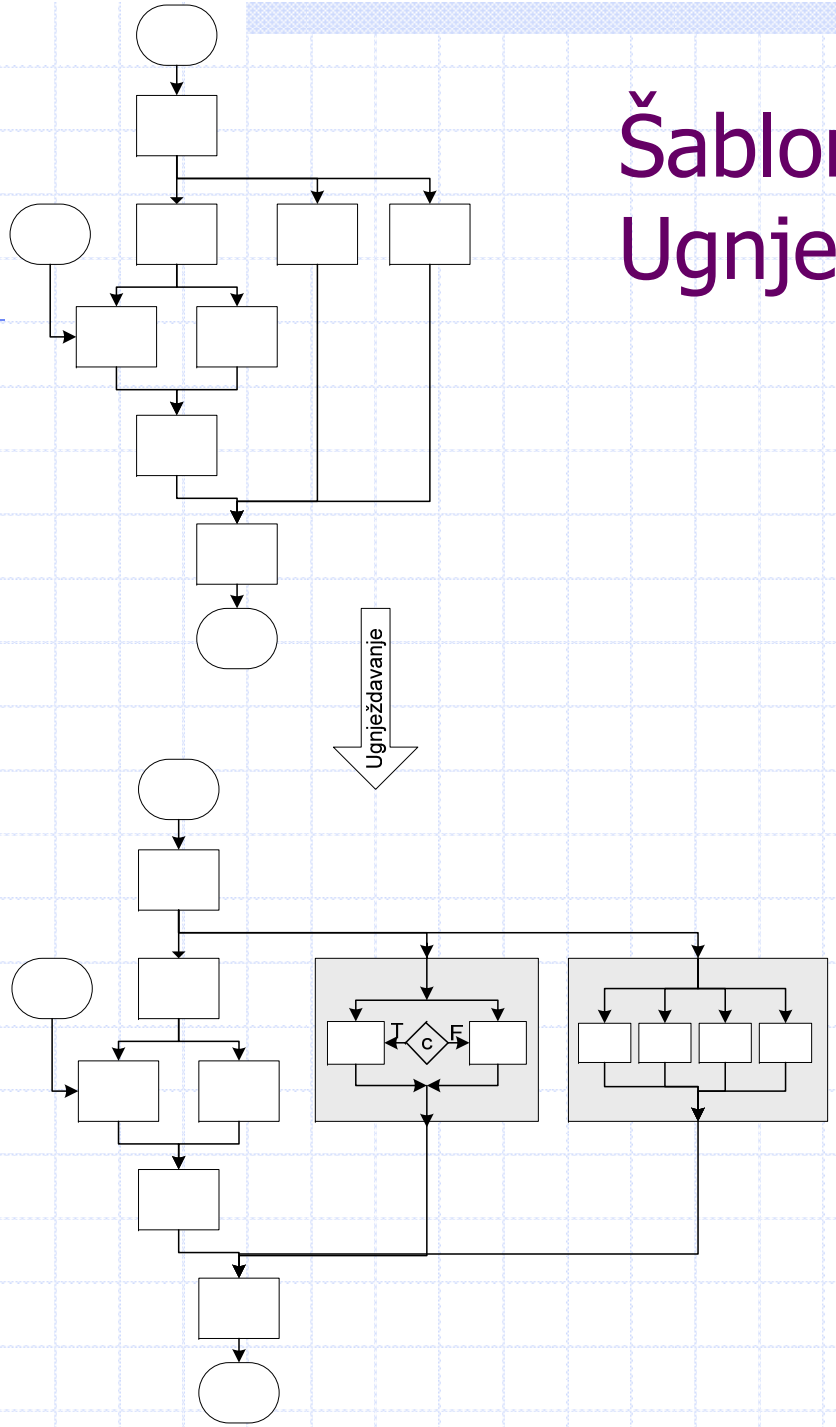
Dijagrami za prikazivanje šablona

- Konvencije dijagrama:
  - Zadaci: pravougaoni simboli
  - Podaci: ovalni simboli
  - Grupisani zadaci: pravougla okruženja
  - Grupisani podaci: zaobljena okruženja
  - Dodatni simboli u obliku raznih poligonalnih oblika
  - Uređenje zavisnosti je dato strelicama
  - Vreme teče od gore prema dole
  - Izbegavaju se strelice koje pokazuju naviše
    - Osim u slučaju iteracije
    - Kada ih nema, visina šablona odgovara rasponu šablona

Šablon Ugnježđavanje (1/4)

- Hijerarhijsko i rekurzivno komponovanje šablona
  - Blokovi zadataka: mesta za kod ili drugi šablon
  - Dubina ugnježđavanja je neograničena
  - SADRŽAVAJUĆI šablon ne sme da uvede ograničenja u pogledu vrste SADRŽANIH šablona
- Svi šabloni podržavaju ugnježđavanje
  - Inače ne bi bilo moguće praviti biblioteke
- U primeru su unutar dva bloka zadataka u šablonu Superskalarna sekvenca ugnježđeni šabloni Spekulativni izbor i Preslikavanje

Šablon Ugnježđavanje (2/4)



Šablon Ugnježđavanje (3/4)

- Ugnježđavanje može biti:
  - STATIČKO (u strukturi programskog koda)
  - DINAMIČKO (to je rekurzija)
- Prednost dinamičkog:
  - Dinamički paralelizam podataka – dobra skalabilnost
- U SPP treba:
  - Koristiti šablone koji se uklapaju u tekući šablon
  - Izbegavati dodatne zavisnosti bilo u toku upravljanja ili toku podataka

Šablon Ugnježđavanje (4/4)

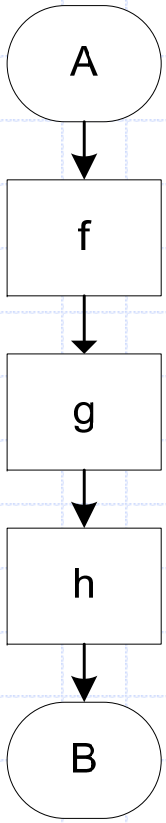
- Problem: fizički resursi su konačni
  - Pojava prevelike pretplaćenosti (oversubscription)
  - Treba inteligentno preslikavati potencijalan na fizički paralelizam
- Neki modeli programiranja uvode ograničenja
  - Ograničen broj ugnježđavanja
  - Direktno preslikavanje programske hijerarhije na fizičku hijerarhiju
  - Npr. OpenCL, CUDA, C++, i donekle OpenMP
- Cilk i TBB: proizvoljna ugnježđavanja
  - Dobro preslikavaju potencijalan paralelizam

Šabloni serijske kontrole toka

- Šabloni strukturne serijske kontrole toka:
  - Sekvenca, Izbor, Iteracija i Rekurzija
  - Treba razumeti pretpostavke zbog paralelizacije
- Sekvenca je uređena lista zadataka
  - koji se izvršavaju po zadatom redosledu
- Osnovna pretpostavka
  - Serijsko izvršenje, čak i kad nema zavisnosti
  - Očuvava redosled ivičnih efekata zadataka
  - Npr. Ako f-ije u sl. primeru ispisuju “f”, “g” i “h”, onda Sekvenca uvek ispisuje “fgh”

Primer serijske Sekvence br. 1:

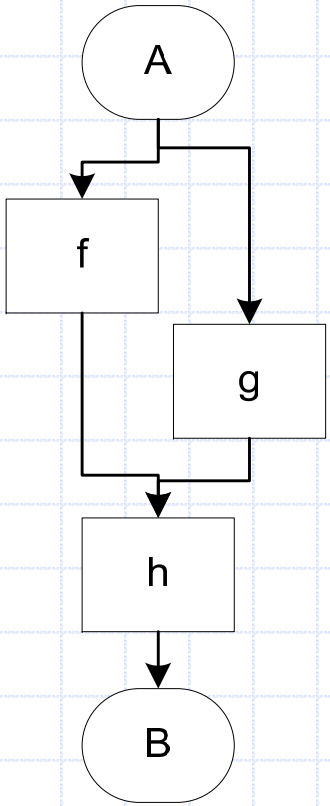
```
1 T = f(A);
2 S = g(T);
3 B = h(S);
```



- Dobija se tekst “fgh” čak i kada nema eksplictnih zavisnosti između ovih zadataka preko podataka

Primer serijske Sekvence br. 2:

```
1 T = f(A);
2 S = g(A);
3 B = h(S,T);
```



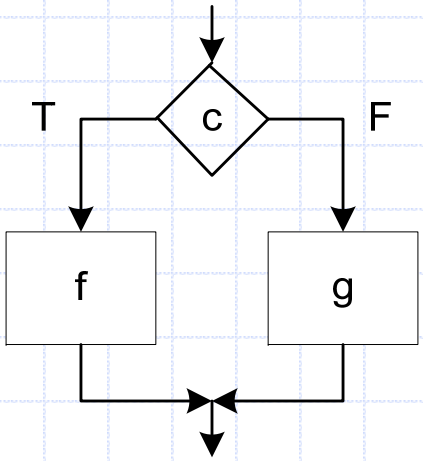
- Funkcija g bi se takođe izvršavala nakon funkcije f, iako za to nema konkretnih razloga

Sekvenca : Superskalarna sekvenca

- Šablon Superskalarna sekvenca je paralelna generalizacija šablona Sekvence
  - Uklanja ovo ograničenje tekstualnog redosleda
  - Zadaci se raspoređuju samo u skladu sa zavisnostima
  - Kod procesora sa izvršenjem instrukcija van redosleda čak se menjaja redosled operacija

Šablon Izbor

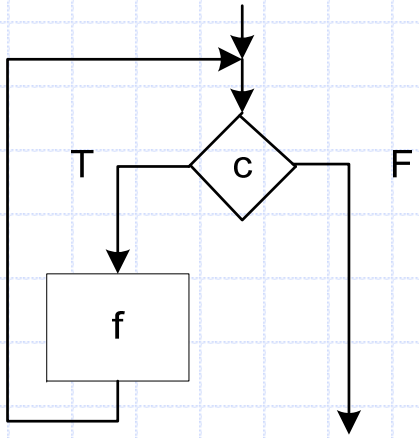
```
1 if (c) {
2     a;
3 } else {
4     b;
5 }
```



- Najpre se izračuna uslov c
- Ako je uslov istinit, izvrši se zadatak a, a ako nije izvrši se zadatak b
- Dve pretpostavke:
  - a i b se ne mogu izvršavati pre nego se izračuna uslov c
  - Izvršava se samo jedan od ova dva zadatka

Šablon Iteracija (1/4)

```
1 while (c) {
2     f;
3 }
```



- WHILE petlja
- Broj iteracija zavisi od podataka
- Raspon ne odgovara visini dijagrama
  - već se petlja mora mentalno izvršiti
  - Raspon = visina traga tog izvršenja

Šablon Iteracija (2/4)

- Problem: zadatak tela petlje f može zavisiti od svoji prethodnih poziva
  - To su PETLJOM-NOŠENE ZAVISNOSTI
  - Petlje se tada mogu paralelizovati na razne načine
- FOR petlja je petlja sa brojačem, tzv. INDEKSOM petlje
  - Koristi ugnježdenu WHILE petlju
  - Petljom-nošena zavisnost od indeksa **i**
  - Ipak, postoje razni načini da se FOR paralelizuje
  - Jer su unapred poznate vrednosti **i** za svaku iteraciju

Šablon Iteracija (3/4)

```
1 for (i = 0; i < n; ++i) {
2     f;
3 }
```

```
1 i = 0;
2 while (i < n) {
3     a;
4     ++i;
5 }
```

- Cilk Plus i OpenMP zabranjuju menjanje indeksa petlje **i** i promenljive **n** unutar tela petlje
  - Jer inače ukupan broj iteracija ne bi bio poznat unapred
- Nekoliko paralelnih šablona se mogu smatrati generalizacijama nekih specifičnih formi petlji,
  - Preslikavanje, Redukcija, Skeniranje, Rekurencija, Razbacivanje, Skupljanje i Pakovanje

Šablon Iteracija (4/4)

- Postoje forme zavisnosti koje sprečavaju paralelizaciju petlji
- Jedan od najvećih izazova paralelizacije:
  - Serijske iteracije se preslikavaju na mnogo različitih vrsta paralelnih strategija
- Složene i skrivene zavisnosti podataka:
  - kao rezultat kombinacije iteracije sa slučajnim pristupom memoriji i pokazivača
  - Primeri na sledećim slajdovima

Primer 1: Da li se ova funkcija može paralelizovati?

```
1  void engine(
2  int n, double x[], int a[], int b[], int c[], int d[],
3  ) {
4      for (int i = 0; i < n; ++i)
5          x[a[i]] = x[b[i]] * x[c[i]] + x[d[i]];
6  }
```

- Odgovor: Možda.
- Zavisnosti su određene elementima nizova a, b, c i d
- Strategija paralelizacije mora da uvaži konkretne vrednosti ovih elemenata

Primer 2: Da li se ova funkcija može paralelizovati?

```
1  void engine2(
2  int n, double x[], double y[], int a[], int b[], int c[], int d[],
3  ) {
4     for (int i = 0; i < n; ++i)
5          y[a[i]] = x[b[i]] * x[c[i]] + x[d[i]];
6  }
```

- Odgovor: Ponovo, možda.
- Nozovi x i y u programskom jeziku C su pokazivači
- Može se paralelizovati ako x i y ne pokazuju na isti niz lokacija (ili na preklopljene nizove lokacija)
- Čak i kad x i y nisu ALIJASI, paralelizacija nije moguća ukoliko u nizu a ima duplikata - trka do podataka

Šablon Rekurzija

- Dinamička forma ugnježdavanja u kojoj funkcija poziva samu sebe, direktno ili indirektno
  - Dodela memorije na steku ili
  - Dodela zatvorenih objekata ako su podržane funkcije višeg reda
- REKURZIJA-NA-REPU je poseban oblik rekurzije
  - Može se pretvoriti u iteraciju
  - Pozivajuća funkcija se vraća odmah nakon rekurzivnog poziva i
  - vraća vrednost, koju vraća rekurzivni poziv

Šabloni strukturne paralelne kontrole toka

- U ovu grupu šablona spadaju:
  - Grananje-Pridruživanje
  - Preslikavanje
  - Obrada suseda
  - Redukcija
  - Skeniranje i
  - Rekurencija

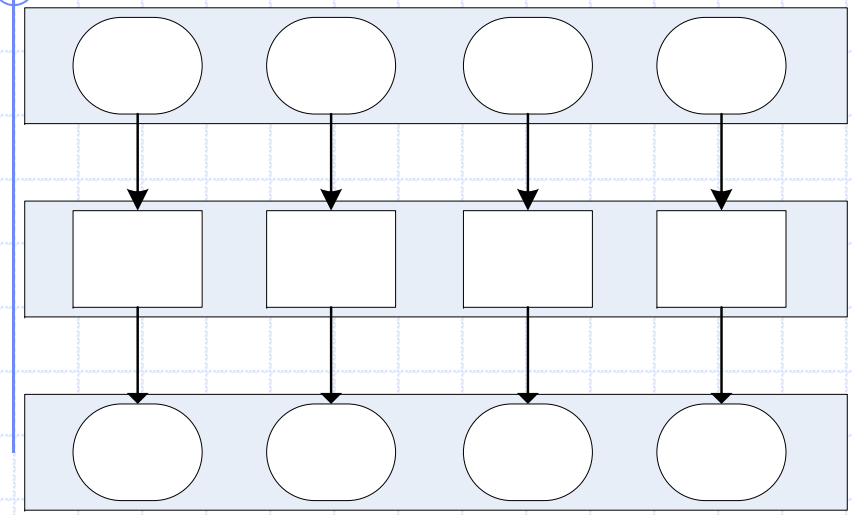
Grananje-Pridruživanje (1/2)

- Kontrolni tok se razgrana na nekoliko paralelnih tokova, koji se kasnije spajaju
- Razni modeli programiranja:
  - Npr. Iskaz s1; s2; … sn; gde se si izvrše paralelno
  - OpenMP grana paralaleni region u više niti
    - koje izvršavaju isti iskaz ili
    - koriste druge konstrukcije da odrede šta niti rade
- Cilk Plus: generalizacija serijskih stabla poziva u paralelna stabla poziva:
  - Funkcija može da bude izmrešćena umesto pozvana

Grananje-Pridruživanje (2/2)

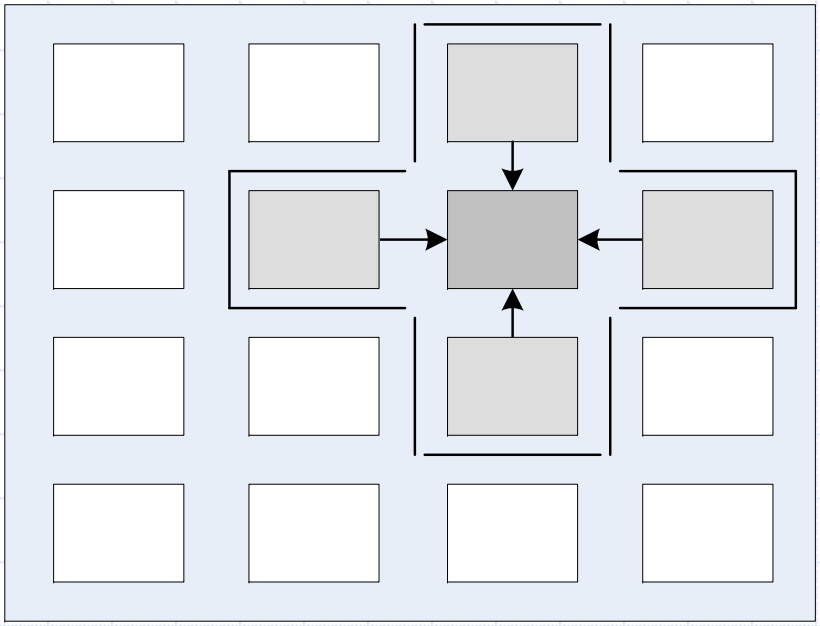
- Cilk Plus mrešćenje:
  - Kontrolni tok grana na pozivajuću i pozvanu funkciju
  - Pozivajuća kasnije izvršava operaciju spajanja (sync)
- Grananje-Pridruživanje != BARIJERA
- BARIJERA je sinhro. konstrukcija preko više niti
  - Tek kad se sve niti skupe, dalje kreću istovremeno
  - Razlika: nakon barijere sve niti nastavljaju izvršenje
  - Nakon spajanja (sync) samo jedna nastavlja izvršenje
  - Barijera imitira spajanje: niti rade isto do sl. Grananja
- Cilk: graf zadataka ugnježden i dvodimenzion

Preslikavanje



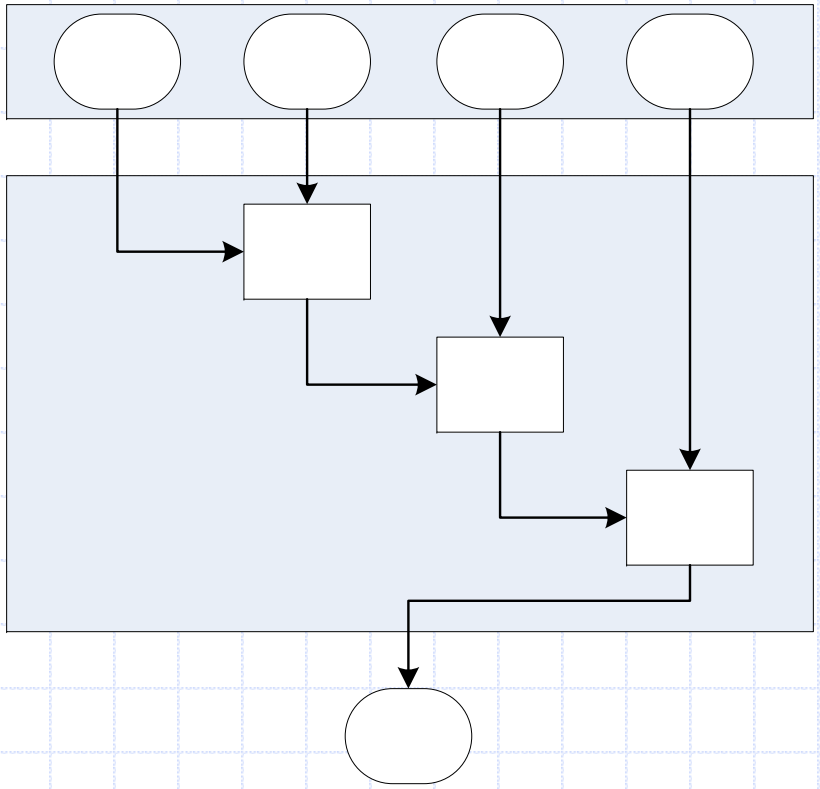
- Preslikavanje se replicira preko skupa indeksa
- OSNOVNA (ELEMENTNA) funkcija
  - Primenjuje se na svaki element ulazne zbirke podataka
  - Mora biti ČISTA (tj. bez ivičnih efekata)
- Zamenjuje iteraciju:
  - Telo zavisi samo od brojača iteracija i
  - podaci se uzimaju korišćenjem brojača iteracija

Obrada suseda



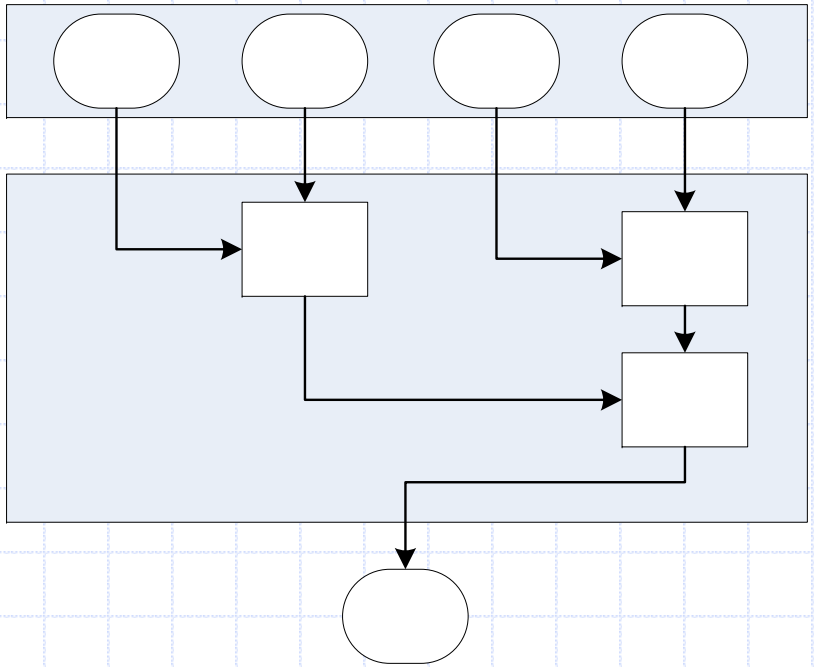
- Generalizuje Preslikavanje
- F-ija elementa i SUSEDA
- SUSEDI: rel. odstojanja
- Popločavanje (eng. tiling):
  - Optimizacija ovog šablona
  - Podaci u keš mem.
- Iteracija + ovaj šablon:
  - Ekvivalentno sa prostorno-vremenskom rekurencijom
- Ivični uslovi kod pristupa nizovima
- Za filtriranje slike, itd.

Redukcija (1/2): Serijska



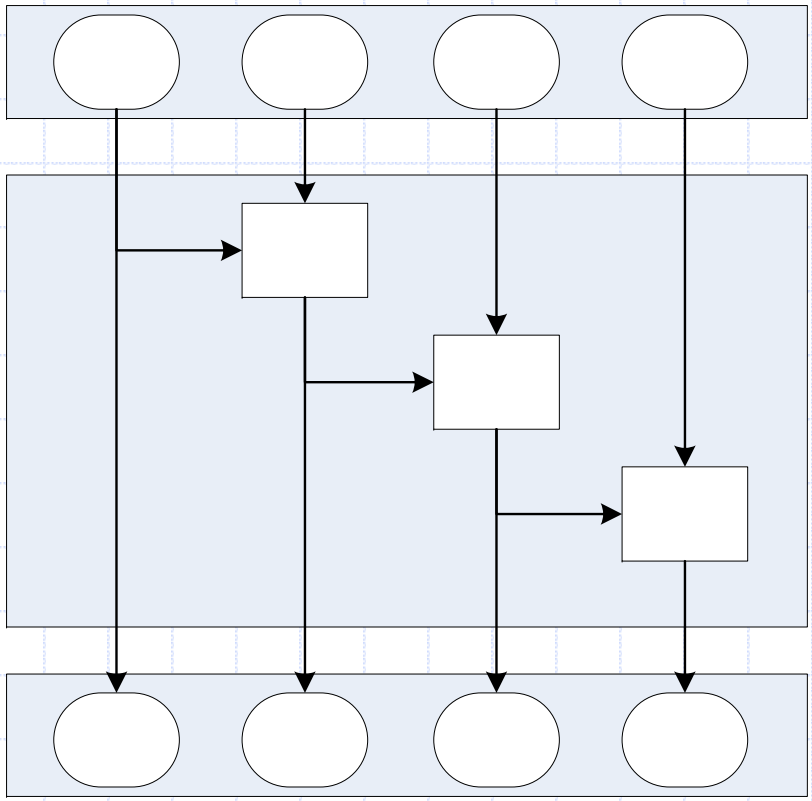
- Kombinuje sve elemente iz zbirke u jedan element
- Kombinujuća funkcija:
  - Asocijativna - mnogi redosledi,
  - ali sa različitim rasponima
  - Ako je komutativna, mogući su i dodatni redosledi

Redukcija (2/2): Paralelna



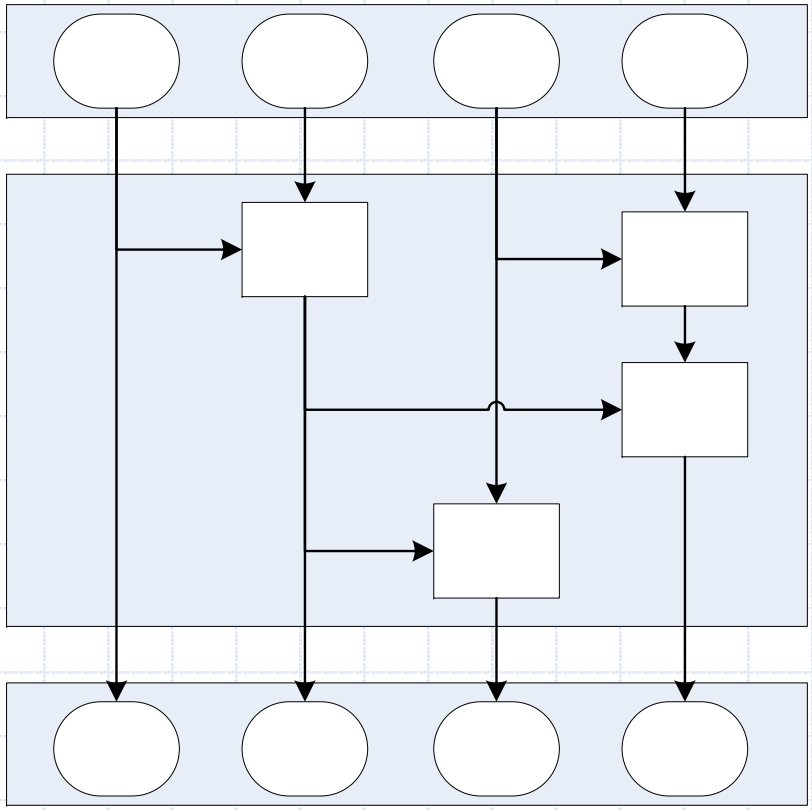
- Redukcija se može paralelizovati korišćenjem strukture stabla
  - Promenom redosleda operacija po zakonu asocijativnosti
  - Paralelizacija redukcije u strukturu stabla
  - Zahteva isti broj operacija kao i serijska verzija
  - Može zahtevati više mem.
- Optimizacija:
  - Lokalna serijska redukcija tehnikom popločavanja
  - Međurezultati se kombinuju dodatnom redukcijom

Skeniranje (1/2): Serijsko



- Svaki el. na izlazu je redukcija el. do te tačke
- Mogućnost paralelizacije:
  - Nije očigledna jer
  - Svaka iteracija zavisi od rezultata prehodnih iteracija
- Skeniranje je spec. slučaj šablona Zamotavanje (Fold)
  - Funkcija naslednika f služi za prelaz iz prethodnog u tekuće stanje na osnovu nekog ulaza
  - Zamotavanje se može paralelizovati ako je f asocijativna, npr. oper. „+“

Skeniranje (2/2): Paralelno



- Paralelizacija skeniranja je manje očigledna
  - Može zahtevati više posla
  - Čak do dva puta više posla
  - U najboljem slučaju raspon od najmanje Θ(lg n)
  - Linearno ubrzanje nije moguće
  - Tražiti alternative
- Skeniranje =
  - Pakovanje + Razbacivanje
  - Pakovanje je deterministično, a Razbacivanje nije

Rekurencija

- Generalizacija petlje
  - ali za sl. gde iteracije mogu biti međusobno zavisne
- Posmatraju se samo proste rekurencije:
  - sa konstantnim odstojanjima između elemenata
- Slična Obradi suseda, ali:
  - susedima se pristupa bilo radi čitanja ili upisa
- POSLEDIČNA: sl. elem. na osnovu predhodnih
- Rekurencije koje se mogu paralelizovati:
  - 1D rekurencija, čiji el. sadrže asocijativne operatore
  - nD rekurencija, koja nastaje iz tela ugnježdenih petlji