6. Zavisnosti po podacima u programskim petljama

6.1. Granice petlje

Granice programske petlje definiše ili programer ili razvojni alat u izvornom programu. Te granice u prevodiocima sekvencijalnih mašina preuzimao je prevodilac i sa takvim unapred definisanim iteracijama, kao ograničenjem, radio optimizacije. Programer ne može da bude opterećen još i analizom zavisnosti po podacima u toku pisanja programa, ali se može pretpostaviti da promene granice programske petlje mogu da dovedu do boljeg kôda za paralelizaciju. Već kod primene Trace scheduling-a na petlje, donekle je razmotavanjem promenjena granica petlje, jer su stapane stare iteracije.

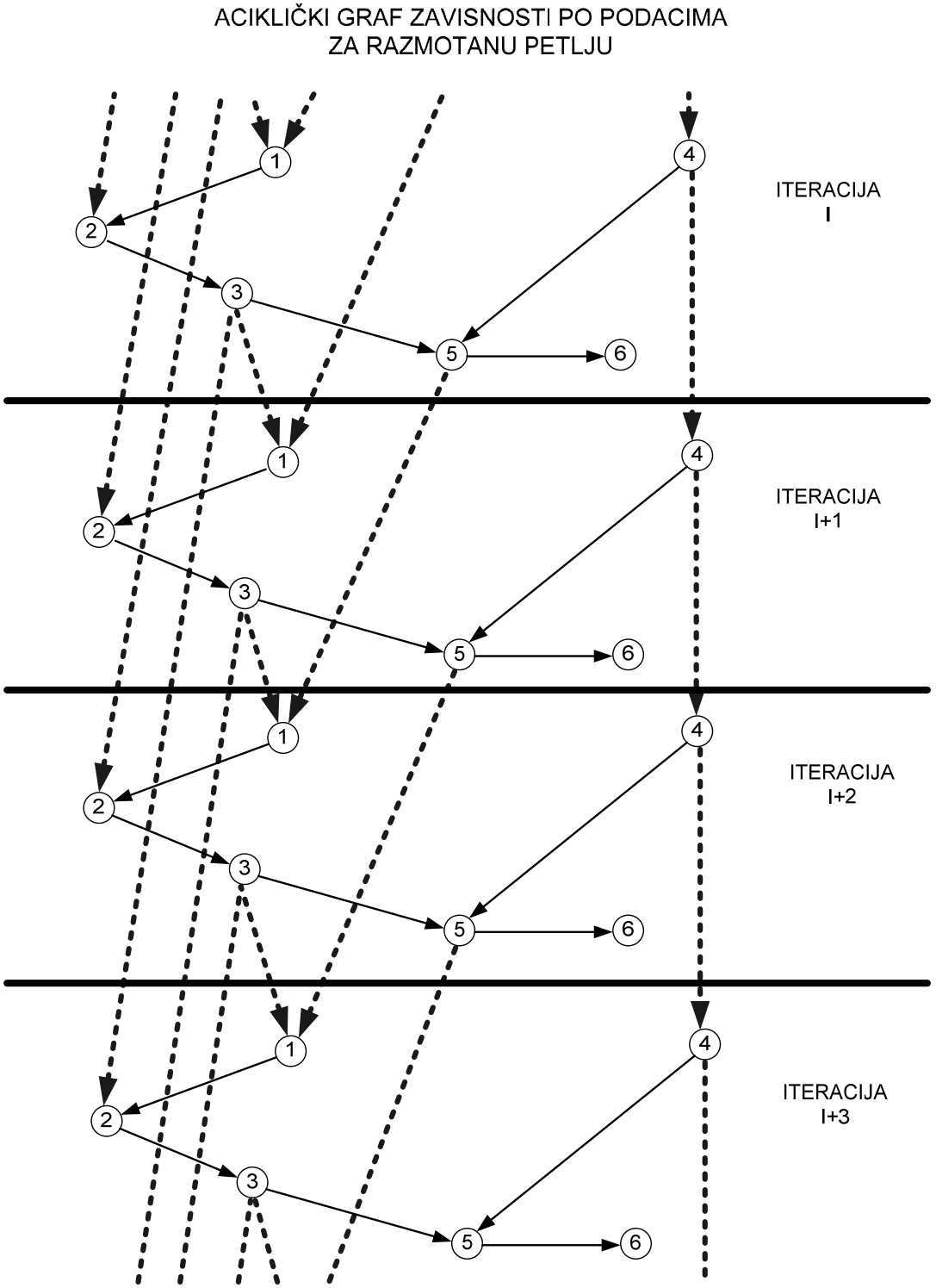
Pretpostavimo da u programskom kôdu postoji jednostavna petlja sa samo tri instrukcije (operacije), označene sa Op_1, Op_2 i Op_3 koja treba da se izvrši n puta. Tada se petlja može predstaviti sa $\{Op_1, Op_2, Op_3\}^n$. Potpuno ekvivalentan kôd dobija se ako se napiše kôd npr. $Op_1, Op_2\{Op_3\ Op_1, Op_2\}^{n-1}\ Op_3$. Ako je operacija Op_3 bila zavisna po podacima od operacije Op_2 unutar originalne iteracije, tada se pojavila zavisnost po podacima preko granica iteracije. Pritom u svakoj novoj iteraciji sem prve, prva operacija Op_3 postaje zavisna od poslednje operacije iz prethodne iteracije. Takve zavisnosti po podacima nazivaju se petljom prenete zavisnosti po podacima (*loop carried data dependencies*). Programer je mogao da u izvornom kôdu napiše bilo koju od te dve prethodne iteracije, pa je samim tim mogao da odredi da li su neke zavisnosti po podacima petljom prenete ili unutar iteracije.

Primer 6.1.
Pretpostavimo da je programer napisao sledeću programsku petlju:

```
        DO I = 3,100
Op1:        A(I) := C(I-1) + E(I-2)
Op2:        B(I) := A(I) * C(I-3)
Op3:        C(I) := B(I) * K1
Op4:        D(I) := D(I-1) + K2
Op5:        E(I) := C(I) + D(I)
Op6:        F(I) := C(I-1) + E(I)
        END
```

Da bi se sagledale sve zavisnosti po podacima, kako one unutar iteracije, tako i one za operacije iz različitih iteracija, najpogodnije je posmatrati zavisnost po podacima na grafu beskonačno razmotane petlje. U cilju sagledavanja zavisnosti po podacima na prihvatljivo velikom uzorku, napravljen je graf zavisnosti po podacima za četiri iteracije. Pritom, horizontalne zadebljane linije predstavljaju granice iteracija i sve grane koje

prelaze granice iteracija prikazuju zavisnosti po podacima operacija iz različitih iteracija (prikazane su isprekidanim linijama). Zbog preglednosti grafa, izostavljena je zavisnost od Op_3 prethodne iteracije ka Op_6 naredne iteracije, jer se radi o tranzitivnoj grani.



Sl. 6.1. Četiri iteracije razmotane petlje iz Primera 6.1.

U programskim petljama radi se repetitivna obrada i rezultati se najčešće smeštaju u strukturirane tipove podataka, pre svega nizove. Određivanje zavisnosti po podacima u programskim petljama za nizove nije u opštem slučaju trivijalno i detaljno je razmatrano u narednom tekstu.

6.2. Generalizacija problema zavisnosti kod petlji i sistemi Diofantskih jednačina

U vreme prevođenja (statički), zavisnosti po podacima ne mogu se generalno ustanoviti kada u programu postoji referenciranje preko pointera (pokazivača). Kada referenciramo pomoću pokazivača (izračunavanje vrednosti pokazivača obavlja se u toku izvršavanja), u vreme prevođenja ne može se znati na koju lokaciju će pokazivati pokazivač. Drugi najvažniji slučaj kada se statički ne mogu uvek utvrditi zavisnosti po podacima je za operacije iz različitih iteracija petlje i on će detaljno biti izložen u ovom poglavlju. Problem zavisnosti po podacima kod petlji biće krajnje generalno postavljen.

Mnogi autori predlažu da se pre analize zavisnosti urade transformacije kojima se olakšava otkrivanje zavisnosti po podacima, i izbacuju redundanse vezane za indekse petlji. Ti postupci su: normalizacija petlji, zamena indeksne promenljive i presavijanje izraza. Opisani su u {AK87} i nisu analizirani u ovom radu. Posmatrajmo prvo najjednostavniji slučaj - zavisnost po podacima za niz koji ima samo jednu dimenziju i kada postoji samo jedna normalizovana programska petlja (nema ugnježdavanja petlji). Ovaj najjednostavniji opšti slučaj može se predstaviti na sledeći način:

```
DO i = 1, N
   X(f(i)) = …
   …….
           = F(X(g(i)))
   …….
END DO
```

Definisanje fiksnog broja iteracija ne umanjuje generalnost petlje, jer će kasnije biti analiziran slučaj kada N nije poznato u vreme prevođenja. Ovde su funkcije f(i) i g(i) izrazi za indeks vektora X, a F je izraz koji sadrži g(i) element vektora X. Pretpostavimo da vektori imaju indekse koji polaze od 1. Funkcije f i g uzimaju vrednosti iz skupa prirodnih brojeva, jer su u pitanju indeksi vektora.

Zavisnost po podacima postoji, ako postoje takvi indeksi i_1 i i_2 da važi:

$$f(i_1) = g(i_2)$$

Na ovaj način otkrivamo i prave zavisnosti i antizavisnosti po podacima. Da bi se analizirale samo prave zavisnosti po podacima, mora se dodati uslov $1 \le i_1 \le i_2 \le N$. Kako se antizavisnosti mogu eliminisati preimenovanjem, u daljoj analizi će se ovaj uslov smatrati jedinim neophodnim da postoji zavisnost po podacima. Oba indeksa petlje i_1 i i_2 moraju istovremeno biti manja ili jednaka od N da bi postojala zavisnost, jer će se

izvršiti samo N iteracija petlje. Prethodno navedenim postupcima normalizacije petlji, početna vrednost indeksa proizvoljne petlje, transformacijama se uvek može svesti na vrednost 1.

Dakle, gornja dva uslova se svode na rešavanje jednačine:

$$f(x) - g(y) = 0$$

u datom opsegu brojeva [1, N], pri čemu mora da važi $1 \le x \le y \le N$. Pritom su x, y, f(x) i g(y) svi iz skupa prirodnih brojeva.

Najčešći slučaj u kôdu je da su f i g linearne funkcije indeksa petlji. Gornja jednačina, se uz tu pretpostavku linearnosti:

$$f(x) = a_1 x + a_0$$

$$g(y) = b_1 y + b_0 ,$$

transformiše u:

$$a_1 x - b_1 y = b_0 - a_0 \qquad (6.1.1.)$$

pri čemu su svi koeficijenti celi brojevi. Najveći broj slučajeva kada f i g nisu linearne funkcije petlji je kada neka od vrednosti koeficijenata a_1, a_0, b_1, ili b_0 nije celobrojna konstanta, već celobrojna promenljiva nezavisna od samog izvršavanja petlji ili parametar potprograma. Dobijeni izraz se može transformisati u opšti oblik linearne Diofantske jednačine koja se za dve promenljive predstavlja kao:

$$ax - by = n \qquad (6.1.2.)$$

Definicija. Linearna Diofantska jednačina je jednačina koja ima oblik:

$$\sum_{i=1}^{N} a_i \cdot x_i = n,$$

gde je $N \ge 1$, $n \in Z$ (Z - skup celih brojeva), $a_i \in Z$ za svako i, postoji bar jedno a_i koje je različito od nule i x_i su celobrojne promenljive.

Linearna Diofantska jednačina sa dve promenljive (6.1.2.) navedena gore, ima rešenje ako i samo ako je najveći zajednički delitelj (NZD) brojeva a i b delilac broja n. Na ovom uslovu baziran je jedan od najjednostavnijih testova utvrđivanja zavisnosti po podacima - takozvani GCD (*Greatest Common Divisor*) test.

6.2.1. GCD test

GCD test zasniva se na teoriji Diofantskih jednačina, koja se odnosi na rešavanje sistema linearnih jednačina, ako je broj jednačina manji od broja nepoznatih. Ovaj test ima nedostatak za paralelizaciju petlji, jer se uslov proverava u celokupnom opsegu celih brojeva, a ne u opsegu vrednosti definisanom granicama petlje.

GCD test važi i za opštu linearnu Diofantsku jednačinu. U daljem tekstu navedena je teorema koja daje potrebne i dovoljne uslove za postojanje rešenja opšte linearne Diofantske jednačine sa N promenljivih.

Teorema o postojanju rešenja linearne Diofantske jednačine. Neka je

$$\sum_{i=1}^{N} a_i \cdot x_i = n$$

linearna Diofantska jednačina i neka je $k = NZD(a_1, a_2, a_3, a_4, \ldots a_n)$. Jednačina ima rešenja ako i samo ako važi da je k delilac broja n.

Dokaz. Dokaz je sproveden samo za N = 2, jer se opšti slučaj dokazuje matematičkom indukcijom. Neka $a \cdot x + b \cdot y = c$ označava linearnu Diofantsku jednačinu, i neka je $g = NZD(a,b)$.

Dokažimo prvo da ako je g delilac broja c, što se piše kao $g|c$, linearna Diofantska jednačina ima rešenja. Dakle, ako važi da je $g = NZD(a,b)$, tada prema poznatoj teoremi (Bezuova teorema) iz teorije brojeva postoje celi brojevi u i v, takvi da važi:

$$g = a \cdot u + b \cdot v$$

Sa druge strane, pošto je g delilac broja c, postoji celi broj c' takav da važi:

$$g \cdot c' = c$$

Iz ovih dveju jednakosti sledi $g \cdot c' = a \cdot u \cdot c' + b \cdot v \cdot c' = a \cdot (u \cdot c') + b \cdot (v \cdot c') = c$. Iz zadnje jednakosti proizilazi da Diofantska jednačina ima rešenje $(u \cdot c', v \cdot c') = (u \cdot c/g, v \cdot c/g)$.

Dokažimo sada da ako linearna Diofantska jednačina ima rešenje, onda mora važiti i da je $g|c$. Pretpostavimo da jednačina ima rešenje (x_0, y_0). Tada važi da je:

$$a \cdot x_0 + b \cdot y_0 = c. \qquad (6.1.3)$$

Iz činjenice da g|a i g|b, direktno zaključujemo da postoje celi brojevi a' i b' tako da važe sledeće jednakosti:

$$g \cdot a' = a \quad \text{i} \quad g \cdot b' = b$$

Jednačinu (6.1.3.) možemo sada napisati na sledeći način:

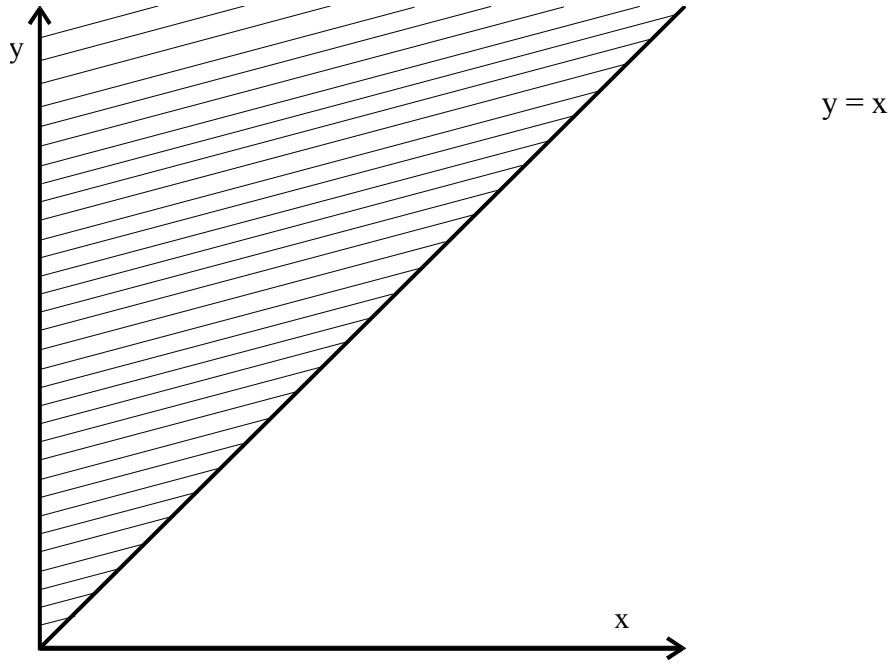
$$g \cdot a' \cdot x_0 + g \cdot b' \cdot y_0 = c.$$

Iz ove jednakosti direktno zaključujemo da i $g|c$, odnosno da je g delilac broja c. □

Iz ove teoreme neposredno se zaključuje da za slučaj jednačine sa dve promenljive (6.1.1.) potreban uslov za postojanje zavisnosti po podacima po GCD testu postaje:

$$NZD(a_1, b_1) \mid (b_0 - a_0),$$

odnosno da je $NZD(a_1, b_1)$ delilac broja $(b_0 - a_0)$. Potrebno je uočiti da je *GCD test* samo potreban uslov za zavisnost po podacima, zato što je neophodno da celobrojna rešenja postoje unutar oblasti $x \leq y$, kao što je prikazano na Slici 6.2. Ova oblast važenja posledica je činjenice da u inicijalnoj petlji, promenljiva iz ranije iteracije ne može da bude zavisna po podacima od operacije iz kasnije iteracije.



Slika 6.2. Oblast prihvatljivih rešenja

GCD test ne uzima u obzir ni granice petlji, pa ukoliko postoji rešenje jednačine, ono može da bude u granicama opsega petlje ili izvan opsega. Zato se postupa na sledeći način:

1. Ako najveći zajednički delitelj od a i b nije delilac broja n, sigurno ne postoji rešenje jednačine, pa samim tim ni zavisnost po podacima.

2. Ako je najveći zajednički delitelj brojeva a i b delilac broja n, onda postoji rešenje u skupu celih brojeva. Zato postoji i zavisnost po podacima za opseg vrednosti indeksa petlje jednak celim brojevima. Međutim, tada zavisnost po podacima u stvarnom opsegu indeksa petlje može, ali ne mora da postoji. Kako ne postoji siguran zaključak, u većini optimizacionih metoda pretpostavlja se da zavisnost po podacima postoji. Ako se sva optimizacija radi u vremenu prevođenja, mora se tada pretpostaviti postojanje zavisnosti, kako bi bili sigurni u ispravnost paralelnog kôda. Ako se optimizacija radi i u toku izvođenja programa, tada se dinamički ponovo proveravaju zavisnosti po podacima, pa će se pogrešna pretpostavka iz vremena prevođenja moći korigovati u toku izvršavanja. Tada se može gubiti samo na brzini izvršavanja, zbog pogrešne pretpostavke o zavisnostima u vreme prevođenja, ali ne i na tačnosti izvršavanja kôda. Ove konstatacije će biti jasnije nakon poglavlja o superskalarnim procesorima.

Kao čest rezultat konzervativnog tumačenja GCD testa u fazi prevođenja, događa se da pretpostavimo da zavisnost po podacima postoji, iako ona stvarno ne postoji u opsegu indeksa petlje. Ovo očigledno predstavlja veliki nedostatak GCD testa, kada se on primenjuje kao jedino sredstvo za određivanje zavisnosti po podacima kod programskih petlji.

6.2.2. Banerjee test

Banerjee test traži sva rešenja Diofantske jednačine samo u oblasti od interesa {BAN 88}. Kod njega se prilikom traženja rešenja uzimaju u obzir i granice petlji koje definišu

oblast u kojoj bi trebalo da se nađu rešenja. Međutim, on daje odgovor na pitanje da li u toj oblasti postoje realna, a ne celobrojna rešenja, uz uslov da se tretiraju samo prave zavisnosti po podacima.

Suština testa zasniva se na sledećem: ako je h(x, y) kontinualna funkcija u domenu realnih brojeva, tada postoji rešenje funkcije h(x, y) = ***a*** u skupu realnih brojeva, gde je a realni broj, ako i samo ako važi sledeći uslov:

$$\min\ h(x, y) < \boldsymbol{a} < \max\ h(x, y)$$

U našem slučaju funkcija h(x, y) je f(x) - g(y), a ***a*** = 0. Pošto su f(x) i g(y) po pretpostavci linearne funkcije, a samim tim i kontinualne, onda je i funkcija h(x, y) = f(x) - g(y) kontinualna. Prema tome, trebalo bi proveriti da li su minimum i maksimum funkcije h(x, y) u domenu $1 \le x \le y \le N$ različitog znaka. Ako je ovaj uslov ispunjen, tada postoji realna nula u datom domenu. To još uvek ne znači da postoji i celobrojna nula.

Kada gornji uslov nije ispunjen, to znači da nema realnih nula, a samim tim nema ni celobrojnih nula, pa sigurno nema ni zavisnosti po podacima. Međutim, kada je gornji uslov ispunjen, ne može se sa sigurnošću tvrditi da li su postojeće nule realne ili celobrojne, pa, ako se oslanjamo samo na statičko određivanje zavisnosti po podacima, moramo pretpostaviti da zavisnost postoji.

Test se jednostavno izvodi pronalaženjem minimuma i maksimuma funkcije h = f - g u zadatom opsegu. Na osnovu razlike znaka minimuma i maksimuma određuje se da li nula pripada opsegu rešenja. Pošto GCD i Banerjee test testiraju da li postoje rešenja polazeći od potpuno drugih polaznih pretpostavki, bilo bi za očekivati da su međusobno komplementarni {TEO 93}, {LA 97}. Opširni testovi različitih autora su pokazali da te komplementarnosti nema.

6.2.3. Totalni test

Totalni test podrazumeva iscrpno ispitivanje

$$f(i_1) = g(i_2)$$

za sve moguće parove vrednosti i_1 i i_2 koje zadovoljavaju uslov $1 \le i_1 \le i_2 \le N$. Ovaj test daje sigurno rešenje, ali ne može se primeniti u prevodiocima, izuzev za male vrednosti N. Ako se pređe na slučaj višedimenzionih nizova u više ugnježdenih petlji, potpuno je neprimenjiv.

6.2.4. Redukovani test

Ovaj test se zasniva na proširenom Euklidovom algoritmu. Osnovna ideja je da se na osnovu ovog algoritma nađe jedno rešenje jednačine. Međutim, pokazano je da su rešenja linearne Diofantske jednačine periodična, i da se na osnovu toga može naći minimalna perioda rešenja, a samim tim i sva rešenja {ZI 90}. Nakon toga pokazano je da ne postoje rešenja koja nisu predstavljena početnim rešenjem na koji se dodaje minimalna perioda pomnožena celim brojem. Dakle, redukovani test nalazi sva rešenja i lako je utvrditi da li su neka od njih u opsegu. Detaljnije se o redukovanom testu može naći u {LA 97}.

6.3. Opšti slučaj višedimenzionog niza u dve ugnježdene petlje

Pretpostavljene su samo dve ugnježdene petlje, uz proizvoljnu dužinu niza. Cilj je samo da se postave relacije zavisnosti, kako bi se uočilo u kojim je aspektima problem kompleksniji. Zbog opštosti, biće pretpostavljeno da svi indeksi višedimenzionog niza zavise od indeksa obe petlje. Taj slučaj može se predstaviti sledećim kôdom:

```
DO i = 1, N
   ……
   DO j = 1, M
     ……
     X(f1(i, j), f2(i, j), f3(i, j),…..) = …
      ……
                                  = F(X(g1(i, j), g2(i, j), g3(i, j),….))
      ……
   END DO
   ……
END DO
```

Prava zavisnost po podacima u okviru petlje postoji, ako za sve vrednosti indeksa k **istovremeno** postoje indeksi i_1, i_2, j_1 i j_2 takvi da važi sledeći uslov:

$$f_k(i_1, j_1) = g_k(i_2, j_2),$$

gde je $k = \overline{1,B}$, B je broj dimenzija nizova X i F, a f_k i g_k prirodni brojevi. Takođe mora biti ispunjen jedan od sledećih uslova vezanih za ograničavanje samo na prave zavisnosti i granice opsega indeksa petlji:

$$1 \le i_1 < i_2 \le N \quad \text{i} \quad j_1, j_2 \in [1, M]$$

ili

$$1 \le i_1 = i_2 \le N \quad \text{i} \quad 1 \le j_1 \le j_2 \le M$$

Dakle, potrebno je naći rešenje sistema od B Diofantskih jednačina i utvrditi da li postoji istovremeno rešenje za sve jednačine. Te Diofantske jednačine po svakoj dimenziji niza mogu se predstaviti u obliku:

$$f_k(x, y) - g_k(z, t) = 0, \qquad k = \overline{1,B} \qquad (6.1.4.)$$

Potrebno je pronaći takve vrednosti za promenljive x, y, z i t da predstavljaju istovremeno rešenje za svaku od jednačina (6.1.4.), i da ispunjavaju uslove:

$$1 \le x < z \le N \quad \text{i} \quad y, t \in [1, M] \qquad (6.1.5.)$$

ili

$$1 \le x = z \le N \quad \text{i} \quad 1 \le y \le t \le M \qquad (6.1.6.)$$

Pošto su $f_k(x, y)$ i $g_k(z, t)$ najčešće linearne funkcije, pretpostavićemo da je svaka jednačina (6.1.4.) sledećeg oblika:

$$a_k \cdot x + b_k \cdot y + c_k \cdot z + d_k \cdot t = n_k \qquad (6.1.7.)$$

Pre samog postupka rešavanja sistema jednačina potrebno je ispitati da li ove jednačine pojedinačno mogu uopšte imati celobrojna rešenja. Za svaku od jednačina moramo prvo ispitati da li je $NZD(a_k, b_k, c_k, d_k)$ delilac broja n_k, jer samo pod tim uslovom uopšte može da postoji celobrojno rešenje samo za svaku tu jednačinu pojedinačno. Uslov postojanja celobrojnog rešenja sistema jednačina je da postoji celobrojno rešenje svake jednačine sistema pojedinačno (generalizacija GCD testa).

Uslov da sve jednačine sistema pojedinačno mogu imati celobrojna rešenja ne znači da zavisnost zaista postoji. Dakle, moguće je da su ta celobrojna rešenja različita za različite jednačine sistema, pa sistem jednačina nema rešenje. Na žalost, generalizovani GCD test ima dvostruki nedostatak:

1. Za svaku pojedinačnu jednačinu pretpostavlja da postoji zavisnost po podacima ako ona ima celobrojna rešenja, mada ne proverava da li su u domenu granica petlji i pravih zavisnosti po podacima,
2. Za sistem jednačina kod kojih svaka jednačina ima pojedinačno celobrojna rešenja pretpostavlja da postoji zajedničko rešenje svih jednačina sistema, iako ne proverava da li je rešenje zajedničko.

Na sličan način generalizuju se i osnovni Banerjee test i Redukovani test. Postoji test koji pokušava da otkrije postojanje istovremenih rešenja svih jednačina i on se zove **λ** test {LI 90}. Njegovo razmatranje nije predmet ove knjige.