2. Paralelizam u kôdu bez programskih petlji

2.1. Prevazilaženje limita u paralelizmu posmatranjem globalnog programa

Da bi se prilikom paralelizacije opisala struktura računarskog programa, neophodno je koristiti ***graf toka kontrole***. On je orijentisani graf koji pokazuje kako se u programu prenosi kontrola između bazičnih blokova. Njegovi čvorovi su bazični blokovi, a orijentisane grane koje izlaze iz blokova ulaze u druge blokove do kojih tok kontrole može doći. U nekim slučajevima se, kao posledica činjenice da uslovna grananja mogu da imaju samo dva ishoda, ograničavaju takvi grafovi tako da najveći broj blokova sledbenika bude 2 i da oni nose oznake T (true) i F (false), saglasno boolean vrednostima (predikatima) koje dovode do prenosa toka kontrole do "pokazanog" sledbenika. U opštijem slučaju, dozvoljava se veći broj sledbenika i definišu predikati (logički izrazi) za svaki mogući ishod. Graf toka kontrole je aciklički ukoliko u kôdu nema programskih petlji. Pod ***dinamičkim tragom*** se podrazumeva jedan niz bazičnih blokova kroz koji program prolazi u toku izvršavanja. Taj niz bazičnih blokova u dinamičkom tragu mora da bude poređan u skladu sa ograničenjima koje definiše graf toka kontrole. Dinamički trag je posledica skupa vrednosti svih ulaznih podataka programa. U ovom delu ćemo se prvo ograničiti samo na kôd bez petlji, zbog postupnog uvođenja mogućih transformacija kôda u cilju paralelizacije. Treba imati u vidu da su prethodno uvedene definicije dinamičkih tragova generalne i uključuju slučaj programskih petlji.

Efikasnu statičku paralelizaciju kôda (u fazi prevođenja) bilo bi lako uraditi, ako bi tada znali tačan redosled izvršavanja bazičnih blokova – dinamički trag. Tada bi se sekvenca bazičnih blokova posmatrala kao jedan ogroman bazični blok, koji bi proizveo veoma veliki graf zavisnosti po podacima i statistički posmatrano, omogućavao da imamo veoma paralelan kôd. Kako se u fazi prevođenja mora uraditi optimizacija kôda za sve moguće dinamičke tragove, neophodno je definisati realističan algoritam kojim bi se približili takvoj idealnoj optimizaciji, ali za sve moguće tragove.

Da bi se uopšte mogla uraditi optimizacija izvan granica bazičnog bloka, neophodno je seliti operacije između bazičnih blokova i time redefinisati pripadnost operacija bazičnim blokovima. Svakako, kandidati za selidbe su samo operacije koje se nalaze na krajevima puteva (lanaca) zavisnosti po podacima u grafu zavisnosti po podacima za bazične blokove. Zato su uvedeni pojmovi operacija slobodnih na vrhu i operacija slobodnih na dnu bazičnih blokova. Operacija je slobodna na vrhu bazičnog bloka ako nije zavisna ni od jedne operacije u bazičnom bloku – to je sinonim za operaciju iz Data_Ready skupa. Operacija je slobodna na dnu bazičnog bloka ako nijedna operacija iz bazičnog bloka nije zavisna po podacima od nje. Operacija može da bude slobodna i na vrhu i na dnu bazičnog bloka, ako nijedna operacija nije zavisna po podacima od nje, niti je ona zavisna po podacima od bilo koje operacije iz tog bazičnog bloka.

Osnovni motiv za uvođenje selidbi operacija između bazičnih blokova je ideja da se operacija slobodna na vrhu kritičnog puta bazičnog bloka može seliti u njemu prethodne bazične blokove, a operacija slobodna na dnu kritičnog puta može seliti u bazične

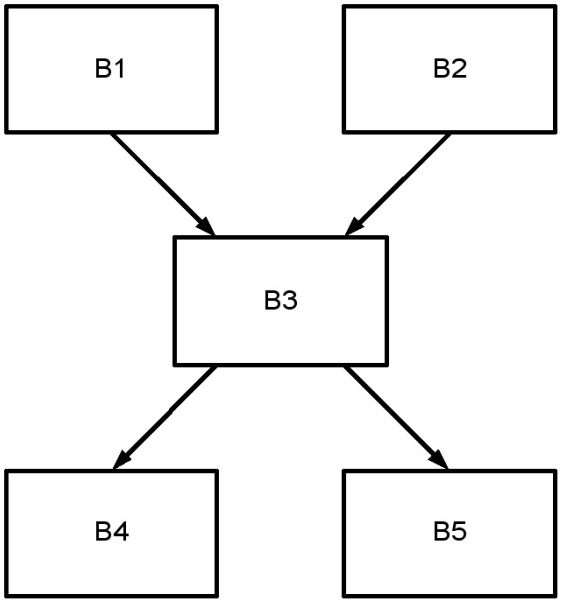
blokove koji njemu slede. Time bi se mogao brže izvršavati bazični blok. Algoritmi koji se koriste za globalnu optimizaciju kôda su znatno kompleksniji od navedene bazične ideje i biće kasnije razmatrani.

Pre nego što se razmatra koje operacije, ako se presele, najviše doprinose paralelizaciji, ppotrebno je definisati pravila kada i pod kojim uslovima su takve selidbe moguće, ako su uopšte moguće. Kada se tom selidbom ne bi produžavali kritični putevi u tim susednim blokovima, a skraćivao bi se kritičan put u bazičnom bloku iz koga se operacija seli, povećao bi se ukupan paralelizam kôda. Naravno da takvo lokalno posmatranje susednih bazičnih blokova ne može dovesti do nivoa paralelizma kao kada se posmatra ceo kôd, ali je nagovešteno kako se može postići ukupno viši nivo paralelizma selidbama operacija između bazičnih blokova. U daljem tekstu se prvo razmatraju osnovna pravila selidbi operacija, da bi se zatim uvela generalizacija tih pravila. Nakon toga će biti definisani globalni kriterijumi za primenu uvedenih pravila za selidbe operacija.

2.2. Osnovne selidbe operacija između bazičnih blokova

Prvo treba definisati pod kojim uslovima smatramo da su transformacije selidbi operacija između bazičnih blokova korektne. U prvom razmatranju, isključivo smatraće se da je kôd istovetan, ako generiše iste rezultate. Važno je uočiti da se prilikom generisanja rezultata mogu događati izuzeci, ali zasada zanemarujemo izuzetke.

Osnovna pravila za selidbu operacija između bazičnih blokova mogu se definisati najjednostavnije preko primera grafa toka izvršavanja elementarnog programa datog na Sl. 2.1. U tom grafu, bazični blokovi B1 i B2 se završavaju sa spojem i nakon tog spoja se nalazi kôd bazičnog bloka B3 koji se završava grananjem. U granama se nalaze bazični blokovi B4 i B5. Elementarni primer uveden je da bi se posmatrali osnovni tipovi selidbi na grananju i spoju {FI 81}. Pritom se transformacije kôda nazivaju selidbama operacija, a osnovni cilj u narednom razmatranju je da se otkriju sve semantički ispravne transformacije.



Sl. 2.1. Selidbe operacija na grananju i spoju

2.2.1. Selidbe preko spoja

Posmatrajmo prvo selidbu operacije preko spoja na gore. U tom slučaju se razmatraju operacije slobodne na vrhu bazičnog bloka B3, jer se jedino one mogu seliti na gore, bez narušavanja zavisnosti po podacima unutar bazičnog bloka B3. Dakle, ako je operacija slobodna na vrhu, može se preseliti preko spoja na gore kopiranjem operacije u obe grane istovremeno. Samim tim se ažuriraju skupovi slobodnih operacija na vrhu i dnu bazičnih blokova B1, B2 i B3. U B1 i B2 preseljena operacija postaje sigurno slobodna na dnu. Operacije iz bazičnih blokova B1 i B2 koje su bile slobodne na dnu, a od kojih je ta preseljena operacija zavisna po podacima, prestaju da budu slobodne na dnu. Ako preseljena operacija nije zavisna ni od jedne operacije u B1 ili B2, ona postaje istovremeno slobodna i na dnu i na vrhu bazičnog bloka u koji je preseljena. Operacije u bazičnom bloku B3, koje su bile zavisne po podacima **samo** od preseljene operacije, postaju slobodne na vrhu bazičnog bloka B3.

Efekat selidbe u bazičnom bloku B3 u ovom slučaju je ili skraćivanje nekog ili nekih puteva u grafu ili nestanak operacije istovremeno slobodne i na vrhu i na dnu u grafu zavisnosti po podacima bazičnog bloka iz koje se operacija seli (B3). Jedan od puteva koji se skraćuje može da bude i kritičan put u bazičnom bloku B3. Istovremeni efekat u bazičnim blokovima B1 i B2 je ili produžavanje nekog ili nekih puteva ili pojava operacije slobodne i na vrhu i na dnu u grafovima zavisnosti po podacima bazičnih blokova u koje je operacija preseljena (B1 ili B2). Neki od tih puteva koji se produžavaju mogu da budu i kritični putevi u bazičnim blokovima B1 i B2.

Opisana transformacija dovodi do ažuriranja skupova slobodnih operacija na vrhu i dnu za sve bazične blokove uključene u transformaciju, čime se otvaraju mogućnosti za selidbe novih operacija koje pre selidbe nisu pripadale skupovima slobodnih operacija na vrhu i na dnu. U konkretnoj selidbi, ako je neka operacija prethodnom selidbom postala slobodna na vrhu u bazičnom bloku B3, tada se ona ponovnom transformacijom selidbe operacije preko spoja na gore može preseliti preko spoja na gore i učiniti neke nove operacije slobodnim na vrhu u bazičnom bloku B3.

Ako posmatramo šta se događa prilikom izvršavanja ovako transformisanog kôda na sekvencijalnoj mašini, može se uočiti da je broj izvršenih operacija identičan bez obzira na dinamički trag koji se izvršava. U ovom pojednostavljenom slučaju, postoje dinamički tragovi koji obuhvataju deo traga (B1,B3) i oni koji obuhvataju deo traga (B2,B3). Jedina razlika nastala opisanom selidbom je da se javljaju identične kopije kôda operacije u obe grane pre spoja, čime se povećava zauzeće programske memorije. Na paralelnoj mašini, ovakva selidba može u principu da dovede i do ubrzavanja i do usporavanja izvršavanja dinamičkih tragova. Sigurno je jedino da transformacija izaziva povećanje veličine programskog kôda kao i kod sekvencijalne mašine, ako se selidba radi u vreme prevođenja. Za ovu selidbu operacija se koristi termin ***selidba operacije preko spoja na gore***. Jedini uslov za korektnost ove transformacije je da operacija pripada skupu slobodnih operacija na vrhu u bazičnom bloku iza spoja. Ako se radi o višestrukom spoju, tada se operacija koja se seli na gore, seli u sve bazične blokove čiji se tokovi kontrole spajaju na tom mestu.

Postoji inverzna transformacija kada u bazičnim blokovima B1 i B2 postoje identične operacije koje su istovremeno slobodne na dnu tih bazičnih blokova. Tada se obe istovremeno mogu preseliti u bazični blok B3, čime se pretvaraju u jednu operaciju koja sigurno postaje slobodna na vrhu bazičnog bloka B3. U ovom slučaju se smanjuje veličina programskog kôda, jer se dve identične kopije kôda operacije iz dva bazična bloka pretvaraju u jedinstvenu kopiju u bazičnom bloku u koji su preseljene. Ova transformacija ugrađena je u programske prevodioce za sekvencijalne mašine bez paralelizma, sa ciljem da se minimizira zauzeće programske memorije, uz postizanje iste brzine izvršavanja. Za ovu selidbu operacija se koristi termin ***selidba operacije(a) preko spoja na dole***. Dvosmislenost oko množine i jednine je posledica prirode transformacije. Ako se posmatra statički, radi se o dvema operacijama koje se stapaju u jednu. Ako se posmatra dinamički, to je jedna operacija koja se može javiti na tragu, jer dinamički, od ta dva traga, samo jedan postoji. Generalizovani uslov za korektnost ove transformacije je da postoje identične operacije slobodne na dnu svih bazičnih blokova iznad spoja.

Ovakvom transformacijom se takođe ažuriraju skupovi slobodnih operacija na vrhu i dnu bazičnih blokova B1, B2 i B3. U B3 preseljena(e) operacija(e) postaje sigurno slobodna na vrhu. Operacije iz bazičnog bloka B3 koje su bile slobodne na vrhu, a koje su zavisne po podacima od preseljene operacije prestaju da budu slobodne na vrhu. Ako ne postoji operacija u B3 zavisna od preseljene operacije, ona selidbom postaje operacija istovremeno slobodna i na dnu i na vrhu bazičnog bloka B3. Sve operacije u bazičnim blokovima B1 i B2 kojima je preseljena operacija bila jedini sledbenik po zavisnosti po podacima postaju slobodne na dnu tih bazičnih blokova. Za razliku od sekvencijalne mašine, kod paralelne mašine ovakva selidba može izazvati promenu trajanja izvršavanja programa i po dinamičkim tragovima koji uključuju deo traga (B1,B3) i po onima koji uključuju deo traga (B2,B3).

2.2.2. Selidbe preko grananja

Selidba preko grananja na dole se može posmatrati preko bazičnih blokova B3, B4 i B5 sa Sl. 2.1., jer se bazični blok B3 završava grananjem. Ako želimo da preselimo neku operaciju preko grananja na dole, mora se ta operacija kopirati u obe grane i naravno mora da istovremeno bude slobodna na dnu bazičnog bloka B3. Dodatno preciziranje operacije slobodne na dnu je neophodno u ovom slučaju: operacija kojom se generiše flag (signal) koji se zatim ispituje kao uslov za skok kod grananja nije u opštem slučaju operacija slobodna na dnu. Zasada će se smatrati da generisanje flag-a (signala) za grananje nije ograničenje za selidbu preko grananja! Identične kopije operacija u bazičnim blokovima B4 i B5 postaju slobodne na vrhu tih bazičnih blokova. Operacije koje su bile slobodne na vrhu u tim bazičnim blokovima, a koje su zavisne po podacima od preseljene operacije prestaju da budu slobodne na vrhu. Operacije od kojih je preseljena operacija bila zavisna po podacima u bazičnom bloku B3 mogu postati slobodne na dnu bazičnog bloka, ako nijedna druga operacija iz B3 nije zavisna po podacima od njih. Osnovni uslov za korektnost ove transformacije je da operacija koja se seli pripada skupu slobodnih operacija na dnu u bazičnom bloku koji se završava grananjem (B3). Dodatni uslov je ograničenje u definisanju operacija slobodnih na dnu, jer neka operacija koja je po ostalim elementima slobodna na dnu, može da generiše uslov za skok (flag), jer ona definiše kontrolne zavisnosti. Kontrolne zavisnosti se detaljnije razmatraju na kraju ovog poglavlja.

Inverzna transformacija je ***selidba dve identične operacije preko grananja na gore***, ako su one slobodne na vrhu bazičnih blokova B4 i B5. One se mogu preseliti istovremeno na gore preko grananja i generisati jednu operaciju slobodnu na dnu bazičnog bloka B3. Kao posledica ove selidbe, neke operacije mogu postati slobodne na vrhu bazičnih blokova B4 i B5, a neke operacije mogu prestati da budu slobodne na dnu bazičnog bloka B3. Odgovarajućim rešenjima se naravno obezbeđuje da tako preseljena operacija ne može da poremeti operaciju kojom se generiše flag (signal) na osnovu koga se obavlja instrukcija skoka. Uslovi za korektnost ove transformacije je da postoje identične operacije koje istovremeno pripadaju skupu slobodnih operacija na vrhu bazičnih blokova iza grananja.

U slučaju ovakvih selidbi, broj operacija koji se mora obavljati na sekvencijalnoj mašini je opet isti, a jedini efekti su povećanje ili smanjenje zauzeća programske memorije. Kod paralelne mašine, ovakva selidba može izazvati promenu vremena izvršavanja programa na jednom ili više dinamičkih tagova.

Kod grananja se javlja još jedna vrsta selidbi koja je modifikacija prethodnih selidbi. Prilikom selidbe operacije slobodne na dnu bloka B3 preko grananja na dole, rezultat operacije se može koristiti u tragovima koji prolaze samo kroz jednu granu. Tada operacija može da se kopira samo u granu u kojoj je rezultat živ {AHO 86}. ***Rezultat je živ*** na nekom mestu ako se, bar na jednom mogućem dinamičkom tragu iza te tačke, obavlja čitanje pre nekog novog upisa. Rezultat je mrtav ako se ni na jednom od svih mogućih dinamičkih tragova ne čita rezultat operacije pre upisa u istu lokaciju. Ako operacije generišu rezultat koji je mrtav, u kompajlerima se eliminišu kao nepotrebne. U slučaju operacije slobodne na dnu bazičnog bloka B3, rezultat bi mogao da bude živ u samo jednoj od grana nakon grananja. Tada se, ako se operacija preseli preko grananja na dole, ne moraju generisati kopije u obe grane, već samo u grani u kojoj je rezultat živ. Ceo postupak se može posmatrati i kao selidba operacije preko grananja na dole sa pravljenjem identičnih kopija u obe grane, a zatim eliminacija mrtvog kôda koju obavlja kompajler (eliminacija operacija koje generišu rezultat koji je mrtav). Za ovu selidbu će se koristiti termin ***Selidba preko grananja na dole u samo jednu granu, kada je rezultat mrtav u drugoj grani.***

Naravno, može se izvesti i inverzna selidba operacije preko grananja na gore, ako je rezultat mrtav u drugoj grani. Tada se operacija slobodna na vrhu jedne od grana seli preko grananja na gore, bez potrebe da postoji takva identična operacija slobodna na vrhu u drugoj grani. Za ovu vrstu transformacije se koristi termin ***Selidba operacije preko grananja na gore kada nema identične operacije slobodne na vrhu u drugoj grani.***

U ovom slučaju se selidbom operacija menja vreme izvršavanja na sekvencijalnoj mašini. Ako operaciju preselimo na dole samo u jednu granu, tada se za sve dinamičke tragove koji prolaze kroz drugu granu izbegava izvršavanje operacije i time skraćuje vreme izvršavanja. Ako operaciju preselimo iz jedne grane na gore preko grananja, a da je rezultat mrtav u drugoj grani, tada se za sekvencijalnu mašinu javlja nepotrebno izvršavanje te operacije za sve dinamičke tragove koji prolaze kroz granu u kojoj je rezultat bio mrtav. Zato se u klasičnim programskim prevodiocima sekvencijalnih mašina selidba na dole preko grananja u samo jednu granu uvek obavlja, kada je to moguće. Međutim, kod paralelne mašine, ako se paralelizacija radi u vreme prevođenja, obično nije kritično koliko se operacija obavlja, jer mašine tipično poseduju dovoljno

paralelizma, već koliko je ukupno trajanje izvršavanja. Zato transformacija kojom se povećava broj operacija koje treba izvršiti može da dovede do ukupno kraćeg trajanja izvršavanja na paralelnoj mašini! Ključna ograničenja u brzini izvršavanja kod paralelnih mašina nisu vezana za ukupan broj operacija koje treba izvršiti, već su to primarno ograničenja koja potiču od pravih zavisnosti po podacima.

2.2.3. Selidbe ciklusa operacije

Odluka o skoku se donosi na kraju nekog ciklusa mašine. Operacije tipično traju veći broj ciklusa, kako je već predstavljeno u modelu prilikom razmatranja paralelizma na nivou bazičnih blokova. Ograničenje kojim bi se uvelo da kompletne operacije mogu da budu samo ili pre ili posle grananja ili spojeva bi dovelo do neiskorišćenih resursa mašine u okolini grananja i spojeva. Zato je neophodno naći način da se operacije rasprostiru svojim ciklusima na susedne bazične blokove. Nameće se pitanje da li je moguće preseliti samo neke cikluse operacije preko grananja i spoja u programu. Da bi se iskoristili resursi mašine, neophodno je upravo da postoje i selidbe dela ukupnog broja ciklusa operacije sa početka ili kraja operacije.

Prva selidba ciklusa koja će biti razmatrana je selidba ciklusa operacije preko spoja na gore. Ako operacija Op_j traje $d(Op_j)$ ciklusa, tada se preko spoja na gore mogu prebaciti ℓ ciklusa ($0 < \ell < d(Op_j)$), a preostalih $d(Op_j) - \ell$ ciklusa se obavlja nakon spoja. Pritom se u programskom kôdu (ako je u pitanju mikrokod sa definisanjem operacija u svim ciklusima) kopiraju delovi kôda iz prvih ℓ ciklusa u obe grane pre spoja. Na taj način se ciklusi operacija iz različitih bazičnih blokova mogu preklapati kao i u slučaju heurističkih algoritama za raspoređivanje (List Scheduling) koji su korišćeni kod bazičnih blokova. Inverzna selidba neće biti posebno razmatrana na ovom mestu.

Selidba operacije preko grananja na dole u ciklusima obavlja se za operacije slobodne na dnu bazičnog bloka, tako što se zadnjih ℓ ciklusa operacije kopira u obe grane ($0 < \ell < d(Op_j)$), dok se prvih $d(Opj) - \ell$ ciklusa nalazi pre grananja. Na ovaj način selidba se mora obaviti samo ako je rezultat operacije Opj živ u obe grane. Ako je rezultat mrtav u nekoj grani, može se izbeći kopiranje zadnjih ℓ ciklusa u obe grane, već se kopiranje tih ℓ ciklusa obavlja samo u jednu granu. Naravno da i u ovom slučaju postoji inverzna selidba, ali ni ona neće biti posebno razmatrana. Selidbe operacija u ciklusima obezbeđuju da pojava grananja i spojeva ne zahteva da resursi mašine ostanu neiskorišćeni u ciklusima oko grananja i spojeva, kada se planiranje paralelizacije kôda radi u vreme prevođenja!

Navedene selidbe operacija preko spoja se mogu raditi bez ikakvih ograničenja. Neke selidbe preko grananja zavise od činjenice da li operacija može da generiše izuzetak. Ako operacija ne može da generiše izuzetak, sve navedene selidbe se mogu raditi bez ograničenja. Kada operacija može da generiše izuzetak, a rezultat je mrtav u jednoj od grana, tada se mora forsirati selidba preko IF-a na dole, samo u granu u kojoj je rezultat živ. To se radi da bi se izbegli nepotrebni izuzeci koji ometaju normalno izvršavanje programa u slučaju kada je tok izvršavanja u granu u kojoj je rezultat mrtav, a izvršavanjem nepotrebne operacije pre IF-a se generiše izuzetak. Dakle, tada se ne sme operacija selidbe operacije iz samo jedne grane zato što je rezultat mrtav u drugoj grani. U poslednjim poglavljima će biti detaljno objašnjena moguća rešenja za ovu selidbu.

2.3. Kontrolne zavisnosti i programski grafovi zavisnosti

U dosadašnjim analizama selidbi operacija, pominjan je problem generisanja flag-a (signala, boolean vrednosti) za uslovni skok kao rezultata operacije koja je po ostalim zavisnostima slobodna na dnu bazičnog bloka koji se završava grananjem. Ta operacija bi morala da se izvrši pre grananja, kako bi se tokom izvršavanja mogla doneti odluka o sledećem bazičnom bloku iza uslovnog skoka u toku kontrole. Dakle, za sve operacije u bazičnim blokovima iza uslovnog grananja postoji zavisnost da li će da se izvršavaju, kao funkcija nekog predikata, odnosno negirane vrednosti istog predikata.

U primeru 2.1.:

```
B:= (A != K)
If (B) then       /* Iskaz 1  */
   C = D/E;    /* Iskaz 2  */
Endif
```

Izvršavanje Iskaz 2 zavisi od toga da li je boolean promenljiva B (predikat) istinita (true, T) ili lažna (false, F). Zavisnost ovog tipa se naziva kontrolna zavisnost. Mada nema suštinske prepreke da se Iskaz 2 izvrši, ***kontrolnom zavisnošću*** se definiše da li nam je rezultat uopšte potreban. Neželjeni upis u lokaciju C i eventualan izuzetak generisan operacijom (npr. za slučaj da je E = 0), na delu kôda koji uopšte nije trebalo da se izvršava su svakako nepoželjni efekti, ako bi se Iskaz 2 izvršio u toku izvršavanja, a kada se na osnovu kasnije izračunate vrednosti predikata ustanovi da to nije bilo potrebno. Ako se pribegava takvom izvršavanju unapred, moraju se potirati svi nepoželjni efekti izvršavanja ovakvih operacija. U daljem tekstu će se podrazumevati da će kontrolno zavisna operacija biti zavisna od predikata, tako da se neće uopšte izvršavati dok ne bude izračunat predikat.

U pojednostavljenom modelu za definisanje ograničenja u paralelizaciji, ovakve zavisnosti se mogu dodati zavisnostima po podacima, kao nezavisni elementi ograničavanja paralelizacije kôda. Kako se boolean uslov (predikat) negde izračunava, to bi dovelo do modeliranja zavisnosti predikata od operacije kojom se on generiše. Isto tako, morali bi se generisati novi čvorovi za dve vrednosti predikata umesto samog predikata i grane zavisnosti ("po podacima") od tih čvorova kao reprezenti kontrolnih zavisnosti. U ovom slučaju su čvorovi istovremeno operacije kojim se određuje uslov i ishodi predikata.

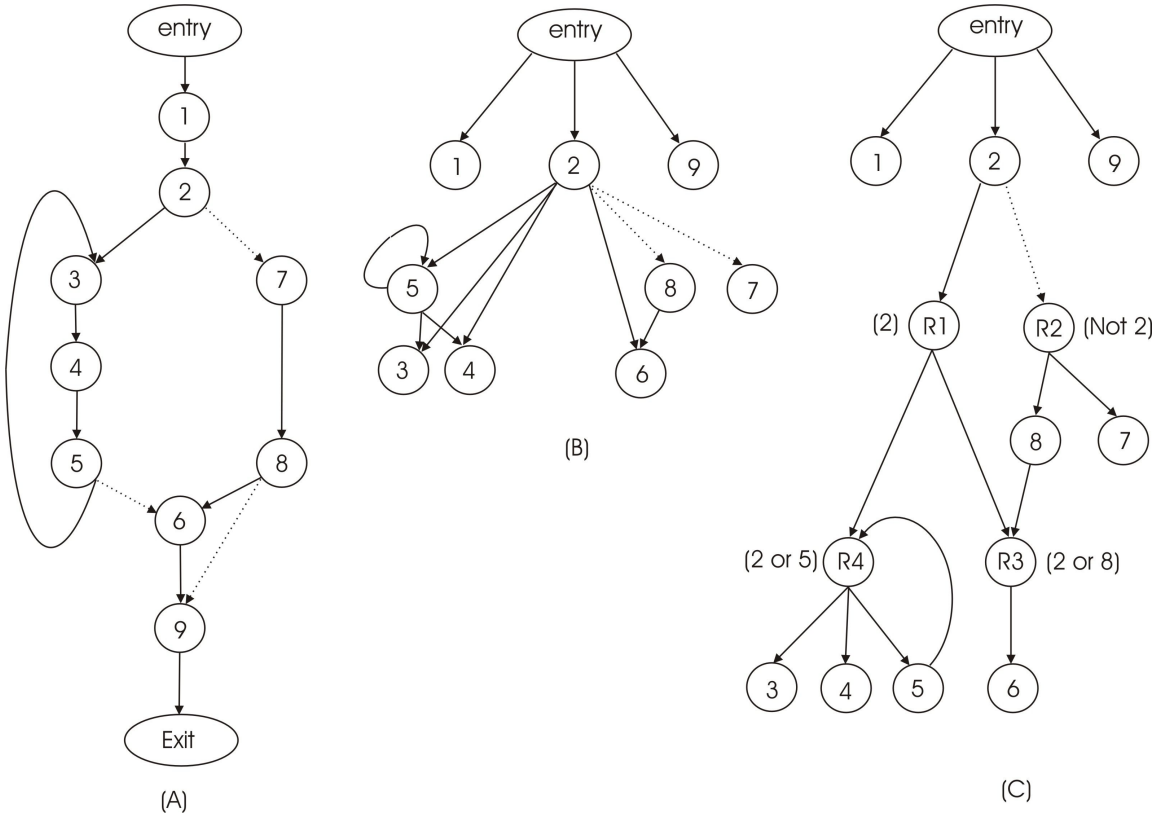
Kontrolne zavisnosti za izvršavanja instrukcija iz grane od vrednosti dobijenog predikata grananja se dakle mogu modelirati zavisnostima po podacima. U grafovskoj predstavi bi to predstavljalo proširenje grafa zavisnosti po podacima sa dodatnim čvorovima i granama proizašlim iz kontrolne zavisnosti. Značenje tih dodatih čvorova nije operacija već operacija i ishod, a pravo značenje grana je suštinski drugačije nego kod pravih zavisnosti po podacima.

Navedeni pojednostavljeni pristup krije u sebi nekoliko mana:

a. Potreba da od instrukcije koja generiše predikat uslovnog skoka generišemo dodatne grane do novih čvorova koji reprezentuju svaki ishod predikata. Osim toga, od tih novih čvorova se generišu dodatne grane ka operacijama sledbenicima u oba bazična bloka iza grananja. Na taj način dolazi do eksplozije broja grana u grafu.
b. Na ovaj način se definiše samo predikat jednog-poslednjeg grananja. U cilju maksimalne paralelizacije, potrebno uvesti globalni predikat kojim se definiše kontrolna zavisnost kao kompleksna logička funkcija većeg broja predikata u prethodnim grananjima.

Da bi se prevazišle navedene mane, uveden je pojam regionskog čvora. On se javlja kao dodatni čvor koji ne prestavlja operaciju. Regionski čvor ima pridružen globalni predikat kojim se definiše kontrolna zavisnost. Grane koje izlaze iz čvora označavaju kontrolnu zavisnost od globalnog predikata. Na ovaj način operacije iz istog bazičnog bloka su uvek zavisne od istog regionskog čvora. Međutim, i **operacije svih bazičnih blokova sa istim globalnim predikatom** na ovaj način postaju zavisne od istog regionskog čvora. Ulazne grane regionskih čvorova koje izlaze iz čvorova za operacije su uvek T (true) ili F (false) grane, jer je izvršavanje neke operacije kojom se generiše flag jedini način da se promeni globalni predikat. Osim takvih grana, postoje još i ulazne grane regionskih čvorova koje izlaze iz regionskih čvorova i ulaze u regionske čvorove. Ovim granama se prikazuju međusobne zavisnosti predikata regionskih čvorova putevi toka kontrole kojima se dolazi do nekog globalnog predikata.
Primer 2.2.

Na Sl. 2.2. prikazan je programski kod sa vrlo malim prosečnim brojem operacija po bazičnom bloku. U delu pod (A) su prikazane sve operacije i ujedno tokovi kontrole i izvršavanja. Ako čvor koji predstavlja operaciju ima izlazne grane sa isprekidanim linijama, to znači da ishod operacije definiše tok kontrole. U navedenom primeru su to operacije 2, 5 i 8, koje rade izračunavanje i generišu flag za uslovni skok. Isprekidana grana tada pokazuje tok kontrole kada je ishod F (false), a puna grana tok kontrole kada je ishod T (true). U delu slike pod (B) je prikazana kontrolna zavisnost pojedinačnih operacija od ishoda prethodnih operacija koje su uticale na kontrolu toka. Sve operacije u programskoj petlji (3, 4, 5) su kontrolno zavisne od izračunavanja uslova za skok same petlje, a nju izračunava operacija 5. One su ujedno kontrolno zavisne od true ishoda flag-a operacije 2 U toj predstavi nema globalnog predikata, pa je teže razlučiti koji su stvarni uslovi za svaku operaciju. Potrebno je uočiti jedan zanimljiv detalj – operacija 6 nije kontrolno zavisna od predikata izračunatog operacijom 5! Razlog za to je pretpostavka da se petlje moraju završiti, odnosno da ne postoje beskonačne petlje U delu pod (C) je prikazana predstava pomoću regionskih čvorova sa globalnim predikatima. Prikazom predikata za svaki regionski čvor i prikazom zavisnosti regionskih čvorova se uočavaju i zavisnosti prilikom izračunavanja predikata i globalni predikat svake operacije. Posebno je interesantno uočiti u ovom primeru kako bazični blokovi mogu imati identične predikate. Primer su operacije 1 i 2 koje čine jedan bazični blok i operacija 9 koja čini drugi bazični blok. Ta dva bazična bloka imaju identične kontrolne zavisnosti. Ovaj slučaj je veoma čest u programskom kôdu. Bazični blokovi pre i posle IF-THEN-ELSE strukture imaju identične kontrolne zavisnosti!



Sl. 2.2. Primer prikaza toka izvršavanja (a), kontrolnih zavisnosti od ishoda grananja (b) i kontrolnih zavisnosti preko regionskih čvorova (c).

Programski grafovi zavisnosti omogućavaju da se jednostavno predstave sva ograničenja u izvršavanju programa. Čak se može zamisliti paralelno izračunavanje kompleksnih predikata u posebnoj jedinici za logička izračunavanja, da bi se izračunali predikati regionskih čvorova. Grafovi zavisnosti po kontroli sa regionskim čvorovima su dobar teorijski model za prikaz globalnih kontrolnih zavisnosti. Danas je osnovni trend paralelizacije da se nekako pokuša uraditi eliminacija kontrolnih zavisnosti, kako bi se povećao paralelizam. U kasnijim poglavljima knjige će biti pokazani različiti načini da se potpuno ili delimično eliminišu kontrolne zavisnosti.