Hoareova logika

Livaja, Petra

SVEUČILIŠTE U SPLITU

PRIRODOSLOVNO MATEMATIČKI FAKULTET

ODJEL ZA MATEMATIKU

PETRA LIVAJA

HOAREOVA LOGIKA

DIPLOMSKI RAD

Split, prosinac 2020.

SVEUČILIŠTE U SPLITU

PRIRODOSLOVNO MATEMATIČKI FAKULTET

ODJEL ZA MATEMATIKU

PETRA LIVAJA

HOAREOVA LOGIKA

DIPLOMSKI RAD

Studentica: Petra Livaja

Mentor: izv. prof. dr. sc. Jurica Perić

Split, prosinac 2020.

Uvod

Računalni program je skup uputa računalu što treba učiniti i kako to izvesti, pisan u nekom programskom jeziku. Slavni britanski računalni znanstvenik Sir Charles Antony Richard Hoare je jednom rekao:

"Najvažnije svojstvo računalnog programa je ostvariti namjere svog korisnika."

Semantika za programski jezik modelira računsko značenje svakog programa. Odnosno, ono što se zapravo događa kada se program izvršava na nekom stvarnom računalu. To izvršavanje je previše složeno da bi se moglo opisati u cijelosti. Umjesto toga, semantika programskog jezika nudi apstraktne entitete koji predstavljaju samo relevantne značajke svih mogućih izvršenja i zanemaruje detalje koji nisu bitni za ispravnost implementacija.

Obično su jedine značajke koje se smatraju relevantnima odnos između ulaza i izlaza te pitanje prekida li se izvršenje ili ne. Detalji specifični za implementacije, kao što su stvarne adrese u računalu na koje su pohranjene vrijednosti varijabli programa, točno vrijeme izvođenja programa i slično, mogu se zanemariti u semantici.

Davanjem formalne semantike programskom jeziku, bavimo se izgradnjom matematičkog modela. Njegova je svrha poslužiti kao osnova za razumijevanje i rasuđivanje o tome kako se programi ponašaju. Ne samo da je matematički

model koristan za razne vrste analiza i provjera, već i na temeljnijoj razini, jer sam pokušaj definiranja značenja programskih konstrukcija može otkriti sve vrste suptilnosti kojih je važno biti svjestan.

Iz povijesnih razloga se smatra da se semantika programskih jezika sastoji od triju sastavnica:

- *Operacijska semantika* opisuje značenje programskog jezika navođenjem načina izvršavanja na apstraktnom računalu, koje je teoretski model programske podrške korištene u teoriji automata. Neki od primjera apstraktnih strojeva su konačni automati (KA), potisni automati (PA), Turingovi strojevi (TS), itd.

- *Denotacijska semantika* je tehnika za određivanje značenja programskih jezika koju je pokrenuo C. Strachey i kojoj je matematički temelj dala Dana Scott. U jednom trenutku nazvana "matematička semantika" ona koristi apstraktnije matematičke koncepte.

- *Aksiomatska semantika* pokušava popraviti značenje programskih konstrukcija uvođenjem pravila provjere unutar programske logike.

Aksiomatsku semantiku je prvenstveno razvio britanski računalni znanstvenik Sir Charles Antony Richard Hoare krajem 1960-ih. Glavni cilj je u početku bio pružiti formalnu osnovu za provjeru ispravnosti apstraktnih algoritama. Kasnije je okosnica primijenjena na definiciju programskih jezika, a razmatranje aksiomatske semantike je utjecalo na dizajn brojnih programskih jezika kao što je imperativni programski jezik Pascal.

Paradigma programskog jezika podrazumijeva konceptualizaciju opisa programa, odnosno sintakse, i izvođenja programa, odnosno semantike. Programske paradigme pomažu razvrstati programske jezike na temelju njihovih značajki.

Središnji koncept imperativne programske paradigme jest instrukcija ili naredba. Stroj tumači svaku instrukciju u programu na način da ona mijenja stanje stroja. Imperativno programiranje je usredotočeno na opisivanje rada programa. Imperativni programski jezici sadrže strukture poput if, else, while, for petlje, klase, objekte i funkcije. Programski jezici kao što su Java, C i C# su imperativni programski jezici.

Ovaj rad prezentira imperativni programski jezik IMP, čije je ime skraćenica od riječi *imperativan*. On je rani sistemski programski jezik, kojeg je razvio Edgar T. Irons krajem 1960-ih do ranih 1970-ih u Agenciji za nacionalnu sigurnost (NSA). IMP se naziva imperativnim jezikom, jer izvršavanje programa uključuje izvođenje niza naredbi za promjenu stanja. U nastavku želimo sustavno provjeriti da program napisan u IMP-u radi ono što od njega zahtijevamo. Stoga, uvodimo Hoareova pravila.

Hoareova logika (poznata pod nazivima Hoareova pravila ili Floyd-Hoareova logika) je formalni sustav s nizom logičkih pravila za rasuđivanje o ispravnosti računalnih programa. Izvorne ideje nalaze se u radu Roberta W. Floyda, koji je objavio sličan sustav. Hoareova logika daje pravila za odnos između tvrdnji o vrijednostima varijabli prije i nakon izvršavanja svake programske konstrukcije. Obično su dotične konstrukcije samo iskazi.

Smatramo prikladnim podijeliti ovaj rad na uvod u jezik IMP, te njegovu aksiomatsku semantiku, a nastaviti u skladu s konstrukcijama programskog jezika: while programi, rekurzivne procedure bez parametara, deklaracija varijabli (lokalne varijable), subskriptne varijable i mehanizme parametara.

Cilj ovog rada je predočiti najrelevantnije probleme koji se odnose na Hoareovu logiku (naime, one koji se odnose na utemeljenost i potpunost) na sistematičan način.

Sadržaj

Poglavlje 1

Uvod u IMP

Ovo poglavlje predstavlja sintaksu programskog jezika IMP, jednostavnog jezika while programa. On je rani sistemski programski jezik, kojeg je razvio Edgar T. Irons krajem 1960-ih do ranih 1970-ih u Agenciji za nacionalnu sigurnost (NSA). IMP se naziva "imperativnim" jezikom, jer izvršavanje programa uključuje izvršavanje niza eksplicitnih naredbi za promjenu stanja. Jezik IMP ima aritmetičke izraze koji uključuju domenu proizvoljno velikih cijelih brojeva, logičke izraze, izraze dodjeljivanja vrijednosti, uvjetne izraze, while petlje i sekvencijalnu kompoziciju izraza. Očekuje se da će sve varijable korištene u IMP programu biti deklarirane na početku programa. Formalno je ponašanje IMP-a opisano pravilima koja određuju kako se evaluiraju njegovi izrazi i izvršavaju njegove naredbe. Pravila pružaju operacijsku semantiku IMP-a.

1.1 IMP - jednostavan imperativni jezik

Za početak nabrojimo skupove pomoću kojih gradimo sintaksu IMP-a:

- N, skup brojeva koji se sastoji od elemenata skupa $\mathbb{Z}$

- $T = \{true, false\}$, skup istinosnih vrijednosti
- Loc, skup lokacija
- $Aexp$, skup aritmetičkih izraza
- $Bexp$, skup Booleovih (logičkih) izraza
- Com, skup naredbi

Pretpostavljamo da je zadana sintaktička struktura brojeva i lokacija. Na primjer, skup Loc se može sastojati od nepraznih nizova slova ili takvih nizova koje slijede znamenke i sl. U ovom neformalnom predstavljanju sintakse upotrijebit ćemo pomoćne varijable, koje će poprimati vrijednosti iz gore navedenih sintaktičkih skupova. Te varijable koje nam govore nešto o promatranom programskom jeziku nazivamo *metavarijablama*.

1.1.1 Konvencija

Predstavljajući sintaksu IMP-a slijedit ćemo sljedeću konvenciju:

- n, m mogu imati vrijednosti iz skupa brojeva N
- X, Y mogu imati vrijednosti iz skupa lokacija Loc
- a može imati vrijednost iz skupa aritmetičkih izraza $Aexp$
- b može imati vrijednost iz skupa logičkih izraza $Bexp$
- c može imati vrijednost iz skupa naredbi Com

Metavarijable koje koristimo mogu se subskriptirati i superskriptirati. Tako, npr. X, X', X_0, X_1, Y'' označavaju elemente skupa Loc.

1.1.2 Pravila formiranja IMP-a

U nastavku slijede pravila formiranja za cijeli IMP.

Za $Aexp$:

1. Za $n \in N$ s $a := n$ definiran je aritmetički izraz, tj. $a \in Aexp$;
2. Za $X \in Loc$ s $a := X$ definiran je aritmetički izraz, tj. $a \in Aexp$;
3. Ako su $a_1, a_2 \in Aexp$ aritmetički izrazi, tada su $a_0 + a_1, a_0 - a_1$ i $a_0 \cdot a_1$ također aritmetički izrazi.
4. Izraz u IMP-u je aritmetički izraz ako je nastao primjenom konačno mnogo koraka uvjeta 1, 2 i 3.

Za $Bexp$:

1. Za istinosnu vrijednost iz skupa $T = \{true, false\}$ definiran je Booleov izraz, tj. za $b := true$ i $b := false$ vrijedi $b \in Bexp$;
2. Ako su $a_0, a_1 \in Aexp$ aritmetički izrazi, tada su s $b := (a_0 = a_1)$ i $b := (a_0 \leq a_1)$ definirani Booleovi izrazi;
3. Ako su $b_0, b_1 \in Bexp$ Booleovi izrazi, tada su $\neg b, b_0 \wedge b_1$ i $b_0 \vee b_1$ također Booleovi izrazi;
4. Izraz u IMP-u je Booleov izraz ako je nastao primjenom konačno mnogo koraka uvjeta 1, 2 i 3.

Za Com:

1. Izrazima $c :=$skip i $c := (X := a)$, $(\forall X \in Loc, \forall a \in Aexp)$ definirane su naredbe iz skupa Com;
2. Ako su $b \in Bexp,\ c_0, c_1 \in Com$, tada su $c_0; c_1$, if b then c_o else c_1 i while b do c također naredbe iz Com;

3. Izraz u IMP-u je naredba ako je nastao primjenom konačno mnogo koraka uvjeta 1 i 2.

Gore spomenute logičke veznike redom nazivamo: $\neg$ negacija, $\wedge$ konjunkcija, $\vee$ disjunkcija. Sada nam je potrebna notacija da bismo izrazili da su dva elementa e_0, e_1 istog sintaktičkog skupa identična, u smislu da su izgrađeni na potpuno isti način.

1.2 Evaluacija aritmetičkih izraza ($Aexp$)

Temelj koji stoji u većini modela koji opisuju ponašanje programa je ideja *stanja* određenog sadržajem na lokaciji. U odnosu na stanje aritmetički izraz se evaluira na cijeli broj, a Booleov izraz na istinosnu vrijednost ($true$ ili $false$).

Rezultirajuće vrijednosti mogu utjecati na izvršenje naredbi, što će dovesti do promjene stanja. U ovom radu formalni opis ponašanja IMP-a će slijediti ovaj smjer: prvo definiramo stanja, zatim evaluaciju cjelobrojnih i logičkih izraza te na kraju izvršenje naredbi. Za početak, uvodimo nove pojmove:

- funkcija $\sigma : Loc \to N$ sa skupa lokacija na skup brojeva koju nazivamo *stanjem*
- Σ, skup stanja

Napomena 1.1. Pišući $\sigma : Loc \to N$ ne zahtijevamo da σ mora biti definirana na cijelom skupu Loc, već može biti definirana na nekom njegovom podskupu.

Tako vrijednost $\sigma(X)$ interpretiramo kao sadržaj lokacije X u stanju σ, za $\sigma \in \Sigma,\ X \in Loc$.

Promotrimo evaluaciju aritmetičkog izraza $a \in Aexp$ u stanju $\sigma \in \Sigma$. Parom $\langle a, \sigma \rangle$ možemo predstaviti situaciju da izraz *a čeka na evaluaciju u stanju* σ.

Definirat ćemo evaluacijsku relaciju između takvih parova i brojeva

$$\langle a, \sigma \rangle \to n$$

koja ima značenje: izraz a u stanju σ se evaluirao u n.

Parove $\langle a, \sigma \rangle$, gdje je a aritmetički izraz i σ stanje nazivamo *konfiguracije* aritmetičkih izraza.

Neka su $n, n_0, n_1 \in N,\ \sigma \in \Sigma,\ X \in Loc,\ a_0, a_1 \in Aexp$. Možemo odrediti evaluacijsku relaciju na način usmjeren prema sintaksi sljedećim pravilima:

Evaluacija brojeva

$$\langle n, \sigma \rangle \to n$$

Prema tome, bilo koji broj je već evaluiran u samog sebe.

Evaluacija lokacija

$$\langle X, \sigma \rangle \to \sigma(X)$$

Prema tome, lokacija se evaluira u svoj sadržaj u tom stanju.

Evaluacija sume

$\dfrac{\langle a_0, \sigma \rangle \to n_0 \ \ \langle a_1, \sigma \rangle \to n_1}{\langle a_0 + a_1, \sigma \rangle \to n}$, gdje je n suma brojeva n_0 i n_1.

Evaluacija oduzimanja

$\dfrac{\langle a_0, \sigma \rangle \to n_0 \ \ \langle a_1, \sigma \rangle \to n_1}{\langle a_0 - a_1, \sigma \rangle \to n}$, gdje je n rezultat oduzimanja brojeva n_1 od n_0.

Evaluacija umnoška

$\dfrac{\langle a_0, \sigma \rangle \to n_0 \ \ \langle a_1, \sigma \rangle \to n_1}{\langle a_0 \cdot a_1, \sigma \rangle \to n}$, gdje je n umnožak brojeva n_0 i n_1.

Kako čitamo takva pravila?

Pravilo sume se može pročitati na sljedeći način: ako $\langle a_0, \sigma \rangle \to n_0$ i $\langle a_1, \sigma \rangle \to n_1$ onda $\langle a_0 + a_1, \sigma \rangle \to n$, gdje je n zbroj brojeva n_0 i n_1.

Pravilo ima *premisu* (pretpostavku) i *konkluziju* (zaključak), a mi smo slijedili uobičajenu praksu pisanja pravila s premisom iznad i konkluzijom ispod ravne linije.

Neka pravila, poput onih za evaluaciju brojeva ili lokacija, ne zahtijevaju premisu. Ponekad se pišu s linijom, npr. kao

$$\frac{}{\langle n, \sigma \rangle \to n}.$$

Pravila s praznim premisama se nazivaju *aksiomima*.

Gornja pravila samo izražavaju kako se lokacije i operacije $+, -, \cdot$ mogu eliminirati iz izraza kako bi se dobio broj u koji se oni evaluiraju.

1.3 Evaluacija Booleovih izraza ($Bexp$)

Neka su $n, m \in N,\ \sigma \in \Sigma,\ a_0, a_1 \in Aexp,\ b_0, b_1, b \in Bexp,\ t_0, t_1, t \in T$. Booleove izraze evaluiramo na istinosne vrijednosti iz skupa $T = \{true, false\}$ koristeći sljedeća pravila:

$$\langle true, \sigma \rangle \to true$$

$$\langle false, \sigma \rangle \to false$$

$$\frac{\langle a_0, \sigma \rangle \to n \ \langle a_1, \sigma \rangle \to m}{\langle a_0 = a_1, \sigma \rangle \to true} \text{ ako su } n \text{ i } m \text{ jednaki}$$

$$\frac{\langle a_0, \sigma \rangle \to n \ \langle a_1, \sigma \rangle \to m}{\langle a_0 = a_1, \sigma \rangle \to false} \text{ ako su } n \text{ i } m \text{ različiti}$$

$$\frac{\langle a_0, \sigma \rangle \to n \ \langle a_1, \sigma \rangle \to m}{\langle a_0 \leq a_1, \sigma \rangle \to true} \text{ ako je } n \text{ manje ili jednako } m$$

$$\frac{\langle a_0, \sigma\rangle \to n \ \langle a_1, \sigma\rangle \to m}{\langle a_0 \leq a_1, \sigma\rangle \to false} \text{ ako ne vrijedi da je } n \text{ manje ili jednako } m$$

$$\frac{\langle b, \sigma\rangle \to true}{\langle \neg b, \sigma\rangle \to false}$$

$$\frac{\langle b, \sigma\rangle \to false}{\langle \neg b, \sigma\rangle \to true}$$

$$\frac{\langle b_0, \sigma\rangle \to t_0 \ \ \langle b_1, \sigma\rangle \to t_1}{\langle b_0 \wedge b_1, \sigma\rangle \to t}, \ \text{ gdje je } t = \begin{cases} \text{true}, & \text{ako } t_0 \equiv true \text{ i } t_1 \equiv true \\ \text{false}, & \text{inače} \end{cases}$$

$$\frac{\langle b_0, \sigma\rangle \to t_0 \ \ \langle b_1, \sigma\rangle \to t_1}{\langle b_0 \vee b_1, \sigma\rangle \to t}, \ \text{ gdje je } t = \begin{cases} \text{true}, & \text{ako } t_0 \equiv true \text{ ili } t_1 \equiv true \\ \text{false}, & \text{inače} \end{cases}$$

1.4 Izvođenje naredbi (Com)

Uloga aritmetičkih i Booleovih izraza tj. elemenata skupova $Aexp$ i $Bexp$ je da evaluiraju vrijednosti u određenom stanju. Uloga programa, pa tako i naredbi iz skupa Com, je da se izvrši da bi došlo do promjene stanja. Kad izvršimo IMP program, pretpostavit ćemo da je u početku stanje takvo da su sve promatrane lokacije postavljene na nulu. Dakle, *inicijalno stanje* $\sigma_0 \in \Sigma$ ima svojstvo $\sigma_0(X) = 0$ za sve lokacije $X \in Loc$. Izvođenje se može završiti u konačno mnogo koraka u završnom stanju ili može nikada ne donijeti završno stanje.

Par $\langle c, \sigma\rangle$ predstavlja *konfiguraciju* iz koje ostaje izvršiti naredbu $c \in Com$ iz stanja $\sigma \in \Sigma$. Definiramo relaciju

$$\langle c, \sigma\rangle \to \sigma'$$

koja označava (potpuno) izvršavanje naredbe c u stanju σ koja se završava u stanju $\sigma' \in \Sigma$.

Napomena 1.2. Neka je $\sigma \in \Sigma$ stanje, $m \in N,\ X, Y \in Loc$. Pišemo $\sigma[m/X]$ za stanje dobiveno iz σ zamjenjujući njegov sadržaj u X sa m, tj. definiramo

$$\sigma[m/X](Y) = \begin{cases} m, & \text{ako je } Y = X \\ \sigma(Y) & \text{ako je } Y \neq X \end{cases}$$

Primjer 1.3. Sada umjesto

$$\langle X := 5, \sigma \rangle \to \sigma'$$

možemo pisati

$$\langle X := 5, \sigma \rangle \to \sigma[5/X],$$

gdje je σ' zapravo stanje σ s promijenjenom vrijednosti 5 na lokaciji X, što sada smijemo označavati s $\sigma[5/X]$.

Relacija izvršavanja za proizvoljne naredbe i stanja dana je sljedećim pravilima:

Pravila za naredbe

Atomarne naredbe:

$$\langle skip, \sigma \rangle \to \sigma$$

$$\frac{\langle a, \sigma \rangle \to m}{\langle X := a, \sigma \rangle \to \sigma[m/X]}$$

Nizanje:

$$\frac{\langle c_0, \sigma \rangle \to \sigma'' \ \ \langle c_1, \sigma'' \rangle \to \sigma'}{\langle c_0; c_1, \sigma \rangle \to \sigma'}$$

Kondicionali:

$$\frac{\langle b, \sigma \rangle \to true \ \ \langle c_0, \sigma \rangle \to \sigma'}{\langle \text{if } b \text{ then } c_0 \text{ else } c_1, \sigma \rangle \to \sigma'}$$

$$\frac{\langle b, \sigma\rangle \to false \;\; \langle c_1, \sigma\rangle \to \sigma'}{\langle \text{if } b \text{ then } c_0 \text{ else } c_1, \sigma\rangle \to \sigma'}$$

While petlje:

$$\frac{\langle b, \sigma\rangle \to false}{\langle \text{while } b \text{ do } c, \sigma\rangle \to \sigma}$$

$$\frac{\langle b, \sigma\rangle \to true \;\; \langle c, \sigma\rangle \to \sigma'' \;\; \langle \text{while } b \text{ do } c, \sigma''\rangle \to \sigma'}{\langle \text{while } b \text{ do } c, \sigma\rangle \to \sigma'}$$

gdje su $\sigma', \sigma'', \sigma \in \Sigma,\ m \in N,\ a \in Aexp,\ X \in Loc,\ b \in Bexp,\ c_0, c_1, c \in Com$.

Poglavlje 2

Aksiomatska semantika IMP-a

Semantika za programski jezik modelira računsko značenje svakog programa. Odnosno, ono što se zapravo događa kada se program izvršava na nekom stvarnom računalu. Obično su jedine relevantne značajke svih mogućih izvršenja programa zapravo odnos između ulaza i izlaza te pitanje prekida li se izvršenje ili ne. Davanjem formalne semantike programskom jeziku bavimo se izgradnjom matematičkog modela. Njegova je svrha poslužiti kao osnova za razumijevanje i rasuđivanje o tome kako se programi ponašaju. Aksiomatsku semantiku razvio je prvenstveno Hoare krajem 1960-ih. Glavni cilj je u početku bio pružiti formalnu osnovu za provjeru apstraktnih algoritama. Hoareova logika daje pravila za odnos između tvrdnji o vrijednostima varijabli prije i nakon izvršavanja svake programske konstrukcije. U ovom poglavlju se okrećemo sustavnoj provjeri programa u IMP-u.

2.1 Ideja

Promotrimo problem kako dokazati da program kojeg smo napisali u IMP-u radi ono što od njega zahtijevamo. Započnimo razmatranje s jednostavnim primjerom.

Primjer 2.1. *Produkt prvih 20 prirodnih brojeva*

Neka je dan program u IMP-u za računanje $\prod_{1 \leq m \leq 20} m$.

$$P := 1;$$

$$N := 1;$$

$$(\text{while } \neg(N = 21) \text{ do } P := P \cdot N; N := N + 1)$$

gdje su $P, N \in Loc$.

Kako bismo dokazali da je ovaj program, kada se završi, takav da je vrijednost od P jednaka $\prod_{1 \leq m \leq 20} m$?

Naravno, jedna stvar koju bismo mogli učiniti je da pokrenemo program u skladu s našom operativnom semantikom i vidimo što ćemo dobiti. Pretpostavimo da malo promijenimo program na način da umjesto "while $\neg(N = 21)$ do" stavimo "while $\neg(N = S + 1)$ do" i da smo proizvoljno dodijelili vrijednost S prije pokretanja programa. U ovom slučaju, rezultirajuća vrijednost od P bi nakon izvršenja trebala biti $\prod_{1 \leq m \leq S} m$, bez obzira na početnu vrijednost od S. Obzirom da S može imati beskonačno mnogo početnih vrijednosti koje smo mogli proizvoljno odabrati prije pokretanja programa, tu činjenicu ne možemo opravdati jednostavnim pokretanjem programa za sve početne vrijednosti S. Vidimo da moramo koristiti logiku za razmišljanje o programu.

Završit ćemo formalnim sustavom dokaza svojstava programa IMP, temeljenim na pravilima dokazivanja za svaku programsku konstrukciju IMP-a. Njegova pravila nazivaju se Hoareovim pravilima ili Floyd-Hoareovim pravilima. U

početku se njihov pristup zalagao ne samo za dokazivanje svojstava programa, već i za pružanje metode za objašnjenje značenja programskih konstrukcija: značenje konstrukcije bilo je specificirano u smislu "aksioma" koji govore kako dokazati njegova svojstva. Iz tog razloga, pristup se tradicionalno naziva aksiomatska semantika.

Vratimo se na prethodni primjer.

Odmah vidimo da za lokacije $P, N \in Loc$ naredbe $P := 1;\ N := 1 \in Com$ inicijaliziraju vrijednosti na lokacijama. Stoga programu dodajemo komentar u vitičastim zagradama. Želimo da metoda opravda posljednji komentar u:

$$P := 1; N := 1;$$
$$\{P = 1 \wedge N = 1\}$$
$$(\text{while } \neg(N = 21) \text{ do } P := P \cdot N; N := N + 1)$$
$$\{P = \textstyle\prod_{1 \leq m \leq 20} m\}$$

što znači da ako je $P = 1 \wedge N = 1$ prije izvršavanje while petlje, tada je $P = \prod_{1 \leq m \leq 20} m$ nakon izvršavanja.

Naravno, kod jednostavnog primjera poput našeg, nije teško provjeriti da je u svakoj iteraciji petlje:

$$P = 1 \cdot 2 \cdot 3 \cdot ... \cdot (N - 1) \qquad (*)$$

što odražava ključni odnos između vrijednosti na lokaciji P i vrijednosti na lokaciji N.

Tvrdnju $(*)$ nazivamo *invarijantom* while-petlje, jer ostaje istinita pod svakom iteracijom petlje.

Na posljetku, kada petlja završi, program se završava.

2.2 Tvrdnje parcijalne korektnosti

Neformalno govoreći, za sada nam se intuitivno čini da sustav dokaza možemo temeljiti na tvrdnjama obrasca

$$\{A\}\ c\ \{B\},$$

gdje su A i B tvrdnje poput onih koje smo već vidjeli u $Bexp$-u, a c je naredba. U sljedećem odjeljku ćemo formalno definirati kakve točno tvrdnje dopuštamo u IMP programima i da one pripadaju skupu $Assn$. Precizna interpretacija takve složene tvrdnje je sljedeća:

Za sva stanja σ koja zadovoljavaju A ako izvršenje c iz stanja σ završava u stanju σ' tada σ' zadovoljava B.

Tvrdnje oblika $\{A\}\ c\ \{B\}$ nazivaju se *tvrdnje parcijalne korektnosti* jer ne govore ništa o naredbi c, ako se ona ne uspijeva okončati.

Primjer 2.2. Promotrimo sljedeći program $c \equiv$ while $true$ do $skip$.
Izvšavanje c iz bilo kojeg početnog stanja se ne okončava. Prema gore navedenom tumačenju, valjana je sljedeća tvrdnja parcijalne korektnosti:

$$\{true\}\ c\ \{false\}$$

jednostavno zbog toga što se izvršavanje c neće završiti.

Općenito, zbog petlje c iz prethodnog primjera, vrijedi svaka tvrdnja o parcijalnoj korektnosti $\{A\}\ c\ \{B\}$.

Usporedimo to s drugim pojmom, onim o potpunoj korektnosti. Pišemo

$$[A]\ c\ [B]$$

što znači da će se izvršavanje c iz bilo kojeg stanja koje zadovoljava A završiti u stanju koje zadovoljava B.

Napomena 2.3. Primijetimo razliku između parcijalne korektnosti, koja zahtijeva da će rezultat izvršavanja, ako ga uspijemo izračunati i vratiti, biti točan i potpune korektnosti, koja dodatno zahtijeva i da se algoritam okonča.

2.3 Jezik tvrdnji $Assn$

Zapitajmo se kakve tvrdnje želimo izraziti u IMP programima. Obzirom da želimo koristiti Booleove izraze, sigurno ćemo uključiti sve tvrdnje iz $Bexp$, a jer želimo tvrdnje graditi pomoću kvantifikatora, "$\forall i...$" i "$\exists i...$", trebat će nam proširenja skupova $Bexp$ i $Aexp$ koja uključuju cjelobrojne varijable i preko kojih možemo kvantificirati. Pokažimo kako uvesti cjelobrojne varijable i kvantifikatore za određeni jezik tvrdnji $Assn$.

Proširenje skupa $Aexp$ označavamo s $Aexpv$ i njegovi elementi se formiraju sljedećim pravilima:

1. Ako je $n \in N$, tada za $a := n$ vrijedi $a \in Aexpv$;
2. Ako je $X \in Loc$, tada za $a := X$ vrijedi $a \in Aexpv$;
3. Ako je $i \in Intvar$, tada za $a := i$ vrijedi $a \in Aexpv$;
4. Ako su $a_1, a_2 \in Aexpv$, tada su $a_0 + a_1, a_0 - a_1, a_0 \cdot a_1 \in Aexpv$;
5. Izraz u IMP-u je iz skupa $Aexpv$ ako je nastao primjenom konačno mnogo koraka uvjeta 1, 2, 3 i 4.

Jedina razlika u odnosu na $Aexp$ je ta što smo u skup $Aexpv$ dodali cjelobrojne varijable iz skupa $Intvar$.

Booleove izraze proširujemo formacijskim pravilima za prethodno opisane općenitije aritmetičke izraze i uvodimo kvantifikatore te implikaciju. Prošireni skup Booleovih izraza nazivamo jezik tvrdnji $Assn$, a dan je sljedećim pravilima:

1. Za $B := true$ i $B := false$ vrijedi $B \in Assn$;
2. Ako su $a_0, a_1 \in Aexpv$, tada za $B := (a_0 = a_1)$ i $B := (a_0 \leq a_1)$ vrijedi $B \in Assn$;
3. Ako su $B_0, B_1 \in Assn, i \in Intvar$, tada su $B_0 \wedge B_1, B_0 \vee B_1, \neg B$, $B_0 \Rightarrow B_1, \forall i B, \exists i B \in Assn$;
4. Izraz u IMP-u je iz skupa $Assn$ ako je nastao primjenom konačno mnogo koraka uvjeta 1, 2 i 3.

2.3.1 Slobodne i vezane varijable

Kažemo da je nastup cjelobrojne varijable i u tvrdnji *vezan* ako se i javlja u dosegu kvantifikatora $\forall i$ ili $\exists i$. Ako nastup varijable nije vezan, kažemo da je *slobodan*. Uvedimo formalnu definiciju ovih pojmova.

Definicija 2.4. Za $a \in Aexpv$ strukturalnom indukcijom definiramo

$$F(a) = F(X) = \emptyset$$
$$F(i) = \{i\}$$
$$F(a_0 + a_1) = F(a_0 - a_1) = F(a_0 \cdot a_1) = F(a_0)F(a_1)$$

za sve $n \in N, X \in Loc, i \in Intvar$ i $a_0, a_1 \in Aexpv$. Iz ovoga je $F(a)$ skup koji nazivamo *skup slobodnih varijabli aritmetičkih izraza, proširenih s cjelobrojnim varijablama*.

Definicija 2.5. Za tvrdnju $A \in Assn$ strukturalnom indukcijom definiramo

$$F(true) = F(false) = \emptyset$$
$$F(a_0 = a_1) = F(a_0 \leq a_1) = F(a_0) \cup F(a_1)$$
$$F(A_0 \wedge A_1) = F(A_0 \vee A_1) = F(A_0 \Rightarrow A_1) = F(A_0) \cup F(A_1)$$
$$F(\neg A) = F(A)$$
$$F(\forall i A) = F(\exists i A) = F(A) \setminus \{i\}$$

za sve $a_0, a_1 \in Aexpv$, cjelobrojne varijable i i tvrdnje A_0, A_1, A. Iz ovoga je $F(A)$ skup koji nazivamo *skup slobodnih varijabli tvrdnje A*.

Za varijablu koja se pojavljuje u tvrdnji A i nije slobodna, kažemo da je *vezana*. Za tvrdnju bez slobodnih varijabli kažemo da je *zatvorena*.

Primjer 2.6. (a) U izrazu $\exists i\ k = i \cdot l$ je cjelobrojna varijabla i vezana, dok su k i l slobodne.

(b) U izrazu $(i \leq 52) \wedge (\exists i\ k = i + 25)$ je prva pojava od i slobodna, a druga vezana, dok je varijabla k slobodna.

2.3.2 Supstitucija

Zamislimo tvrdnju A kao

$$---i---i-$$

sa slobodnim pojavama cjelobrojne varijable $i \in Intvar$. Neka je a aritmetički izraz koji, zbog jednostavnosti, ne sadrži cjelobrojne varijable. Tada je

$$A[a/i] \equiv ---a---a-$$

rezultat zamjene i s a. Ako je a imala cjelobrojne varijable, onda bi bilo potrebno preimenovati neke vezane varijable od A da bismo izbjegli da varijable u a postanu vezane kvantifikatorima u A.

Definicija 2.7. Neka je i cjelobrojna varijabla i a aritmetički izraz bez cjelobrojnih varijabli. *Supstitucija u aritmetičkom izrazu* dana je strukturalnom indukcijom:

$$n[a/i] \equiv n \quad X[a/i] \equiv X$$

$$j[a/i] \equiv j \quad i[a/i] \equiv a$$

$$(a_0 + a_1)[a/i] \equiv (a_0[a/i] + a_1[a/i])$$

$$(a_0 - a_1)[a/i] \equiv (a_0[a/i] - a_1[a/i])$$

$$(a_0 \cdot a_1)[a/i] \equiv (a_0[a/i] \cdot a_1[a/i])$$

Definicija 2.8. Neka je i cjelobrojna varijabla i a aritmetički izraz bez cjelobrojnih varijabli. *Supstitucija* i s a u tvrdnjama dana je strukturalnom indukcijom:

$$true[a/i] \equiv true \quad false[a/i] \equiv false$$

$$(a_0 = a_1)[a/i] \equiv (a_0[a/i] = a_1[a/i]) \quad (a_0 \leq a_1)[a/i] \equiv (a_0[a/i] \leq a_1[a/i])$$

$$(A_0 \wedge A_1)[a/i] \equiv (A_0[a/i] \wedge A_1[a/i]) \quad (A_0 \vee A_1)[a/i] \equiv (A_0[a/i] \vee A_1[a/i])$$

$$(\neg A)[a/i] \equiv (\neg A[a/i]) \quad (A_0 \Rightarrow A_1)[a/i] \equiv (A_0[a/i] \Rightarrow A_1[a/i])$$

$$(\forall j\ A)[a/i] \equiv \forall j(A[a/i]) \quad (\forall i\ A)[a/i] \equiv \forall i\ A$$

$$(\exists j\ A)[a/i] \equiv \exists j(A[a/i]) \quad (\exists i\ A)[a/i] \equiv \exists i\ A$$

gdje su $a_0, a_1 \in Aexpv, A_0, A_1, A$ tvrdnje i j je cjelobrojna varijabla, $j \neq i$.

2.4 Semantika tvrdnji

Obzirom da smo skup aritmetičkih izraza proširili tako da sadrži i cjelobrojne varijable, prvo moramo interpretirati cjelobrojne varijable kao posebne cijele brojeve. To je uloga interpretacije.

Definicija 2.9. *Interpretacija* je funkcija koja pridružuje cijeli broj svakoj cjelobrojnoj varijabli, tj. funkcija $I : Intvar \to \mathbb{Z}$.

Sada možemo vrijednost izraza $a \in Aexpv$ u interpretaciji I i u stanju σ označiti s $[a]I_\sigma$.

Napomena 2.10. Prisjetimo se da elemente skupa $Aexpv$ aritmetičkih izraza s cjelobrojnim varijablama formiramo na sljedeći način:

$$a := n \mid X \mid i \mid a_0 + a_1 \mid a_0 - a_1 \mid a_0 \cdot a_1,$$

gdje n može imati vrijednosti iz skupa brojeva N, X može imati vrijednosti iz skupa lokacija Loc, i može imati vrijednosti iz skupa cjelobrojnih varijabli $Intvar$.

Definicija 2.11. *Semantika aritmetičkih izraza s cjelobrojnim varijablama* dana

je strukturalnom indukcijom:

$$[n]I_\sigma = n \quad [X]I_\sigma = \sigma(X) \quad [i]I_\sigma = I(i)$$

$$[a_0 + a_1]I_\sigma = [a_0]I_\sigma + [a_1]I_\sigma$$

$$[a_0 - a_1]I_\sigma = [a_0]I_\sigma - [a_1]I_\sigma$$

$$[a_0 \cdot a_1]I_\sigma = [a_0]I_\sigma \cdot [a_1]I_\sigma$$

Napomena 2.12. Koristimo oznaku $I[n/i]$ da bismo označili interpretaciju dobivenu iz interpretacije I zamjenom vrijednosti cjelobrojne varijable i u n, tj.

$$I[n/i](j) = \begin{cases} n, & \text{ako je } j \equiv i \\ I(j) & \text{inače} \end{cases}$$

S obzirom na danu interpretaciju, u nastavku izravno definiramo ona stanja koja zadovoljavaju neku tvrdnju. Prikladno je pritom proširiti skup stanja Σ na skup $\Sigma_\perp$ koji uključuje vrijednost $\perp$ povezanu s računanjem koje nije određeno. Dakle, $\Sigma_\perp =_{def} \Sigma \cup \{\perp\}$.

Definicija 2.13. Neka je $A \in Assn$. Relacijom $\sigma \models^I A$ označimo da *stanje σ zadovoljava A u interpretaciji I*. Strukturalnom indukcijom na tvrdnjama, za

interpretaciju I, definiramo za sva stanja $\sigma \in \Sigma$:

$$\sigma \models^I true$$

$$\sigma \models^I (a_0 = a_1) \text{ ako je } [a_0]I_\sigma = [a_1]I_\sigma$$

$$\sigma \models^I (a_0 \leq a_1) \text{ ako je } [a_0]I_\sigma \leq [a_1]I_\sigma$$

$$\sigma \models^I A \wedge B \text{ ako je } \sigma \models^I A \text{ i } \sigma \models^I B$$

$$\sigma \models^I A \vee B \text{ ako je } \sigma \models^I A \text{ ili } \sigma \models^I B$$

$$\sigma \models^I \neg A \text{ ako nije } \sigma \models^I A$$

$$\sigma \models^I A \Rightarrow B \text{ ako (nije } \sigma \models^I A \text{) ili } \sigma \models^I B$$

$$\sigma \models^I \forall i\ A \text{ ako } \sigma \models^{I[n/i]} A \text{ za sve } n \in N$$

$$\sigma \models^I \exists i\ A \text{ ako } \sigma \models^{I[n/i]} A \text{ za neki } n \in N$$

$$\bot \models^I A$$

Gore navedeno nam formalno govori što znači da je tvrdnja istinita u stanju, nakon što odlučimo protumačiti cjelobrojne varijable na određeni način s fiksnom interpretacijom I.

Semantika logičkih izraza pruža još jedan način da se kaže što znači da su određene vrste tvrdnji $true$ ili $false$ u nekom stanju.

Definicija 2.14. Neka je $b \in Bexp, \sigma \in \Sigma, I$ interpretacija.

$$[b]_\sigma = true \text{ ako i samo ako je } \sigma \models^I b$$
$$[b]_\sigma = false \text{ ako i samo ako je } \sigma \not\models^I b$$

Neka je I interpretacija. Često je prilikom uspostavljanja svojstava o tvrdnjama i tvrdnjama parcijalne korektnosti korisno razmotriti proširenje tvrdnje u odnosu na interpretaciju I, tj. na skup stanja u kojima je ta tvrdnja istinita.

Definicija 2.15. Neka je A zadana tvrdnja. *Proširenje tvrdnje A s obzirom na interpretaciju I* je skup

$$A^I = \{\sigma \in \Sigma_\perp \mid \sigma \models^I A\}.$$

2.4.1 Tvrdnje parcijalne korektnosti

Tvrdnja parcijalne korektnosti ima oblik

$$\{A\}\ c\ \{B\},$$

gdje su $A, B \in Assn$ i $c \in Com$.

Napomena 2.16. Primijetimo da tvrdnje parcijalne korektnosti nisu u skupu $Assn$.

Definicija 2.17. Neka je I interpretacija i $\sigma \in \Sigma$. Definiramo odnos ispunjavanja između stanja i tvrdnje parcijalne korektnosti, obzirom na I, s:

$$\sigma \models^I \{A\}\ c\ \{B\} \text{ ako i samo ako } (\sigma \models^I A \Rightarrow [c]_\sigma \models^I B)$$

Drugim riječima, stanje σ ispunjava tvrdnju parcijalne korektnosti $\{A\}\ c\ \{B\}$ obzirom na interpretaciju I ako i samo ako bilo koji uspješan izračun od c iz početnog stanja σ završava u stanju koje zadovoljava B.

Definicija 2.18. Neka je $c \in Com$ naredba. *Značenje $\mathcal{M}_I(c)$ u interpretaciji* I je parcijalna funkcija sa skupa stanja Σ u skup stanja Σ.

Napomena 2.19. Značenje $\mathcal{M}_I(c)$ smo definirali kao parcijalnu funkciju jer ne vrijedi svojstvo totalnosti da bi bila funkcija. Iz razloga što za ona početna stanja za koja se naredba c ne izvrši ili se njeno izvršavanje ne okončava, ne dolazi do promjene stanja, pa tada ne dolazimo u završno stanje.

Dakle, svakoj naredbi $c \in Com$ dodijelimo njezino značenje $\mathcal{M}_I(c)$ u interpretaciji I definirano s:

$\mathcal{M}_I(c)(\sigma) = \tau \in \Sigma$ takav da postoji izračun od c u interpretaciji I koji započinje

u stanju δ, a završava u τ, gdje je $\sigma \in \Sigma$ neko zadano početno stanje.

Primjer 2.20. Neka je dana interpretacija I i stanje $\sigma \in \Sigma$ te $[X]I_\sigma = \pi$. Neka je dana naredba $c \in Com$ koja glasi $X := X + X$. Tada je značenje naredbe c u interpretaciji I s početnim stanjem δ jednako stanju τ, tj. $\mathcal{M}_I(c)(\sigma) = \tau$, gdje za stanje τ vrijedi $[X]_\tau I = 2\pi$.

Vidimo da je lako definirati $\mathcal{M}_I(c)$ tako da se zabilježi predviđeno značenje programa. Sada konačno možemo definirati istinitost tvrdnje parcijalne korektnosti u interpretaciji I.

Definicija 2.21. Neka su $A, B \in Assn$ i naredba $c \in Com$. Za tvrdnju parcijalne korektnosti $\{A\}\ c\ \{B\}$ kažemo da je *istina u interpretaciji* I ako za sva stanja $\delta, \tau \in \Sigma$ vrijedi ako $\delta \models^I A$ i $\mathcal{M}_I(c)(\delta) = \tau$, onda $\tau \models^I B$.

U gornjoj definiciji s $\delta \models^I A$ označavamo da je tvrdnja A istina u stanju δ za interpretaciju I, a s $\mathcal{M}_I(c)(\delta) = \tau$ značenje u interpretaciji I naredbe c u stanju δ je upravo ono na stanju τ.

Definicija 2.22. Neka su $A, B \in Assn$ i naredba $c \in Com$. Kažemo da je tvrdnja parcijalne korektnosti $\{A\}\ c\ \{B\}$ valjana i pišemo

$$\models \{A\}\ c\ \{B\}$$

ako za sve interpretacije I i za sva stanja σ vrijedi

$$\sigma \models^I \{A\}\ c\ \{B\}.$$

Za tvrdnju $A \in Assn$ kažemo da je valjana i pišemo $\models A$ ako i samo ako za sve interpretacije I i sva stanja σ vrijedi $\sigma \models^I A$

Dakle, tvrdnja parcijalne korektnosti je valjana ako je istinita u svim interpretacijama I.

Primjer 2.23. Za sve A, B i c vrijedi:

$$\models \{false\}\ c\ \{B\} \quad \models \{A\}\ c\ \{true\}$$

jer $false$ nije istina ni u jednom stanju i $true$ je istina u svim stanjima.

Primjer 2.24. $\models \{Y \leq 3\}\ X = 2 \cdot Y + 1\ \{(X \leq 7) \wedge (Y \leq 3)\}$

Pokažimo da je ova tvdnja valjana.

Treba dokazati da vrijedi

$$\sigma \models^I \{Y \leq 3\}\ X = 2 \cdot Y + 1\ \{(X \leq 7) \wedge (Y \leq 3)\},$$

a to je ispunjeno ako i samo ako vrijedi

$$(\sigma \models^I Y \leq 3) \Rightarrow ([X = 2 \cdot Y + 1]_\sigma \models^I (X \leq 7) \wedge (Y \leq 3)).$$

Imamo dva slučaja:

1. Neka su $\sigma \in \Sigma$ i I takvi da vrijedi $\sigma \models^I Y \leq 3$.

   Tada u stanju σ za X vrijedi

   $$X = 2 \cdot Y + 1 \leq 2 \cdot 3 + 1 \leq 7.$$

Stoga je ispunjeno $(X \leq 7) \wedge (Y \leq 3)$. Dakle, $true \Rightarrow true$ je $true$.

2. Ako su $\sigma \in \Sigma$ i I takvi da ne vrijedi $\sigma \models^I Y \leq 3$, tj. ispunjeno je $\sigma \models^I Y > 3$, tada za X vrijedi

$$X = 2 \cdot Y + 1 > 2 \cdot 3 + 1 > 7$$

te $(X \leq 7) \wedge (Y \leq 3)$ nije ispunjeno. Dakle, $false \Rightarrow false$ je $true$.

Dakle, za sve interpretacije I i sva stanja σ vrijedi

$$\sigma \models^I \{Y \leq 3\}\ X = 2 \cdot Y + 1\ \{(X \leq 7) \wedge (Y \leq 3)\},$$

a to je ispunjeno ako i samo ako vrijedi

$$(\sigma \models^I Y \leq 3) \Rightarrow ([X = 2 \cdot Y + 1]_\sigma \models^I (X \leq 7) \wedge (Y \leq 3)).$$

Stoga, polazna tvrdnja parcijalne korektnosti je valjana.

Poglavlje 3

Hoareova pravila za *while* programe

U ovom poglavlju predstavljamo pravila dokazivanja koja generiraju valjane tvrdnje parcijalne korektnosti. Pravila dokazivanja su usmjerena prema sintaksi, reduciraju dokazivanje tvrdnje parcijalne korektnosti složene naredbe na dokazivanje tvrdnje parcijalne korektnosti njezine neposredne podnaredbe. Pravila dokazivanja se često nazivaju Hoareova pravila, a sustav dokaza koji se sastoji od kolekcije pravila Hoareova logika.

3.1 Sustav dokaza $\mathcal{H}$

Neka su $A, A', B, B', C \in Assn,\ a \in Aexpv,\ X \in Loc,\ c, c_0, c_1 \in Com,\ b \in Bexp.$

Pravilo za skip

$$\{A\}\ skip\ \{A\}$$

Pravilo za dodjeljivanje

$$\{B[a/X]\}\ X := a\ \{B\}$$

Pravilo za nizanje

$$\frac{\{A\}\ c_0\ \{C\} \quad \{C\}\ c_1\ \{B\}}{\{A\}\ c_0; c_1\ \{B\}}$$

Pravilo za kondicional

$$\frac{\{A \wedge b\}\ c_0\ \{B\} \quad \{A \wedge \neg b\}\ c_1\ \{B\}}{\{A\}\ \text{if } b \text{ then } c_0 \text{ else } c_1\ \{B\}}$$

Pravilo za while petlju

$$\frac{\{A \wedge b\}\ c\ \{A\}}{\{A\}\ \text{while } b \text{ do } c\ \{A \wedge \neg b\}}$$

Pravilo posljedice

$$\frac{\models (A \Rightarrow A') \quad \{A'\}\ c\ \{B'\} \quad \models (B' \Rightarrow B)}{\{A\}\ c\ \{B\}}$$

Rezultirajući sustav označimo sa $\mathcal{H}$.

Napomena 3.1. *Teoremi* su matematičke tvrdnje čije se istinitosti utvrđuju dokazom. *Dokaz* teorema je konačan niz tvrdnji, pri čemu je svaka tvrdnja izvedena logično (tj. korištenjem pravila zaključivanja) iz prethodnih tvrdnji ili korištenjem teorema čija je istinitost već utvrđena. Zadnja tvrdnja u nizu je tvrdnja teorema. Dokaz je dakle deduktivni postupak kojim počevši od hipoteza dolazimo do zaključaka. Hoareova logika je primjer formaliziranja deduktivnog sustava, koji sadrži pravila dokazivanja.

Napomena 3.2. Hoareova pravila se smatraju *sustavom dokaza*, izvodi se nazivaju *dokazima*, a svaki zaključak (konkluzija) izvoda *teorem*. Pišemo $\vdash \{A\}\ c\ \{B\}$ za $A, B \in Assn, c \in Com$ kada je tvrdnja parcijalne korektnosti {A} c {B} teorem, tj. kada je njenu istinitost moguće utvrditi dokazom, koristeći Hoareova pravila.

Primjer 3.3. Iz prethodnog primjera znamo da

$$\models \{Y \leq 3\}\ X = 2 \cdot Y + 1\ \{(X \leq 7) \wedge (Y \leq 3)\}$$

Očito $\models \{Y \leq 1\}\ X = 2 \cdot Y + 1\ \{(X \leq 7) \wedge (Y \leq 3)\}$.
Skup stanja koja ispunjavaju $Y \leq 1$ je podskup skupa stanja koja zadovoljavaju $Y \leq 3$. Stoga izračun započet u stanju gdje je npr. $Y = 0 \leq 1$ ispunjava $Y \leq 3$. Slično tome, skup stanja koja ispunjavaju $X \leq 10$ su nadskup skupa stanja koja zadovoljavaju $X \leq 7$. Znamo da izračun završava s vrijednosti od X takvom da je $X \leq 7$ i ta vrijednost je također manja ili jednaka 10.

Pravila je lako razumjeti, s mogućom iznimkom pravila dodjeljivanja i *while* petlje. Ako je tvrdnja istinita u stanju prije izvođenja *skip* koje označava preskok, to je sigurno i nakon toga, jer je stanje nakon preskoka nepromijenjeno.
Da bismo se uvjerili da je pravilo dodjeljivanja ispravno, može se pokušati provjeriti za određenu tvrdnju poput $X = 3$ za jednostavno dodjeljivanje kao što je $X := X + 5$.
Pravilo za nizanje (kompozicija) izražava da ako su $\{A\}\ c_0\ \{C\}$ i $\{C\}\ c_1\ \{B\}$ valjani, tada je i $\{A\}\ c_0; c_1\ \{B\}$: ako je uspješno izvršenje c_0 iz stanja koje zadovoljava A završeno u stanju koje zadovoljava C i ako je uspješno izvršenje c_1 iz stanja koje zadovoljava C završeno u stanju koje zadovoljava B, tada bilo koje uspješno izvršenje c_0 nakon kojeg slijedi c_1 iz stanja koje zadovoljava A završeno u stanju koje zadovoljava B.

Dvije premise u pravilu za kondicional odnose se na dvije grane kondicionala. U pravilu za *while* petlju: while b do c, tvrdnja A se naziva invarijantnom, jer pretpostavka da je $\{A \wedge b\}\ c\ \{A\}$ valjana kaže da je tvrdnja A sačuvana potpunim izvršenjem tijela petlje, a kod *while* petlje takva izvršavanja se odvijaju samo iz stanja koja zadovoljavaju b. Iz stanja koje zadovoljava A, ili se izvršavanje *while* petlje nikad ne okonča ili se odvije konačan broj izvršavanja tijela petlje, a svako započne u stanju koje zadovoljava b. U posljednjem slučaju, kako je A invarijanta, finalno stanje zadovoljava A i također $\neg b$ na izlasku iz petlje. Pravilo posljedice posjeduje valjane implikacije. Svaka instanca pravila posljedice ima premise, uključujući i $\models (A \Rightarrow A')$ i $\models (B' \Rightarrow B)$ pa stvaranje instanci pravila posljedice s ciljem primjene u dokazu ovisi prvo o dokazivanju valjanosti tvrdnji $(A \Rightarrow A')$ i $(B' \Rightarrow B)$. Općenito, ovo može biti vrlo težak zadatak - takve implikacije mogu izraziti kompliciranе činjenice o aritmetici. Srećom, budući da programi često ne uključuju duboke matematičke činjenice demonstracija ovih valjanosti često se može obaviti s osnovnom matematikom.

3.2 Utemeljenost $\mathcal{H}$

Svako pravilo dokazivanja bi trebalo očuvati valjanost u smislu da ako su pretpostavke u premisi pravila valjane, onda je valjana i konkluzija. Kada ovo vrijedi, pravilo dokazivanja nazivamo utemeljenim. Uvedimo formalne definicije utemeljenog pravila dokazivanja i utemeljenog sustava dokaza.

Definicija 3.4. Pravilo dokazivanja nazivamo *utemeljenim* ako za sve interpretacije I ono čuva istinu u interpretaciji I za tvrdnje iz $Assn$ (i u slučaju pravila posljedice čuva valjanost tvrdnje parcijalne korektnosti).

Definicija 3.5. Ako je svako pravilo sustava dokaza utemeljeno, tada kažemo

da je sustav dokaza *utemeljen*.

Lako se pokaže da su aksiomi od $\mathcal{H}$ valjani i da su pravila dokazivanja od $\mathcal{H}$ utemeljena.
Ova činjenica implicira (indukcijom po duljini dokaza) sljedeći teorem koji kaže da je sustav dokaza $\mathcal{H}$ utemeljen.

Teorem 3.6. Za svaku interpretaciju I, skup tvrdnji $Assn$ i tvrdnju parcijalne korektnosti $\{A\}\ c\ \{B\}$ vrijedi sljedeće: ako su sve tvrdnje iz $Assn$ istinite u interpretaciji I i $Assn \vdash^{\mathcal{H}} \{A\}\ c\ \{B\}$, tada je $\{A\}\ c\ \{B\}$ istinita u interpretaciji I.

Drugim riječima, ako je $Tr_I \vdash^{\mathcal{H}} \{A\}\ c\ \{B\}$ onda $\models^I \{A\}\ c\ \{B\}$, gdje Tr_I označava skup svih tvrdnji iz $Assn$ koje su istina u interpretaciji I.

3.3 Problem potpunosti $\mathcal{H}$

Željeli bismo da sustav dokaza bude "dovoljno jak" da se sve valjane tvrdnje parcijalne korektnosti mogu dobiti kao teoremi tj. da sustav dokaza bude *potpun* u tom smislu. Sam sustav $\mathcal{H}$ je nepotpun.

Primjer 3.7. Tvrdnja parcijalne korektnosti $\{A[a/X]\}\ X := a\ \{true\}$ je istina u svakoj interpretaciji I, a u $\mathcal{H}$ je ipak nedokaziva. Naime, ne postoji način da se koristeći pravila iz $\mathcal{H}$ dokaže $\models (A \Rightarrow true)$, pa vidimo da je $\mathcal{H}$ nepotpun.

Naravno da bismo željeli da sustav dokaza bude učinkovit u smislu da je rutinska stvar provjeriti da li nešto što je predloženo kao instanca pravila uistinu to i jest. To bi trebalo biti rutina u smislu da postoji metoda izračuna u obliku programa koja pri ulazu stvarne instance pravila vraća potvrdan rezultat. Na ulazima koji nisu instace pravila vrijedi sljedeće: metoda izračuna ne daje potvrdan rezultat i nije nužno da se njeno izvršavanje zaustavi.

Ne možemo tvrditi da je sustav dokaza $\mathcal{H}$ učinkovit jer nemamo računsku metodu za provjeru instanci pravila posljedice. Imati takvo što ovisi o mogućnosti izračunavanja kako bi se provjerilo da li su tvrdnje *Assn* valjane. Veliki austrijski logičar Kurt Gödel pokazao je da je logički nemoguće imati učinkovit dokazni sustav u kojem se mogu dokazati valjane tvrdnje iz *Assn*.

Teorem 3.8. Gödelov teorem nepotpunosti (1931)
Ne postoji učinkoviti sustav dokaza za *Assn* takav da se teoremi podudaraju s valjanim tvrdnjama iz *Assn*.

Gödelov teorem nepotpunosti implicira da ne postoji potpun sustav dokaza za precizno utvrđivanje valjanih tvrdnji. Hoareova pravila nasljeđuju ovu nepotpunost. Međutim, razdvajanjem nepotpunosti jezika tvrdnji *Assn* od nepotpunosti zbog neadekvatnosti u aksiomima i pravilima za konstrukcije programskog jezika, možemo dobiti relativnu potpunost u smislu Cook-a.

3.3.1 Potpunost $\mathcal{H}$ u Cook-ovom smislu

Iako znamo da sustav dokaza $\mathcal{H}$ nije učinkovit zbog prirode pravila posljedice, ostajemo pri njemu. Određivanje da imamo instancu pravila posljedice ovisi o tome da su neke tvrdnje valjane. Ipak, možemo promatrati potpunost ovog sustava. Da je potpun, utvrdio je S.Cook. Definirajmo

$$post_I(A, c) = \{\tau \in \Sigma : (\exists \delta \in \Sigma)\ \delta \models^I A \wedge \mathcal{M}_I(c)(\delta) = \tau\}$$
$$pre_I(c, B) = \{\delta \in \Sigma : (\forall \tau \in \Sigma)\ \mathcal{M}_I(c)(\delta) = \tau \rightarrow \tau \models^I B\}$$

Uočimo da su ovi skupovi karakterizirani sljedećim ekvivalencijama:
$\models^I \{A\}\ c\ \{B\}$ ako i samo ako $\{\delta : \delta \models^I A\} \subset pre_I(c, B)$ ako i samo ako $post_I(A, c) \subset \{\delta : \delta \models^I B\}$.

Ako je tvrdnja parcijalne korektnosti valjana tada postoji njen dokaz koristeći Hoareova pravila, to jest za bilo koju tvrdnju parcijalne korektnosti $\{A\}\ c\ \{B\}$

$$\models \{A\}\ c\ \{B\} \text{ implicira } \vdash \{A\}\ c\ \{B\}$$

premda se činjenica da je to dokaz može osloniti na to da su određene tvrdnje u $Assn$-u valjane.

Definicija 3.9. Za jezik IMP kažemo da je *izraziv u odnosu na interpretaciju I i skup naredbi Com* ako za sve tvrdnje $A \in Assn$ i naredbe $c \in Com$ vrijedi da postoji tvrdnja $B \in Assn$ takva da definira $post_I(A, c)$ u IMP-u. Ako je I takva da je IMP izraziv u odnosu na I i Com, pišemo $I \in Exp(\text{IMP}, Com)$.

Definicija 3.10. Sustav dokaza $\mathcal{H}$ za $Assn$ je *potpun u Cook-ovom smislu* ako za svaku interpretaciju $I \in Exp(\text{IMP}, Com)$ i za svaku tvrdnju parcijalne korektnosti $\{A\}\ c\ \{B\}$ vrijedi: ako $\models^I \{A\}\ c\ \{B\}$, onda $Tr_I \vdash^{\mathcal{H}} \{A\}\ c\ \{B\}$.

Neka ja $I \in Exp(\text{IMP}, Com)$.

Ako $\models^I \{A\}\ X := a\ \{B\}$, tada očito $\models^I (A \Rightarrow B[a/X])$, pa po pravilima dodjeljivanja i posljedice vrijedi $Tr_I \vdash^{\mathcal{H}} \{A\}\ X := a\ \{B\}$.

Ako $\models^I \{A\}\ c_0; c_1\ \{B\}$, tada očito $\models^I \{A\}\ c_0\ \{C\}$ i $\models^I \{C\}\ c_1\ \{B\}$, gdje C definira $post_I(C, c_O)$, pa indukcijom i pravilom posljedice slijedi $Tr_I \vdash^{\mathcal{H}} \{A\}\ c_0; c_1\ \{B\}$.

Slučajevi za *skip* i za if b then c_0 else c_1 slijede izravno.

Ako $\models^I \{A\}$ while b do $c\ \{B\}$, onda moramo pronaći invarijantu petlje C takvu da $\models^I \{C \wedge b\}\ c\ \{C\}$, $\models^I (A \Rightarrow C)$ i $\models^I (C \wedge b) \Rightarrow \neg B$. Potom, indukcijom slijedi $Tr_I \vdash^{\mathcal{H}} \{A\}$ while b do $c\ \{B\}$.

Poglavlje 4

Procedure bez parametara

U računalnom programiranju procedura je logička cjelina koja se sastoji od slijeda programskih instrukcija koje izvršavaju određeni zadatak. Procedura se tada može koristiti tj. pozivati u programima gdje god treba izvršiti taj zadatak. U našem kontekstu, procedura je slijed naredbi iz skupa Com, a dodatno uvodimo i naredbu *poziv procedure* i time dolazimo do pojma rekurzije. Tijelo procedure je naziv za slijed narebi koje grade tu proceduru. Kažemo da je procedura rekurzivna, ako se unutar tijela procedure nalazi naredba pozivanja te iste procedure, koja će se barem jednom izvršiti. Zbog jasnoće, izdvojili smo probleme koji se tiču procedura od problema djelokruga i mehanizama parametara. Dalje se raspravlja o bezparametarskim procedurama, a obrada parametara se nalazi u 7. poglavlju. Radi jednostavnosti, ograničavamo pažnju na slučaj deklaracije jedne procedure. Svi se rezultati ovog poglavlja mogu generalizirati na slučajeve deklaracije više od jedne procedure.

4.1 Nerekurzivne procedure

Prvo ćemo razmotriti jednostavniji slučaj procedura, a to su nerekurzivne procedure. Pretpostavimo da imamo deklariranu proceduru $p \Longleftarrow c_0$, gdje je $c_0 \in Com$ tijelo procedure p i proširimo skup naredbi Com dozvoljavajući naredbama da sadrže pozive procedure p. Nazovimo ovu proširenu klasu naredbi Com_1.

Svaki poziv procedure p odnosi se na deklaraciju $p \Longleftarrow c_0$. Zahtjev da $c_o \in Com$ povlači da procedura p nije rekurzivna.
Da bismo se bavili pozivima procedura u dokazima korektnosti, dopunjujemo sustav dokaza sljedećim pravilom.

Pravilo pozivanja procedure

$$\frac{\{A\}\ c_0\ \{B\}}{\{A\}\ p\ \{B\}}$$

Da bismo razmatrali problem utemeljenosti i potpunosti rezultirajućeg sustava, prvo moramo proširiti funkciju značenja $\mathcal{M}_I$ na naredbe iz Com_1.

Za $c \in Com_1$ neka je $c[c_0/p]$ program koji je rezultat zamjene svih pojava procedure p u c sa c_0. Za $c \in Com_1 \setminus Com$ definiramo da je $\mathcal{M}_I(c)$ jednak $\mathcal{M}_I(c[c_0/p])$.

Stoga $\mathcal{M}_I(p) = \mathcal{M}_I(c_0)$, što povlači da je pravilo poziva procedure utemeljeno. Iz činjenice da su pravila iz $\mathcal{H}$ utemeljena u slučaju Com i definicije od $\mathcal{M}_I(c)$ za $c \in Com_1$ trivijalno slijedi utemeljenost pravila iz $\mathcal{H}$ u slučaju Com_1.

4.2 Pravilo rekurzije

Kada je deklarirana procedura $p \Longleftarrow c_0$ rekurzivna, tj. kada je $c_0 \in Com_1 \setminus Com$, gore navedeni sustav je i dalje utemeljen, ali očito nepotpun. Pokušamo li

dokazati $\{A\}\ p\ \{B\}$ do rezultata nije moguće doći u konačno mnogo koraka već se proces dokazivanja ne okončava. Da bismo prevladali ove poteškoće, potrebno je usvojiti sljedeće pravilo.

Pravilo rekurzije

$$\frac{\{A\}\ p\ \{B\} \vdash \{A\}\ c_0\ \{B\}}{\{A\}\ p\ \{B\}}$$

Obrazloženje ovog pravila je sljedeće: zaključiti $\{A\}\ p\ \{B\}$ iz činjenice da $\{A\}\ c_0\ \{B\}$ može se dokazati (koristeći preostala pravila) iz pretpostavke $\{A\}\ p\ \{B\}$.

Primjer 4.1. Primjer dokaza koristeći pravilo rekurzije

Promotrimo deklaraciju procedure $p \Longleftarrow c_0$ za

$c_0 \equiv \text{if } X = 0 \text{ then } Y := 1 \text{ else } X := X - 1; p; X := X + 1; Y := Y \cdot X$

Sada dokazujemo $\{X \geq 0\}\ p\ \{Y = X!\}$ u sustavu $\mathcal{H}$ nadopunjenom s pravilom rekurzije.

Po pravilu rekurzije, dovoljno je pokazati

$$\{X \geq 0\}\ p\ \{Y = X!\} \vdash^{\mathcal{H}} \{X \geq 0\}\ c_0\ \{Y = X!\}.$$

Pretpostavimo

$$\{X \geq 0\}\ p\ \{Y = X!\}. \tag{4.1}$$

Po pravilu za dodjeljivanje

$$\{Y \cdot X = X!\}\ Y := Y \cdot X\ \{Y = X!\}$$

i

$$\{Y \cdot (X + 1) = (X + 1)!\}\ X := X + 1\ \{Y \cdot X = X!\}$$

pa, po pravilu nizanja

$$\{Y \cdot (X + 1) = (X + 1)!\}\ X := X + 1; Y := Y \cdot X\ \{Y = X!\}. \tag{4.2}$$

Kako je implikacija

$$\models [Y = X! \Rightarrow Y \cdot (X + 1) = (X + 1)!] \tag{4.3}$$

istina, to po pravilu posljedice i (4.1)

$$\{X \geq 0\}\ p\ \{Y \cdot (X + 1) = (X + 1)!\}. \tag{4.4}$$

(4.2) i (4.4) impliciraju, po pravilu za nizanje

$$\{X \geq 0\}\ p; X := X + 1; Y := Y \cdot X\ \{Y = X!\}. \tag{4.5}$$

S druge strane, kako vrijedi

$$\models (X \geq 0 \wedge X \neq 0 \Rightarrow X - 1 \geq 0) \tag{4.6}$$

i po pravilu za dodjeljivanje

$$\{X - 1 \geq 0\}\ X := X - 1\ \{X \geq 0\},$$

koristeći pravilo posljedice dobijamo

$$\{X \geq 0 \wedge X \neq 0\}\ X := X - 1\ \{X \geq 0\}. \tag{4.7}$$

Po pravilu za nizanje, iz (4.5) i (4.7) pravilom za nizanje dobijamo

$$\{X \geq 0 \wedge X \neq 0\}\ X := X - 1; p; X := X + 1; Y := Y \cdot X\ \{Y = X!\}. \tag{4.8}$$

Kako je

$$\models (X \geq 0 \wedge X = 0 \Rightarrow 1 = X!) \tag{4.9}$$

istina i po pravilu dodjeljivanja,

$$\{1 = X!\}\ Y := 1\ \{Y = X!\},$$

koristeći pravilo posljedice dobijamo

$$\{X \geq 0 \wedge X = 0\}\ Y := 1\ \{Y = X!\}. \tag{4.10}$$

(4.8) i (4.10) konačno impliciraju po pravilu za kondicional

$$\{X \geq 0\}\ c_0\ \{Y = X!\},$$

što je i trebalo dokazati.

Naravno, strogo govoreći, samo smo dokazali da (4.3),(4.6),(4.9) $\vdash^{\mathcal{H}+Pravilo\ rekurzije}$ $\{X \geq 0\}\ p\ \{Y = X!\}$.

4.3 Sustav dokaza $\mathcal{G}$

Sustav $\mathcal{H}$ nadopunjen s pravilom rekurzije je nepotpun. Stoga ovaj sustav nadopunjavamo sljedećim aksiomom i pravilima dokazivanja, što vodi k potpunom sustavu dokaza.

Aksiom invarijance

$$\{A\}\ p\ \{A\}, \quad gdje\ je \quad F(A) \cap V(c_0) = \emptyset.$$

Pravilo supstitucije I

$$\frac{\{A\}\ c\ \{B\}}{\{A[\tilde{Y}/\tilde{Z}]\}\ p\ \{B[\tilde{Y}/\tilde{Z}]\}}, \quad gdje\ je \quad \tilde{Z} \cap V(c_0) = \emptyset \quad i \quad \tilde{Y} \cap V(c_0) = \emptyset.$$

Pravilo supstitucije II

$$\frac{\{A\}\ p\ \{B\}}{\{A[\tilde{Y}/\tilde{Z}]\}\ p\ \{B\}}, \quad gdje\ je \quad \tilde{Z} \cap V(c_0, B) = \emptyset.$$

Pravilo konjunkcije

$$\frac{\{A\}\ p\ \{B\}, \{A'\}\ p\ \{C\}}{\{A \wedge A'\}\ p\ \{B \wedge C\}.}$$

Ovdje $\tilde{Y}$ i $\tilde{Z}$ označavaju nizove varijabli jezika IMP, $A[\tilde{Y}/\tilde{Z}]$ označava simultanu supstituciju varijabli iz $\tilde{Y}$ varijablama iz $\tilde{Z}$ u A. $V(c_0)$ označava skup svih varijabli koje se pojavljuju u c_0 i $F(A)$ označava skup svih slobodnih varijabli od A.

Rezultirajući sustav označimo s $\mathcal{G}$.

Primjer 4.2. Primjer dokaza u $\mathcal{G}$

Da bismo vidjeli kako se koriste dodana pravila u $\mathcal{G}$ u stvarnim dokazima, sada želimo dokazati formulu $\{X = Z\}\ p\ \{X = Z\}$, gdje je p procedura faktorijel iz prethodnog primjera. Da bismo dokazali ovu formulu, dovoljno je ustanoviti premisu $\{X = Z\}\ p\ \{X = Z\} \vdash \{X = Z\}\ c_0\ \{X = Z\}$ pravila rekurzije.
Pretpostavimo

$$\{X = Z\}\ p\ \{X = Z\}.$$

Po pravilu supstitucije I,

$$\{X = U\}\ p\ \{X = U\},$$

i po aksiomu invarijance

$$\{U = Z - 1\}\ p\ \{U = Z - 1\},$$

pa po pravilu konjunkcije i pravilu posljedice imamo

$$\{X = Z - 1\}\ p\ \{X = Z - 1\}. \tag{4.11}$$

Sada primjenom pravila za dodjeljivanje, kompoziciju i posljedice, iz (4.11) dobijamo

$$\{X = Z\}\ X := X - 1; p; X := X + 1; Y := Y \cdot X\ \{X = Z\}.$$

Po pravilu posljedice možemo sjediniti predtvrdnju s $X \neq 0$.
Također vrijedi i

$$\{X = Z \wedge X = 0\}\ Y := 1\ \{X = Z\}.$$

Po pravilu za kondicional sada dobijamo $\{X = Z\}\ c_0\ \{X = Z\}$, pa smo ustanovili željenu premisu.

Prethodni dokaz nije koristio pravilo supstitucije II. Međutim, to je pravilo potrebno, što možemo pogledati u sljedećem primjeru.

Primjer 4.3. Primjer dokaza u $\mathcal{G}$ u kojem se koristi pravilo rekurzije II

Pokažimo da vrijedi $\{X \geq 0\}\ p\ \{Y \geq 1\}$.

U dokazu koristimo već dokazane formule

$$\{X \geq 0\}\ p\ \{Y = X!\} \text{ i } \{X = Z\}\ p\ \{X = Z\}$$

i sljedeću instancu aksioma invarijance

$$\{Z \geq 0\}\ p\ \{Z \geq 0\},$$

za dobivanje

$$\{X \geq 0 \wedge X = Z \wedge Z \geq 0\}\ p\ \{Y = X! \wedge X = Z \wedge Z \geq 0\}$$

po pravilu konjunkcije.

Koristeći pravilo posljedice sada dobijamo

$$\{X = Z \wedge X \geq 0\}\ p\ \{Y \geq 1\}.$$

Konačno, po pravilu supstitucije II imamo

$$\{X = X \wedge X \geq 0\}\ p\ \{Y \geq 1\},$$

pa vrijedi

$$\{X \geq 0\}\ p\ \{Y \geq 1\}$$

po pravilu posljedice.

4.4 Utemeljenost $\mathcal{G}$

Prije nego što se usredotočimo na pitanje utemeljenosti i potpunosti $\mathcal{G}$ potrebno je definirati funkciju značenja $\mathcal{M}_I$ na programima iz $Com_1 \setminus Com$. To radimo na način koji pojednostavljuje naša razmatranja u vezi sa utemeljenosti $\mathcal{G}$.

Neka je dana procedura p. Označimo s Ω naredbu iz Com koja se nikad ne zaustavlja. Definirajmo $c_0^{(n)} \in Com$ indukcijom po n:

$$c_0^{(0)} \equiv \Omega$$

$$c_0^{(n+1)} \equiv c_0 \, [\, c_0^{(n)}/p \,]$$

Izravan dokaz indukcijom po strukturi od c pokazuje da za sve $c_1, c_2 \in Com$ i $c \in Com_1$ vrijedi: ako $\mathcal{M}_I(c_1) \subset \mathcal{M}_I(c_2)$, onda $\mathcal{M}_I(\, c \, [\, c_1/p \,] \,) \subset \mathcal{M}_I(\, c \, [\, c_2/p \,] \,)$. Ovo implicira (indukcijom po n) da za sve $\mathcal{M}_I(c_0^{(n)}) \subset \mathcal{M}_I(c_0^{(n+1)})$.
Dakle, za sve $c \in Com_1$, $\mathcal{M}_I(\, c \, [\, c_0^{(n)}/p \,] \,) \subset \mathcal{M}_I(\, c \, [\, c_0^{(n+1)}/p \,] \,)$.
Za $c \in Com_1$ sada definiramo $\mathcal{M}_I(c)$ stavljajući

$$\mathcal{M}_I(c) = \bigcup_{n=0}^{\infty} \mathcal{M}_I(\, c \, [c_0^{(n)}/p \,] \,).$$

Posebno,

$$\mathcal{M}_I(p) = \bigcup_{n=0}^{\infty} \mathcal{M}_I(c_0^{(n)}).$$

Po gore navedenom $\mathcal{M}_I(c)$ je parcijalna funkcija.

Sada želimo dokazati da je sustav dokaza $\mathcal{G}$ utemeljen, odnosno da za svaku interpretaciju I i formulu parcijalne korektnosti $\{A\}\ c\ \{B\}$ vrijedi: ako $Tr_I \vdash_{\mathcal{G}} \{A\}\ c\ \{B\}$, onda $\models_I \{A\}\ c\ \{B\}$.

Napomena 4.4. Prisjetimo se da smo s $\mathcal{H}$ označili sustav Hoareovih pravila koji sadrži pravilo za skip, dodjeljivanje, nizanje, kondicional, while petlju i pravilo posljedice.

Napomena 4.5. Činjenica da $\mathcal{G}$ nije uobičajeni sustav dokaza u smislu logike prvog reda prisiljava nas da budemo pažljivi pri dokazivanju njegove utemeljenosti. Primijetimo da npr. nije jasno u kojoj mjeri možemo upotrijebiti činjenicu da su pravila iz $\mathcal{H}$ utemeljena u slučaju deklariranja nerekurzivne procedure i u kojem smislu se pravilo rekurzije mora dokazati utemeljenim.

Da bismo riješili ove probleme, prvo pretvaramo sustav $\mathcal{G}$ u sustav dokaza $\mathcal{K}$ koji koristi uobičajeni pojam dokaza.

Formule iz $\mathcal{K}$ su implikacije $\Phi \to \Psi$ koje nazivamo *fraze korektnosti*, gdje su Φ i Ψ konačni skupovi formula parcijalne korektnosti. Ako je Φ prazan skup, pišemo Ψ umjesto $\Phi \to \Psi$.

Za svaki aksiom φ_0 i pravilo dokazivanja

$$\frac{\varphi_1, ..., \varphi_n}{\varphi_{n+1}}$$

iz $\mathcal{G}$, u K usvajamo aksiom $\Phi \to \varphi_0$ i pravilo dokazivanja

$$\frac{\Phi \to \varphi_1, ..., \varphi_n}{\Phi \to \varphi_{n+1}}.$$

Također, imamo i pravilo

$$\frac{\{A\}\ p\ \{B\} \to \{A\}\ c_0\ \{B\}}{\{A\}\ p\ \{B\}},$$

koje odgovara pravilu rekurzije, *pravilo sakupljanja*

$$\frac{\Phi \to \Psi_1, ..., \Phi \to \Psi_n}{\Phi \to \Psi_1, ..., \Psi_n}$$

i *selekcijski aksiom* $\Phi \to \varphi$, gdje je $\varphi \in \Phi$.

U nastavku koristimo oznaku $\langle p \Longleftarrow c \mid \varphi \rangle$ umjesto φ da naznačimo da se svaki poziv procedure p u φ odnosi na deklaraciju procedure $p \Longleftarrow c$, gdje je $\varphi = \{A\}\ c\ \{B\}$.

Definicija 4.6. Neka je I interpretacija.

1. Implikacija $\langle p \Longleftarrow c_0 \mid \Phi \to \Psi \rangle$ se naziva I *– dobrom* ako za svaki n, $\langle p \Longleftarrow c_0^{(n)} \mid \Phi \to \Psi \rangle$ je istina u interpretaciji I.

2. Za deklaraciju nerekurzivne procedure $p \Longleftarrow c$:

(a) $\langle p \Longleftarrow c \mid \Phi \to \Psi \rangle$ je istina u interpretaciji I ako istinitost u interpretaciji I od $\langle p \Longleftarrow c \mid \Phi \rangle$ povlači istinitost u interpretaciji I od $\langle p \Longleftarrow c \mid \Psi \rangle$

(b) $\langle p \Longleftarrow c \mid \Phi \rangle$ je istina u interpretaciji I ako za sve $\varphi \in \Phi$, $\langle p \Longleftarrow c \mid \varphi \rangle$ je istina u interpretaciji I.

Definicija 4.7.

1. Fraza korektnosti se naziva *dobrom* ako je I-dobra za sve intepretacije I.
2. Pravilo dokazivanja iz $\mathcal{K}$ se naziva *dobrim* ako za sve interpretacije I čuva I-dobrotu fraza korektnosti.
3. Sustav dokaza $\mathcal{K}$ se naziva *dobrim* ako su svi njegovi aksiomi i pravila dokazivanja dobri.

Primijetimo da je za svaki skup T tvrdnji iz $Assn$, $T \cup \{\varphi\} \vdash_{\mathcal{G}} \Psi$ ako i samo ako $T \vdash_{\mathcal{K}} \varphi \to \Psi$. Dokaz se provodi indukciom po duljini dokaza. Dakle, posebno, za svaki skup T tvrdnji iz $Assn$, $T \vdash_{\mathcal{G}} \varphi$ ako i samo ako $T \vdash_{\mathcal{K}} \varphi$. Ovo zajedno s opažanjem $\langle p \Longleftarrow c_0 \mid \varphi \rangle$ je istina u interpretaciji I ako i samo ako je I-dobra, implicira sljedeću tvrdnju:

Tvrdnja 4.8. Ako je sustav $\mathcal{K}$ dokazivanja dobar, onda je sustav dokaza $\mathcal{G}$ utemeljen.

Tvrdnja 4.9.

1. Ako je za svaki n $\langle p \Longleftarrow c_0^{(n)} \mid \varphi \rangle$ valjana, onda je $\langle p \Longleftarrow c_0 \mid \Phi \to \varphi \rangle$ dobro.
2. Ako je za svaki n pravilo dokazivanja

$$\frac{\langle p \Longleftarrow c_0^{(n)} \mid \Psi_1 \rangle}{\langle p \Longleftarrow c_0^{(n)} \mid \Psi_2 \rangle}$$

utemeljeno, onda je pravilo dokazivanja

$$\frac{\langle p \Longleftarrow c_0 \mid \Psi_1 \rangle}{\langle p \Longleftarrow c_0 \mid \Psi_2 \rangle}$$

dobro.

Stoga su aksiomi i pravila dokazivanja iz $\mathcal{K}$ koji su prijevodi aksioma i pravila dokazivanja $\mathcal{H}$ dobri, kao što je sustav $\mathcal{H}$ utemeljen u slučaju deklariranja nerekurzivne procedure.

Po prethodnim tvrdnjama, da bi se dokazala utemeljenost od $\mathcal{G}$ sada je dovoljno dokazati sljedeće:

1. valjanost aksioma invarijance u slučaju deklaracije nerekurzivne procedure;

2. utemeljenost pravila supstitucije I i II i pravila konjunkcije u gornjem slučaju;

3. dobrota pravila
$$\frac{\{A\}\ p\ \{B\} \to \{A\}\ c_0\ \{B\}}{\{A\}\ p\ \{B\}},$$
to jest, da za bilo koju interpretaciju I vrijedi: ako za sve n $\langle p \Longleftarrow c_0^{(n)} \mid \{A\}\ p\ \{B\} \to \{A\}\ c_0\ \{B\}\rangle$ je istina u interpretaciji I, onda za sve n je i $\langle p \Longleftarrow c_0^{(n)} \mid \{A\}\ p\ \{B\}\rangle$ istina u interpretaciji I;

4. dobrota selekcijskog aksioma i pravila sakupljanja.

Dokaz tvrdnji 1. i 2. slijedi direktno.

Da bismo dokazali 3., pretpostavimo da je za danu interpretaciju I i sve n

$$\langle p \Longleftarrow c_0^{(n)} \mid \{A\}\ p\ \{B\} \to \{A\}\ c_0\ \{B\}\rangle \tag{4.12}$$

istina u interpretaciji I.

Očito je $\langle p \Longleftarrow c_0^{(0)} \mid \{A\}\ p\ \{B\}\rangle$ istina u I. Pretpostavimo sada da je za neki n

$$\langle p \Longleftarrow c_0^{(n)} \mid \{A\}\ p\ \{B\}\rangle \tag{4.13}$$

istina u I.

Tada po (4.12) vrijedi da je

$$\langle p \Longleftarrow c_0^{(n)} \mid \{A\}\ c_0\ \{B\}\rangle \tag{4.14}$$

istina u I. Ali $c_0\ [\ c_0^{(n)}/p\ ] \equiv c_0^{(n+1)}$, pa (4.14) implicira da je $\langle p \Longleftarrow c_0^{(n+1)} \mid \{A\}\ p\ \{B\}\rangle$ istina u interpretaciji I. Dakle, po indukciji, (4.13) vrijedi za sve n.

4. je očito istina.

Dakle, sustav $\mathcal{G}$ je uistinu utemeljen.

4.5 Potpunost $\mathcal{G}$ u Cook-ovom smislu

Želimo dokazati potpunost $\mathcal{G}$ za Com_1 u Cook-ovom smislu. Neka je dana naredba $c_0 \in Com_1$. Neka je $\tilde{X}$ niz svih varijabli iz skupa *Loc* koje se pojavljuju u naredbi c_0 i $\tilde{Z}$ niz po volji odabranih varijabli iz *Loc*, a takav da je broj varijabli u $\tilde{Z}$ jednak broju varijabli u $\tilde{X}$. Pretpostavimo da je $I \in Exp(\mathrm{IMP}, Com_1)$. Postoji tvrdnja $B_0 \in Assn$ koja definira $post_I(\tilde{X} = \tilde{Y}, p)$.

Formula parcijalne korektnosti $\{\tilde{X} = \tilde{Z}\}\ p\ \{B_0\}$ se naziva *najopćenitija formula za* p, jer bilo koja druga formula parcijalne korektnosti o p koja je istinita (u I) može biti izvedena u $\mathcal{G}$ iz $\{\tilde{X} = \tilde{Z}\}\ p\ \{B_0\}$. Ova tvrdnja je sadržaj sljedeće leme.

Lema 4.10. Ako $\models_I \{A\}\ c\ \{B\}$, onda $Tr_I \cup \{\tilde{X} = \tilde{Z}\}\ p\ \{B_0\} \vdash_{\mathcal{G}} \{A\}\ c\ \{B\}$.

Dokaz. Dokaz se provodi indukcijom po duljini c. Ako je $c \neq p$, dokaz je identičan dokazu potpunosti za $\mathcal{H}$.

Pretpostavimo da je $c = p$.

Neka vrijedi

$$\{\tilde{X} = \tilde{Z}\}\ p\ \{B_0\} \tag{4.15}$$

Po aksiomu invarijance

$$\{A_1[\tilde{Z}/\tilde{X}]\}\ p\ \{A_1[\tilde{Z}/\tilde{X}]\}, \tag{4.16}$$

gdje je $A_1 = A[\tilde{U}/\tilde{Z}]$ i $\tilde{U}$ je niz novih varijabli jednake duljine kao i $\tilde{Z}$. (4.15) i (4.16) impliciraju po pravilu konjunkcije da

$$\{\tilde{X} = \tilde{Z} \wedge A_1[\tilde{Z}/\tilde{X}]\}\ p\ \{B_0 \wedge A_1[\tilde{Z}/\tilde{X}]\}. \tag{4.17}$$

Sada želimo pokazati da

$$\models_I B_0 \wedge A_1[\tilde{Z}/\tilde{X}] \rightarrow B_1, \text{ gdje je } B_1 = B[\tilde{U}/\tilde{Z}]. \tag{4.18}$$

Pretpostavimo $\models_I (B_0 \wedge A_1[\tilde{Z}/\tilde{X}])(\tau)$.

Po definiciji B_0 postoji stanje δ takvo da $\mathcal{M}_I(p)(\sigma) = \tau$ i $\mathcal{M}_I(\tilde{X} = \tilde{Z})(\sigma)$.

Sada pretpostavimo da $\models_I \neg A_1[\tilde{Z}/\tilde{X}](\sigma)$.

Sada po valjanosti aksioma invarijance i zbog $\mathcal{M}_I(p)(\sigma) = \tau$ vrijedi $\models_I \neg A_1[\tilde{Z}/\tilde{X}](\tau)$.

Ovo je kontradikcija s pretpostavkom, pa $\models_I A_1[\tilde{Z}/\tilde{X}](\sigma)$.

Kako je $\models_I (\tilde{X} = \tilde{Z} \wedge A_1[\tilde{Z}/\tilde{X}]) \rightarrow A_1$, sada dobijamo $\models_I A_1(\sigma)$.

No, po potpunosti pravila supstitucije I, također $\models_I \{A_1\}\ p\ \{B_1\}$, pa konačno $\models_I B_1(\tau)$.

Time je dokazano (4.18).

(4.17) i (4.18) impliciraju po pravilu posljedice da

$$\{\tilde{X} = \tilde{Z} \wedge A_1[\tilde{Z}/\tilde{X}]\}\ p\ \{B_1\}.$$

Sada po pravilu supstitucije II

$$\{\tilde{X} = \tilde{X} \wedge A_1\}\ p\ \{B_1\},$$

kako $\tilde{X} = \tilde{X} \wedge A_1 \equiv (\tilde{X} = \tilde{Z} \wedge A_1[\tilde{Z}/\tilde{X}])[\tilde{X}/\tilde{Z}]$. Po pravilu posljedice

$$\{A_1\}\ p\ \{B_1\},$$

pa po pravilu supstitucije I

$$\{A_1[\tilde{Z}/\tilde{U}]\}\ p\ \{B_1[\tilde{Z}/\tilde{U}]\}.$$

Očito, $\models_I A \leftrightarrow A_1[\tilde{Z}/\tilde{U}]$ i $\models_I B \leftrightarrow B_1[\tilde{Z}/\tilde{U}]$, pa po pravilu posljedice vrijedi

$$\{A\}\ p\ \{B\},$$

što je i trebalo dokazati. □

Sljedeća lema pokazuje da se hipoteza $\{\tilde{X} = \tilde{Z}\}\ p\ \{B_0\}$, korištena u prethodnoj lemi, može zapravo dokazati u $\mathcal{G}$. Primijetimo da su u gornjem dokazu pomoćne varijable u $\tilde{Z}$ korištene za "zamrzavanje" vrijednosti varijabli iz $\tilde{X}$ prije poziva procedure.

Lema 4.11. $Tr_I \vdash_{\mathcal{G}} \{\tilde{X} = \tilde{Z}\}\ p\ \{B_0\}$.

Dokaz. Dokaz slijedi direktno. Po definiciji B_0, $\models_I \{\tilde{X} = \tilde{Z}\}\ p\ \{B_0\}$, pa

$$\models_I \{\tilde{X} = \tilde{Z}\}\ c_0\ \{B_0\}$$

kako je $\mathcal{M}_I(p) = \mathcal{M}_I(c_0)$.

Po prethodnoj lemi,

$$Tr_I \cup \{\tilde{X} = \tilde{Z}\}\ p\ \{B_0\}$$

$$\vdash_{\mathcal{G}} \{\tilde{X} = \tilde{Z}\}\ c_0\ \{B_0\},$$

pa po pravilu rekurzije,

$$Tr_I \vdash_{\mathcal{G}} \{\tilde{X} = \tilde{Z}\}\ p\ \{B_0\}.$$

□

Potpunost od $\mathcal{G}$ je direktna posljedica prethodnih dviju Lema.

Poglavlje 5

Deklaracija varijabli

Razmotrimo sada posljednju klasu Com^V koja, kao i klasa Com, sadrži izjave o dodjeljivanju. Zatvorena je za upotrebu pravila kompozicije(;), while i kondicionala, ali dodatno zadovoljava i sljedeće pravilo:

ako je $c \in Com^V$, onda za svaki $X \in Loc$ vrijedi: begin new $X; c$ end $\in Com^V$.

U prethodnom, new X označava deklaraciju varijable $X \in Loc$ koja vrijedi unutar bloka begin new $X; c$ end; X je *lokalna varijabla* obzirom na ovaj blok.

Primjer 5.1. Pokažimo primjenu prethodno uvedenog uvjeta. Neka je dana

$c := Y := 2;$

begin new $X; X := 1$

while $X < 5$ do $Y := Y \cdot X;\ X := X + 1;$

end;

begin new $X; X := 0$

while $X < 20$ do $X := X + 5;\ Y := Y + X;$

end.

Pri završetku prve *while* petlje naredbe $c \in Com^V$ varijabla $X \in Loc$ se "briše"

iz programa te ju možemo ponovno deklarirati u novom bloku i nanovo koristimo u sljedećoj while petlji. Na kraju izvođenja programa, varijabla $Y \in Loc$ ima vrijednost 68.

Definicija 5.2. Pojava varijable $X \in Loc$ u naredbi $c \in Com^V$ naziva se *vezanom* kad god se nalazi u podnaredbi od c koja je oblika begin $X; c_1$ end, za neku $c_1 \in Com^V$. Pojava X u c je *slobodna* ako nije vezana.

Označimo s $F(c)$ skup svih varijabli iz Loc koje se pojavljuju slobodne u $c \in Com^V$.

Napomena 5.3. Prisjetimo se da smo u potpoglavlju kod jezika tvrdnji $Assn$ strukturalnom indukcijom definirali skup slobodnih varijabli tvrdnje $A \in Assn$ i označili ga s $F(A)$. Sada s $F(c, A)$ označavamo skup slobodnih varijabli iz Loc u naredbi $c \in Com^V$ i tvrdnji $A \in Assn$ takav da vrijedi $F(c, A) = F(c) \cap F(A)$. Slično, $F(A, c, B) = F(A) \cap F(c) \cap F(B)$ je skup svih slobodnih varijabli u naredbi $c \in Com^V$ i tvrdnjama $A, B \in Assn$.

Sa $c[Y/X]$ označavamo supstituciju $Y \in Loc$ za sve slobodne pojave $X \in Loc$ u naredbi $c \in Com^V$. Definicija je analogna supstituciji $A[Y/X]$ za tvrdnju $A \in Assn$.

Neka nam Λ označava posebnu konstantu na koju inicijaliziramo sve lokalne varijable. Mogli bismo promatrati Λ kao simbol koji označava nedefiniranu vrijednost ("$undefined$"). Sada donosimo sljedeće dokazno pravilo:

Pravilo deklaracije varijable

$$\frac{\{A \wedge Y = \Lambda\}\ c\ [\ Y/X\ ]\ \{B\}}{\{A\}\ \text{begin new } X; c\ \text{end}\ \{B\}}, \text{ gdje } Y \notin F(A, c, B) \text{ i}$$

$A, B \in Assn,\ X, Y \in Loc,\ c \in Com^V$.

Ovdje se za varijable $X, Y \in Loc$ provodi preimenovanje X s Y kako bi se

razlikovala pojava lokalne varijable X u $c \in Com^V$ i moguće slobodne pojave nelokalne varijable X u A i B, za $A, B \in Assn$. Izraz $Y = \Lambda$ obuhvaća ideju inicijalizacije.

5.1 Semantika za deklaraciju varijabli

Da bismo razmatrali pitanje utemeljenosti pravila deklaracije varijable, kao i obično, moramo definirati funkciju značenja.

Napomena 5.4. Prisjetimo se da smo u prvom poglavlju stanje definirali kao funkciju $\sigma : Loc \to N$ sa skupa lokacija na skup brojeva. Pritom nismo zahtijevali da σ mora biti definirana na cijelom skupu Loc, već na nekom njegovom podskupu. Sa Σ smo označili skup stanja.

U nastavku sa $dom(\sigma)$ označavamo domenu funkcije σ tj. skup lokacija $X \in Loc$ za koje je $\sigma(X)$ definirano.

Za stanje $\sigma \in \Sigma$, ako $X \notin dom(\sigma)$ i $d \in Intvar$, onda $\sigma \cup (X, d)$ označava proširenje stanja σ dodavanjem vrijednosti d kada se primjenjuje na X. $DROP(\sigma, X)$ označava stanje dobiveno iz σ uklanjanjem varijable X iz njegove domene $dom(\sigma)$.

Značenje naredbe definiramo na isti način kao i prije, s tim da je jedini novi slučaj blok. Za sve naredbe $c \in Com^V$ pretpostavljamo da je $\mathcal{M}_I(c)(\sigma)$ nedefiniran ako $F(c) \not\subseteq Loc$.

Neka je $c_1 \equiv$ begin new $X; c$ end i pretpostavimo da vrijedi $F(c_1) \subseteq Loc$. Definiramo

$$\mathcal{M}_I(c_1)(\sigma) = DROP(\ \mathcal{M}_I(c\ [\ Y/X\ ]\ )(\sigma \cup (Y, \underline{\Lambda}))\ ,\ Y\ )$$

gdje je Y prva varijabla na koju nailazimo prolazeći po c_1, a koja nije element Loc i $\underline{\Lambda}$ je vrijednost dodjeljena konstanti Λ.

5.2 Utemeljenost i potpunost sustava $\mathcal{H}$+*Pravilo deklaracije varijabli*

Utemeljenost pravila deklaracije varijable slijedi iz činjenice da

$$\mathcal{M}_I(\ c\ [\ Y/X\ ]\ )(\ \sigma\ \cup\ (Y,d)\ ) = \mathcal{M}_I(\ c\ [\ Z/X\ ]\ )(\ \sigma\ \cup\ (Z,d)\ ), \tag{5.1}$$

za bilo koju naredbu c i $Y, Z \notin Loc$.

Da bismo dokazali tu činjenicu, trebalo bi zapravo pojačati tvrdnju i radije dokazati da je za bilo koju naredbu c i $Y_1, ..., Y_n, Z_1, ..., Z_n \notin Loc$,

$$\mathcal{M}_I(\ c\ [\ Y_1/X_1\ ]...[\ Y_n/X_n\ ]\ )(\ \sigma\ \cup\ (Y_1,d_1)...(Y_n,d_n)\ )$$

$$= \mathcal{M}_I(\ c\ [\ Z_1/X_1\ ]...[\ Z_n/X_n\ ]\ )(\ \sigma\ \cup\ (Z_1,d_1)...(Z_n,d_n)\ ).$$

Posljednja tvrdnja se može dokazati indukcijom po strukturi naredbe c, gdje samo slučaj blokova zahtijeva određen oprez. Utemeljenost pravila iz $\mathcal{H}$ je već dokazana. Dakle, sustav $\mathcal{H}$+ *Pravilo deklaracije varijabli* je utemeljen.

Sada ćemo promatrati problem potpunosti sustava $\mathcal{H}$+ *Pravilo deklaracije varijabli*. Lako se može vidjeti da je sustav potpun u Cook-ovom smislu. Slučajevi blokova rješavaju se koristeći (5.1), a ostali slučajevi su isti kao u dokazu potpunosti $\mathcal{H}$.

5.3 Dodavanje procedura

Riješivši slučaj while programa, prelazimo na programe koji omogućuju pozive procedura.

Razmotrimo proširenje Com_1^V od Com^V u kojem je naredbama dozvoljeno da sadrže pozive nerekurzivne bezparametarske procedure p. Pretpostavljamo

deklaraciju $p \Longleftarrow c_0$, gdje je $c_0 \in Com^V$.

Program $c \in Com_1^V \setminus Com^V$ poprima značenje koje mu je dodjeljeno odredbom

$$\mathcal{M}_I(c) = \mathcal{M}_I(\ c\ [\ c_0/p\ ]\ ),$$

gdje $c\ [\ c_0/p\ ]$ predstavlja zamjenu svih pojava p s c_0. Potrebno je pažljivo definirati $c\ [\ c_0/p\ ]$. $c\ [\ c_0/p\ ]$ je definirano indukcijom, s tim da je glavna odredba:

$$\text{begin new } X; c \text{ end } [\ c_0/p\ ]$$

$$\equiv \begin{cases} \text{begin new } X; c\ [\ c_0/p\ ] \text{ end} & \text{,ako } X \notin F(c_0) \\ \text{begin new } X'; c\ [\ X'/X\ ][\ c_0/p\ ] \text{ end, gdje } X' \notin F(c, c_0) & \text{,inače.} \end{cases}$$

Cilj ove odredbe je izbjeći vezanje bilo koje od slobodnih pojava varijabli iz c_0 kroz postupak supstitucije. Ostale odredbe su definirane prirodnim putem. Kako bismo dokazali ispravnost naredbi iz Com_1^V, sada koristimo dodatno pravilo pozivanja procedure. Međutim, da bismo primijenili pravilo deklaracije varijabli na naredbe iz Com_1^V potreban je dodatan zahtjev, a to je da $Y \notin F(c_0)$. Kako bismo ilustrirali upotrebu pravila deklaracije varijabli u vezi s pravilom pozivanja procedure, dat ćemo sljedeći primjer.

Primjer 5.5. Razmotrimo deklaraciju procedure $p \Longleftarrow X := Z$ i naredbu $c \equiv Z := 1; \text{begin new} Z; Z := 0; p \text{ end}.$

Sada dokazujemo $\{true\}\ c\ \{X = 1\}$. Po pravilu za dodjeljivanje

$$\{Z = 1\}\ X := Z\ \{X = 1\},$$

pa po pravilu pozivanja procedure

$$\{Z = 1\}\ p\ \{X = 1\}.$$

Po pravilima dodjeljivanja, posljedice i nizanja

$$\{Z = 1 \wedge Y = \Lambda\}\ Y := 0; p\ \{X = 1\},$$

iz čega dobivamo željenu formulu $\{true\}\ c\ \{X = 1\}$.

Sada dokazujemo utemeljenost i potpunost u Cook-ovom smislu sustava $\mathcal{H}+$ *Pravilo pozivanja procedure, Pravilo deklaracije varijabli.*

Gore definirana semantika blokova daje značenje za programe iz Com_1^V u smislu značenja programa iz Com^V. Stoga možemo lako reducirati problem na utemeljenost i potpunost u Cook-ovom smislu sustava $\mathcal{H}+$ *Pravilo deklaracije varijabli* za Com^V. Jedini slučaj u oba dokaza koji zahtijeva neko objašnjenje je slučaj bloka.

Ako $X, Y \notin F(c_0)$, tada za $c \in Com_1^V$,

$$\mathcal{M}_I(\ c\ [\ Y/X\ ]\ ) = \mathcal{M}_I(\ c\ [\ Y/X\ ]\ [\ c_0/p\ ]\ ) = \mathcal{M}_I(\ c\ [\ c_0/p\ ]\ [\ Y/X\ ]\ )$$

i

$$\mathcal{M}_I(\text{ begin new } X; c \text{ end }) = \mathcal{M}_I(\text{ begin new } X; c\ [\ c_0/p\ ] \text{ end }).$$

Ako $Y \notin F(c_0$ i $X \in F(c_0)$, onda

$$\mathcal{M}_I(\ c\ [\ Y/X\ ]\ ) = \mathcal{M}_I(\ c\ [\ Y/X\ ]\ [\ c_0/p\ ]\ ) = \mathcal{M}_I(\ c\ [\ X'/X\ ]\ [\ c_0/p\ ]\ [\ Y/X'\ ]\ )$$

i

$$\mathcal{M}_I(\text{begin new } X; c \text{ end}) = \mathcal{M}_I(\text{begin new } X; c \text{ end } [\ c_0/p\ ])$$

$$= \mathcal{M}_I(\text{begin new } X'; c[\ X'/X\ ]\ [\ c_0/p\ ] \text{ end}), \text{ gdje } X' \notin F(c, c_0).$$

To pokazuje da u oba slučaja utemeljenost pravila deklaracije varijabli koje se primjenjuje na programe iz Com_1^V uistinu proizlazi iz utemeljenosti pravila deklaracije varijabli primjenjenog na programe iz Com^V. Slično reduciranje se provodi u odgovarajućem slučaju u dokazu potpunosti.

Sada razmotrimo slučaj rekurzivnih procedura. Da bismo pružili semantiku u slučaju kada je p rekurzivna procedura, postupamo kao i prije, kod promatranja utemeljenosti sustava $\mathcal{G}$. Ako sada pretpostavimo da $c_0 \in Com_1^V \setminus Com^V$,

možemo definirati značenje naredbe $c \in Com_1^V \setminus Com^V$ tako da stavimo

$$\mathcal{M}_I(S) = \bigcup_{n=0}^{\infty} \mathcal{M}_I(\ c\ [\ c_0^{(n)}/p\ ]\ ), \text{ gdje je } c_0^{(0)} \equiv \Omega$$

$$c_0^{(n+1)} = c_0\ [\ c_0^{(n)}/p\ ].$$

Koristeći ove definicije, sustav dokaza $\mathcal{G}+$ *Pravilo deklaracije varijabli* za Com_1^V je utemeljen i potpun u Cook-ovom smislu. Argumenti koji se koriste u odjeljcima u kojim se dokazuje utemeljenost i potpunost u Cook-ovom smislu mogu se i ovdje primijeniti bez ikakvih promjena. Dodatni slučaj blokova se svodi na gore već obrađeni slučaj naredbi iz Com^V.

5.4 Problemi s neinicijaliziranim varijablama

U pravilu deklaracije varijable smo ugradili pretpostavku da se svaka lokalna varijabla inicijalizira pomoću formule $Y = \Lambda$. Da bismo dokazali ispravnost (i potpunost) ovog pravila, bili smo kasnije primorani razmotriti ovu pretpostavku definirajući semantiku za blokove. No, je li potrebna ta pretpostavka?
Mogli bismo izbaciti formulu $Y = \Lambda$ iz premise pravila i, pružajući semantiku, inicijalizirati svaku lokalnu varijablu u, recimo, prvi element domene od I koji nije u rasponu σ. Treba biti jasno da s takvim izmjenama zahtjev (5.1) zadržava valjanost, tako da i dokazi potpunosti i utemeljenosti ostaju valjani.
Zašto onda nismo usvojili ovo jednostavnije rješenje?
Odgovor je suptilan. Razmotrimo sljedeću tvrdnju parcijalne korektnosti:

$$\{true\}\ \text{begin new } Z; X := Z \text{ end; begin new } U; Y := U \text{ end } \{X = Y\}.$$

Prema semantici koju smo prihvatili, kao i semantici koju smo upravo predložili, ova formula je istina pod bilo kojim interpretacijom I. To je lako dokazati i u

sustavu $\mathcal{H}$+*Pravilo deklaracije varijabli*, jer se jednostavno

$$\{true\} \text{ begin new } Z; X := Z \text{ end } \{X = \Lambda\}$$

i

$$\{X = \Lambda\} \text{ begin new } U; Y := U \text{ end } \{X = Y\}$$

mogu dokazati.

Ako, međutim, usvojimo upravo predloženi prijedlog, ne možemo pronaći srednju tvrdnju koja bi igrala ulogu $X = \Lambda$ otprije. Štoviše, predložena semantika rezultira neizrazivošću IMP-a u odnosu na bilo koju interpretaciju I i Com^V! Slučaj u kojem je $|I| = 1$ treba biti isključen ovdje, jer je predložena semantika tada pogrešno definirana. Da bismo to vidjeli, imajmo na umu da skup $post_I(true, \text{ begin new } Z; X := Z \text{ end})$ nije moguće definirati nijednom formulom iz IMP-a. Stoga je dokaz potpunosti doista valjan, ali prazan.

Svi problemi ovdje su uzrokovani uporabom neinicijaliziranih lokalnih varijabli. Te poteškoće bismo mogli jednostavno izbjeći odbacivanjem naredbi u kojima neke lokalne varijable nisu inicijalizirane. Takva se klasa naredbi lako može definirati, a u njezinu rješavanju može se primijeniti novo-predloženi pristup.

Poglavlje 6

Subskriptne varijable

Do sada smo dopuštali dodjeljivanje vrijednosti samo jednostavnim varijablama. Dopuštanje subskriptnih varijabli u izrazima i dodjeljivanjima dovodi do proširenja prethodne sintakse.

Niz varijabli je skupina varijabli pohranjenih pod istim imenom, ali s različitim vrijednostima indeksa. Svaki element niza ima ime (koje je jednako imenu niza) i indeks (između zagrada) koji omogućuje odabir elementa. Nizovi mogu biti jednodimenzionalni (polja) ili višedimenzionalni. Subskriptne varijable koriste se za spremanje vrijednosti istog tipa u niz, a deklariraju se davanjem imena varijable zajedno s indeksom. Primjer: $R[10]$. Ovdje je R ime varijable, a 10 je subskriptna vrijednost pomoću koje možemo nizu R na 11. mjestu dodijeliti vrijednost koja je istog tipa kao i preostale vrijednosti niza. Da bi stvari bile jednostavne ograničavamo našu pažnju na slučaj jednodimenzionalnih nizova izostavljajući bilo koje specifikacije granica.

Označimo s AV skup varijabli niza. Sintaksu IMP-a proširujemo dopuštajući izraze oblika $R[t]$ za $R \in AV$ i t izraz, te sada dopuštamo dodjeljivanje $R[s] := t$,

gdje je $R \in AV$, s, t izrazi (aritmetički ili logički).

6.1 Pravilo dodjeljivanja za subskriptne varijable

U nastavku ćemo pretpostaviti da su kondicionali oblika if $s = t$ then t_1 else t_2 (tzv. uvjeti jednakosti) dozvoljeni. Da bismo stekli bolju sliku problema, prvo razmotrimo slučaj dodjeljivanja kada je subskript jednostavna varijabla.
S $A\ [\ t/R[s]\ ]$ označavamo zamjenu subskriptne varijable $R[s]$ izrazom t. Definira se indukcijom, pri čemu je glavna odredba

$$R[z]\ [\ t/R[x]\ ] \equiv \text{if } z = x \text{ then } t \text{ else } R[z].$$

Predložen je sljedeći aksiom:

$$\{A[\ t/R[x]\ ]\}\ R[x] := t\ \{A\}.$$

Primjer 6.1. Pomoću prethodnog aksioma, možemo dokazati

$$\{x = y\}\ R[x] := 1\ \{R[y] = 1\}$$

budući da je

$$(R[y] = 1)\ [\ 1/R[x]\ ] \equiv \text{if } y = x \text{ then } 1 \text{ else } R[y] \quad = 1,$$

ovo slijedi iz $x = y$. Također, vrijedi

$$\{true\}\ R[x] := 1\ \{R[x] = 1\}.$$

Ako sada dozvolimo proizvoljne izraze kao subskripte nailazimo na poteškoće.

Primjer 6.2. Formula

$$\{true\}\ R[t] := 1\ \{R[t] = 1\}$$

više nije valjana! Da bismo to vidjeli, promotrimo da

$$\text{if } R[1] = R[2] = 2 \text{ then } \{true\}\ R[R[2]] := 1\ \{\ R[R[2]] := 1\ \}$$

nije istina.

To pokazuje da gornja definicija zamjene mora biti na odgovarajući način definirana za opći slučaj subskripata koji su proizvoljni izrazi. Možda je najjednostavnije rješenje ovog problema zaobići ga. Gornja supstitucija i dalje vodi k ispravnim rezultatima kad se koristi za proizvoljne subskriptne varijable, ako se primjenjuje samo na naredbe koje omogućuju jednostavne varijable kao subskripte. Stoga je glavna odredba ove zamjene sljedeća

$$R[z][\ t/R[s]\ ] \equiv \text{ if } z = s \text{ then } t \text{ else } R[s],$$

a slučaj u kojem je z proizvoljni izraz se jednostavno ne obrađuje.
Neka je sada $A \in Assn$ proizvoljan i neka A' označava tvrdnju ekvivalentnu A koja se dobiva kvantificiranjem svih subskripata, koji nisu jednostavne varijable.

Primjer 6.3. Ako je $A = R[R[2]] = 1$, tada je $A'\ \exists z\ (R[z] = 1 \wedge z = R[2])$

Sada proširujemo gornju zamjenu stavljanjem $A[\ t/R[s]\ ] \equiv A'[\ t/R[s]\ ]$. Konačno stižemo do općeg oblika pravila.

Pravilo dodjeljivanja za subskriptne varijable

$$\{A[\ t/R[s]\ ]\}\ R[s] := t\ \{A\},$$

gdje je $A \in Assn,\ R \in AV,\ s, t$ izrazi (aritmetički ili logički).
Primijetimo sličnost između ovog pravila i pravila za dodjeljivanje iz sustava $\mathcal{H}$.

Primjer 6.4. Dokažimo da vrijedi

$$\{\models (R[2] = 2 \Rightarrow R[1] = 1)\}\ R[R[2]] := 1\ \{R[R[2]] = 1\}.$$

Imamo

$$(R[R[2]] = 1)[\ 1/R[R[2]]\ ] \equiv (\exists z\ (R[z] = 1 \wedge R[2] = z))[\ 1/R[R[2]]\ ]$$

$\equiv \exists z$ ((if $z = R[2]$ then 1 else $R[z]$) $= 1 \wedge$(if $2 = R[2]$ then 1 else $R[2]$) $=$ z).

Posljednja formula slijedi iz tvrdnje

$$\models (R[2] = 2 \Rightarrow R[1] = 1).$$

Naime, ako vrijedi $R[2] = 2$, onda izaberemo $z = 1$, a u protivnom $z = R[2]$. Dakle, po pravilu posljedice i pravilu dodjeljivanja za subskriptne varijable dobijamo željeni rezultat.

Pravilo dodjeljivanja za subskriptne varijable je također potpuno u sljedećem smislu. Ako $\models_I \{A\}\ R[s] := t\ \{B\}$, onda $\models_I (A \Rightarrow B[\ t/R[s]\ ])$. Dakle, ako $\models_I \{A\}\ R[s] := t\ \{B\}$, onda $Tr_I \vdash_{\textit{Pravilo posljedice+Pravilo dodjekjivanja za subskriptne varijable}} \{A\}\ R[s] := t\ \{B\}$, gdje je Tr_I skup svih tvrdnji koje su istinite u interpretaciji I.

Poglavlje 7

Mehanizmi parametara

Mehanizmi parametara spadaju u problematična pitanja u okviru Hoareove logike. Jedan od razloga je taj što prolazak parametara uvijek sintaktički modelira neki oblik zamjene varijabli u naredbi, a to vodi do različitih suptilnih problema među varijablama. Ove su poteškoće posebno akutne u prisutnosti rekurzije.

Za razliku od izlaganja prethodnih poglavlja, ovdje prezentacija ne može biti potpuna, jer nisu riješeni svi problemi u ovom području. U nastavku pokušavamo razjasniti koja točno pitanja vode do poteškoća i navesti što još preostaje učiniti u ovom području.

Započinjemo s promatranjem mehanizma parametara poziva po imenu procedure. Ovi se rezultati ne tiču većine uobičajenih značajki programskih jezika. Međutim, tehnike uvedene za njihovo rješavanje čine odgovarajuću osnovu za proučavanje češćih mehanizama parametara. Stoga je korisno najprije promatrati te značajke.

7.1 Poziv nerekurzivne procedure po imenu

Promotrimo deklaraciju procedure oblika

$$p \Longleftarrow \langle \text{name}(\tilde{X} : \tilde{V}) \mid c_0 \rangle,$$

gdje je $(\tilde{X} : \tilde{V})$ formalna lista parametara i $c_0 \in Com^V$ tijelo procedure. $\tilde{X}$ i $\tilde{V}$ su odvojene liste različitih varijabli, a varijable u $\tilde{V}$ se ne mogu pojaviti slijeva od bilo koje naredbe dodjeljivanja u c_0. c_0 ne sadrži pozive procedura, pa p nije rekurzivna.

U proširenju od Com^V, zvanom Com_2^V, dopuštamo pozive procedura oblika $p(\tilde{U} : \tilde{T})$, gdje je $\tilde{U}$ lista različitih varijabli, $\tilde{T}$ lista izraza koji ne sadrže varijable iz $\tilde{U}$ i nema varijable u $(\tilde{U} : \tilde{T})$ različite od formalnih parametara, a koja se pojavljuje slobodna u tijelu procedure c_0.

Za sve pozive procedura spomenute u nastavku se pretpostavlja da udovoljavaju gornjim ograničenjima.

$c_0[\tilde{U}, \tilde{T}/\tilde{X}, \tilde{V}]$ označava rezultat istodobne zamjene stvarnim parametrima $\tilde{U}, \tilde{T}$ odgovarajućih slobodnih pojava formalnih parametara $\tilde{X}$ i $\tilde{V}$ u c_0.

Napomena 7.1. Kad kažemo formalni parametri mislimo na varijable koje se koriste u proceduri, a stvarni parametri su konkretne vrijednosti koje pridjeljujemo tim varijablama.

Sada dopunjavamo dokazni sustav $\mathcal{H}$+*Pravilo deklaracije varijabli* sa sljedeća tri pravila dokazivanja:

Pravilo pozivanja procedure II

$$\frac{\{A\}\ c_0\ \{B\}}{\{A\}\ p(\tilde{X} : \tilde{V})\ \{B\}}.$$

Pravilo zamjene parametra

$$\frac{\{A\}\ p(\tilde{X}' : \tilde{V}')\ \{B\}}{\{p[\tilde{U}, \tilde{T}/\tilde{X}', \tilde{V}']\}\ p(\tilde{U} : \tilde{T})\ \{B[\tilde{U}, \tilde{T}/\tilde{X}', \tilde{V}']\}}, \quad \text{gdje je } \tilde{U} \cap F(A, B) \subseteq \tilde{X}'.$$

Pravilo zamjene varijable

$$\frac{\{A\}\ p(\tilde{U}:\tilde{T})\ \{B\}}{\{A[\ \tilde{S}/\tilde{Z}\ ]\}\ p(\tilde{U}:\tilde{T})\ \{B[\tilde{S}/\tilde{Z}]\}}, \quad \text{gdje se nijedna varijabla iz } \tilde{S} \text{ ni } \tilde{Z}$$

ne pojavljuje slobodna u $c_0[\tilde{U}, \tilde{T}/\tilde{X}, \tilde{V}]$.

Rezultirajući sustav označimo sa $\mathcal{C}$.

Posljednja dva pravila je prilično teško razumjeti zbog ograničenja nametnutih na supstituciju izraza i varijabli. Da bismo dobili bolju predstavu o tome kako se koriste ta pravila dokaza, promotrimo sljedeći primjer.

Primjer 7.2. Primjer dokaza u $\mathcal{C}$

Neka je zadana deklaracija procedure $p \Longleftarrow \langle \text{name}(X : V) \mid X := V; A := V \rangle$. Prema nametnutim ograničenjima, pozivi $p(Y : Y + 1)$ ili $p(Y : A + 1)$ su onemogućeni, ali pozivi $p(Z : Y + 1), p(V : Y + 1)$ ili $p(V : X + 1)$ su dozvoljeni. Sada dokazujemo

$$\{X = Z\}\ p(V : Y + 1)\ \{V = Y + 1 \wedge A = Y + 1 \wedge X = Z\}.$$

U tu svrhu moramo preimenovati formalni parametar X procedure, koji se pojavljuje slobodan u tvrdnjama. Stoga prvo dokazujemo

$$\{U = Z\}\ p(V : Y + 1)\ \{V = Y + 1 \wedge A = Y + 1 \wedge U = Z\}.$$

Imamo

$$\{U = Z\}\ X := V; A := V\ \{V = Y + 1 \wedge X = V \wedge A = V \wedge U = Z\},$$

pa po pravilu pozivanja procedure II

$$\{U = Z\}\ p(X : V)\ \{X = V \wedge A = V \wedge U = Z\}.$$

Sada, po pravilu zamjene parametra

$$\{U = Z\}\ p(X : V')\ \{X = V' \wedge A = V' \wedge U = Z\},$$

pa još jednom, po istom pravilu

$$\{U = Z\}\ p(V : Y + 1)\ \{V = Y + 1 \wedge A = Y + 1 \wedge U = Z\}.$$

Konačno, po pravilu zamjene varijable

$$\{X = Z\}\ p(V : Y + 1)\ \{V = Y + 1 \wedge A = Y + 1 \wedge X = Z\}.$$

Napomena 7.3. Izravan korak od poziva $p(X : V)$ do $p(V : Y + 1)$ nije dopušten. Naime, pravilo zamjene parametra ovdje zahtijeva da je stvarni parametar V u pozivu $p(V : Y + 1)$ jednak X u $p(X : V)$, kako se V pojavljuje slobodan u tvrdnji $X = V \wedge A = V \wedge U = Z$.

Napomena 7.4. Vrijedi napomenuti da su ograničenja u pravilima zamjene nužna.

Da bismo to vidjeli, razmotrimo sljedeće primjere.

Primjer 7.5. Neka je deklarirana procedura $p \Longleftarrow \langle \text{name}(X :) \mid X := 0 \rangle$. Sada imamo $\{U = 1\}\ p(X :)\ \{U = 1\}$, ali naravno $\{U = 1\}\ P(U :)\ \{U = 1\}$ nije istinito. Ovo pokazuje da je restrikcija u pravilu zamjene parametra nužna.

Primjer 7.6. $\{X = 1\}\ P(X :)\ \{X = 1\}$ nije istinito, pa je odgovarajuća restrikcija u pravilu zamjene varijable također nužna.

Nastavimo s raspravom o sustavu $\mathcal{C}$. Neka je $c \in Com_2^V \setminus Com^V$. Značenje od c definiramo stavljajući

$$\mathcal{M}_I(c) = \mathcal{M}_I(c(\ c_0/p\ )),$$

gdje je $c(c_0/p) \in Com^V$ rezultat doslovne zamjene svakog poziva procedure $p(\tilde{U} : \tilde{T})$ koji se pojavljuje u c sa $c_0[\ \tilde{U}, \tilde{T}/\tilde{X}, \tilde{V}\ ]$.

Jedini novi slučaj u dokazivanju potpunosti je onaj za pozive procedura. Predstavit ćemo argument koji djeluje samo ako formalni i stvarni parametri

nemaju zajedničke varijable i ako tvrdnje A i B nemaju slobodne pojave formalnih parametara.

Pretpostavimo da je

$$\models_I \{A\}\ p(\tilde{U} : \tilde{T})\ \{B\}.$$

Tada po definiciji od $\mathcal{M}_I$ vrijedi

$$\models_I \{A\}\ c_0[\tilde{U}, \tilde{T}/\tilde{X}, \tilde{V}]\ \{B\}.$$

Iz ovog slijedi, po gornjim ograničenjima da

$$\models_I (\{\tilde{V} = \tilde{T} \wedge p[\ \tilde{X}/\tilde{U}\ ]\}\ c_0\ \{B[\ \tilde{X}/\tilde{U}\ ]\}),$$

kako se ovdje ne događaju sukobi varijabli.

c_0 ne sadrži pozive procedura, pa, po potpunosti sustava $\mathcal{H}$+*Pravilo deklaracije varijabli*

$$Tr_I \vdash_{\mathcal{C}} \{\tilde{V} = \tilde{T} \wedge A[\ \tilde{X}/\tilde{U}\ ]\}\ c_0\ \{B[\ \tilde{X}/\tilde{U}\ ]\}.$$

Po pravilu pozivanja procedura

$$Tr_I \vdash_{\mathcal{C}} \{\tilde{V} = \tilde{T} \wedge A[\ \tilde{X}/\tilde{U}\ ]\}\ p(\tilde{X} : \tilde{V})\ \{B[\ \tilde{X}/\tilde{U}\ ]\}.$$

Po pravilu zamjene parametra

$$Tr_I \vdash_{\mathcal{C}} \{\tilde{T} = \tilde{T} \wedge A\}\ p(\tilde{U} : \tilde{T})\ \{B\},$$

pa, konačno, po pravilu posljedice

$$Tr_I \vdash_{\mathcal{C}} \{A\}\ p(\tilde{U} : \tilde{T})\ \{B\}.$$

Napomena 7.7. Primijetimo da se ovdje može primijeniti pravilo zamjene parametra, jer je pretpostavka $\tilde{U} \cap F(\tilde{T}) = \emptyset$ i po nametnutim restrikcijama $\tilde{U} \cap (\tilde{X} \cup \tilde{V}) = \emptyset$, pa se nijedna varijabla iz $\tilde{U}$ ne pojavljuje slobodna u tvrdnjama iz premise pravila.

Napomena 7.8. Primijetimo da se pravila pozivanje procedure II, zamjene parametra i zamjene varijable mogu zamijeniti jednostavnijim pravilom

$$\frac{\{A\}\ c_0[\tilde{U}, \tilde{T}/\tilde{X}, \tilde{V}]\ \{B\}}{\{A\}\ p(\tilde{U} : \tilde{T})\ \{B\}},$$

pri čemu bi utemeljenost i potpunost bila očuvana. Da je ovo pravilo usvojeno, kao rezultat bi stvarni dokazi bili uglavnom duži od odgovarajućih dokaza u dokaznom sustavu $\mathcal{C}$, gdje je dovoljno samo jednom dokazati opće svojstvo $\{A\}\ c_0\ \{B\}$ tijela procedure.

7.2 Poziv rekurzivne procedure po imenu

Odgovarajući dokazni sustav za ovaj slučaj je sljedeća modifikacija sustava $\mathcal{G}$:

1. u pravilu rekurzije, p se mijenja s $p(\tilde{X} : \tilde{V})$
2. aksiom invarijance ima formu

   $$\{A\}\ p(\tilde{U} : \tilde{T})\ \{A\},$$

   gdje A nema slobodnih varijabli koje se pojavljuju slobodne u $c_0[\tilde{U}, \tilde{T}/\tilde{X}, \tilde{V}]$
3. pravilo supstitucije I se mijenja s pravilom zamjene varijable
4. pravilo supstitucije II poprima formu

   $$\frac{\{A\}\ p(\tilde{U} : \tilde{T})\ \{B\}}{\{A[\tilde{S}/\tilde{Z}]\}\ p(\tilde{U} : \tilde{T})\ \{B\}}$$

5. u pravilu konjunkcije p se mijenja s $p(\tilde{U} : \tilde{T})$
6. dodaje se pravilo zamjene parametra.

Sada želimo definirati značenje programa. Budući da procedure sada imaju parametre, moramo biti oprezni kako bi pravilno obradili odgovarajuće zamjene

stvarnih parametara u odgovarajućem redoslijedu. Stoga postupamo na malo drugačiji način.

Indukcijom po n definiramo D_n - niz deklaracija procedura

$$p_n \Longleftarrow \langle \text{name}(\tilde{X} : \tilde{V} \mid c_0^{(n)}),$$

gdje je

$$c_0^{(0)} \equiv \Omega$$

i

$$c_0^{(n+1)} \equiv c_0[\, p_n/p \,].$$

Izraz oblika $c[\, p_n/p \,]$ označava rezultat zamjene identifikatora procedure p s p_n u programu c.

Definiramo značenje programa c stavljanjem

$$\mathcal{M}_I(c) = \bigcup_{n=0}^{\infty} \mathcal{M}_I(c[\, p_n/p \,]),$$

pri čemu se pretpostavlja kontekst deklaracija procedura D_n.

Posebno,

$$\mathcal{M}_I(p(\tilde{U} : \tilde{T})) = \bigcup_{n=0}^{\infty} \mathcal{M}_I(p_n(\tilde{U} : \tilde{T})).$$

Zbog ovih definicija

$$\mathcal{M}_I(c[\, p_n/p \,]) = \mathcal{M}_I(c^{[n]}),$$

gdje je

$$c^{[n]} \equiv c[\, p_n/p \,]\langle c_0^{(n)}/p_n\rangle ... \langle c_0^{(0)}/p_0\rangle.$$

Uočimo da je $c^{[n]}$ rezultat ponavljane doslovne zamjene svakog poziva procedure $p(\tilde{U} : \tilde{T})$ s $c_0[\tilde{U}, \tilde{T}/\tilde{X}, \tilde{V}]$ izvedenog od vrha koji je c do dubine n, nakon čega slijedi doslovna zamjena svakog poziva procedure s Ω.

Strogo govoreći, prethodne definicije zahtijevaju proširenje razmatrane sintakse omogućavanjem sustava $D_1, ..., D_n$ deklaracija nerekurzivnih procedura.

Dokaz utemeljenosti gornjeg sustava se sada može utvrditi na osnovu obrazloženja korištenog u odjeljku u kojem smo pokazivali utemeljenost sustava $\mathcal{G}$. Zbog utemeljenosti proširenja $\mathcal{C}$, koji se bavi sustavom nerekurzivnih procedura, dovoljno je dokazati valjanost aksioma invarijance, utemeljenost pravila supstitucije II i pravila konjunkcije u slučaju sustava deklaracija nerekurzivnih procedura i dobrotu pravila rekurzije. Naravno, u svim slučajevima mislimo na modificirane verzije aksioma i pravila dokaza. Svi su dokazi izravni.

Dokaz potpunosti analogan je dokazu potpunosti sustava $\mathcal{G}$, koji je prethodno naveden. Za najopćenitiju formulu za p sada biramo tvrdnju parcijalne korektnosti $\{\tilde{Z'} = \tilde{Z}\}\ p(\tilde{X} : \tilde{V})\ \{\tilde{B_0}\}$, gdje je $\tilde{Z'}$ niz svih varijabli koje se pojavljuju slobodne u c_0, $\tilde{Z}$ je niz nekih novih varijabli jednake duljine kao $\tilde{Z'}$ i B_0 je tvrdnja iz $Assn$ koja definira $post_I(\tilde{Z'} = \tilde{Z}, p(\tilde{X} : \tilde{V}))$. Sada se moramo pozabaviti slučajem pozivanja procedura sa stvarnim parametrima, različitim od formalnih. Ostali slučajevi su jednaki kao i prije. Argument koji je Cook koristio u dokazu potpunosti $\mathcal{C}$ pokazuje sljedeće implicitno:

Postoje dvije tvrdnje A_1, B_1 koje ovise o A, B, c_0 i $\tilde{U}, \tilde{T}$ takve da

1. pravilo dokazivanja
$$\frac{\{A\}\ p(\tilde{U} : \tilde{T})\ \{B\}}{\{A_1\}\ p(\tilde{X} : \tilde{V})\ \{B_1\}}$$
je utemeljeno u slučaju deklaracije nerekurzivne procedure

2. $\{A_1\}\ p(\tilde{X} : \tilde{V})\ \{B_1\} \vdash_{\mathcal{C}-Pravilo\ pozivanja\ procedure\ II} \{A\}\ p(\tilde{U} : \tilde{T})\ \{B\}$.

U posebnom slučaju dokaza potpunosti $\mathcal{C}$ koji smo ovdje razmotrili možemo uzeti $A_1 \equiv \tilde{V} = \tilde{T} \wedge A[\ \tilde{X}/\tilde{U}\ ]$ i $B_1 \equiv B[\tilde{X}/\tilde{U}]$. Da su uvjeti 1 i 2 zadovoljeni odmah proizlazi iz argumenta koji je ovdje prezentiran.

Zahvaljujući argumentima korištenim u odjeljku o utemeljenosti $\mathcal{G}$, gornje pravilo dokaza je utemeljeno i u slučaju deklaracije rekurzivne procedure tako-

đer. Pretpostavimo sada

$$\models_I \{A\}\ p(\tilde{U} : \tilde{T})\ \{B\}.$$

Po gornjem

$$\models_I \{A_1\}\ p(\tilde{X} : \tilde{V})\ \{B_1\}.$$

Po prvoj lemi iz odjeljka o potpunosti sustava $\mathcal{G}$ u Cook-ovom smislu, dobijamo

$$\{\tilde{Z}' = \tilde{Z}\}\ p(\tilde{X} : \tilde{V})\ \{B_0\} \vdash \{A_1\}\ p(\tilde{X} : \tilde{V})\ \{B_1\}.$$

Dakle, zbog uvjeta 2

$$\{\tilde{Z}' = \tilde{Z}\}\ p(\tilde{X} : \tilde{V})\ \{B_0\} \vdash \{A\}\ p(\tilde{U} : \tilde{T})\ \{B\},$$

što je i trebalo dokazati. Ostatak dokaza je jednak kao u odjeljku o utemeljenosti sustava $\mathcal{G}$.

Literatura

[1] Ben-Ari, M.: *Mathematical Logic for Computer Science - Third Edition* [*Department of Science Teaching, Weizmann Institute of Science, Rehovot, Israel*], Springer

[2] Bergstra, J. A. i Klop, J. W.: *Proving program inclusion using Hoare's logic* [*Theoretical Computer Science 30 (1984) 1-48, North-Holland*]

[3] Bergstra, J. A. i Tucker, J. V.: *Hoare's logic and Peano's arithmetic* [*Theoretical Computer Science 22 (1983) 265-284, North-Holland Publishing Company*]

[4] Bergstra, J. A. i Tucker, J. V.: *Expressiveness and the Completeness of Hoare's Logic* [*Journal of computer and system sciences 25, 267-284 (1982)*]

[5] Brucker, A. D. i Wolff, B.: *Extensible Universes for Object-oriented Data Models*

[6] Clarke, E. M. jr., German, S. M. i Halpern, J. Y.: *Effective Axiomatizations of Hoare Logics* [*Harvard University, Cambridge, Massachusetts*]

[7] Gordon, M. J. C.: *Mechanizing Programming Logics in Higher Order Logic* [*Cambridge*]

[8] Gries, D. i Levin, G.: *Assignment and Procedure Call Proof Rules* [*Cornell University, Lipanj 1980.*]

[9] Hoare, C. A. R.: *An Axiomatic Basis for Computer Programming* [*The Queen's University of Belfast, *Northern Ireland, Listopad 1969.*]

[10] Hoare, C. A. R.: *Procedures and parameters: An axiomatic approach*

[11] Hoare, C. A. R., Richard, C. A. i Niklaus, W.: *An axiomatic definition of the programming language Pascal* [*ETH Zürich, Research Collection, 1972.*]

[12] Ivanov, I. i Nikitchenko, M.: *Inference Rules for the Partial Floyd-Hoare Logic Based on Composition of Predicate Complement* [*Taras Shevchenko National University of Kyiv, Ukraine*]

[13] Kammerer, R. A.: *Hoare's Axiomatic Semantics* [*Computer Science Department, University of California*]

[14] Krzystof, R.: *Ten Years of Hoare's Logic: A Survey - Part I, Listopad 1981.*

[15] Krzystof, R.: *Ten Years of Hoare's Logic: A Survey - Part II: Nondeterminism, Veljača 1983.*

[16] Krzystof, R. i Ernst-Rudiger, O.: *Fifty Years of Hoare's Logic, Studeni 2019.*

[17] Manna, Z.: *Second-order mathematical theory of computation* [*Computer Science Department, Stanford University*]

[18] Mosses, P. D.: *Formal Semantics of Programming Languages-An Overview* [*Department of Computer Science University of Wales Swansea Swansea, United Kingdom, Electronic Notes in Theoretical Computer Science 148 (2006) 41–73*]

[19] Myreen, M. O. i Gordon, M. J. C.: *Hoare Logic for Realistically Modelled Machine Code* [*Computer Laboratory, University of Cambridge, Cambridge, UK*]

[20] Nikitchenko, M., Shkilniak, O. i Shkilniak, S.: *Program Logics based on algebras with the Composition of Predicate Complement*

[21] Owicki, S. S.: *Axiomatic proof techniques for parallel programs* [*Department of Computer Science, Cornell University, Srpanj 1975.*]

[22] Parkinson, M., Bornat, R. i Calcagno, C.: *Variables as Resource in Hoare Logics* [*The Computer Society, 2006*]

[23] Pratt, V. R.: *Semantical Considerations on Floyd-Hoare logic* [*MIT/LCS/TR-168, Rujan 1976.*]

[24] Pratt, V. R.: *Semantical Considerations on Floyd-Hoare logic* [*Massachusetts Institute of Technology, Cambridge, MA 02139*]

[25] Rosu, G., Ellison, C. i Schulte, W.: *Matching Logic: An Alternative to Hoare/Floyd Logic* [*Conference Paper, Lipanj 2010.*]

[26] Rosu, G. i Stefanescu, A.: *From Hoare Logic to Matching Logic Reachability* [*2012.*]

[27] Schwarz, J.: *Generic commands - a tool for partial correctness formalisms* [*Department of Artificial Intelligence, University of Edinburgh, Listopad 1975.*]

[28] Slonneger, K. i Kurtz, B. L.: *Formal Syntax and Semantics of Programming Languages - A Laboratory Based Approach,* [*Addison-Wesley Publishing Company, 1995.*]

[29] Sokolowski, S.: *Axioms for Total Correctness,* [*Acta Informatica 9*]

[30] Vuković, M.: *Matematička logika* [*Sveučilište u Zagrebu, PMF-Matematički odjel, Svibanj 2007.*]

[31] Wand, M.: *A New Incompleteness Result for Hoare's System* [*Indiana University, Bloomington, lndtana*]

[32] Winskel, G.: *The Formal Semantics of Programming Languages - An introduction* [*Foundations of Computing Series, The MIT press, Cambridge, Massachusetts, London, England. 1994.*]

[33] Wirth, N. i Hoare, C. A. R.: *A Contribution to the Development of ALGOL* [*Programming Languages*]

[34] Xu, Z., Sui, Y. i Zhang, W.: *Completeness of Hoare Logic with inputs over the Standard Model* [*Theoretical Computer Science 612 (2016) 23-28*]

[35] Xu,Z., Sui, Y. i Zhang, W.: *Completeness of Hoare Logic Relative to the Standard Model* [2017.]

TEMELJNA DOKUMENTACIJSKA KARTICA

SVEUČILIŠTE U SPLITU
PRIRODOSLOVNO MATEMATIČKI FAKULTET
ODJEL ZA MATEMATIKU

DIPLOMSKI RAD

HOAREOVA LOGIKA

Petra Livaja

Sažetak:

U ovom diplomskom radu prezentiramo Hoareovu logiku koja daje pravila za odnos između tvrdnji o vrijednostima varijabli prije i nakon izvršavanja programa napisanog u IMP-u, tj. za provjeru parcijalne korektnosti. Proučavamo probleme utemeljenosti Hoareovih pravila i potpunosti. Obzirom da ona nasljeđuju nepotpunost iskazanu Gödelovim teoremom, u nastavku promatramo potpunost u Cook-ovom smislu i pritom odvajamo nepotpunost jezika Assn od nepotpunosti uzrokovane aksiomima i pravilima izgradnje programa. Postupno gradimo složenije sustave dodavajući nova pravila te im provjeravamo potpunost u Cook-ovom smislu i utemeljenost. Osim za while programe, dodajemo pravila za procedure bez parametara (za rekurzivne i nerekurzivne procedure), deklaraciju varijabli i osvrćemo se na problem s neinicijaliziranim varijablama, subskriptne varijable i mehanizme parametara, kod kojih promatramo poziv po imenu procedure.

Ključne riječi:

IMP, provjera ispravnosti računalnog programa, aksiomatska semantika, potpunost i utemeljenost sustava dokaza, tvrdnje parcijalne korektnosti

Podatci o radu:

70 stranica, 35 literaturnih navoda, napisano na hrvatskom jeziku

Mentor:

izv. prof. dr. sc. Jurica Perić

Članovi povjerenstva:

prof. dr. sc. Milica Klaričić Bakula

doc. dr. sc. Goran Erceg

Povjerenstvo za diplomski rad je prihvatilo ovaj rad *18. prosinca 2020.*

TEMELJNA DOKUMENTACIJSKA KARTICA

UNIVERSITY OF SPLIT, FACULTY OF SCIENCE

DEPARTMENT OF MATHEMATICS

MASTER'S THESIS

HOARE'S LOGIC

Petra Livaja

Abstract:

In this thesis, we present Hoare's logic that provides rules for the relationship between assertions about the values of variables before and after the execution of a program written in IMP, i.e., to check partial correctness. We study the problems of the validity of Hoare's rules and completeness. Since they inherit the incompleteness expressed by Gödel's theorem, we observe the completeness in Cook's sense below and separate the incompleteness of the Assn language from the incompleteness caused by axioms and rules of program construction. In addition to while programs, we add rules for non-parameter procedures (for recursive and non-recursive procedures), declaration of variables, and consider the problem with uninitialized variables, subscript variables, and parameter mechanisms, in which we observe the call by the name of the procedure.

Key words:

IMP, software verification, axiomatic semantics, soundness and completeness of proof system, partial correctness assertions

Specifications:

70 pages, 35 references, written in Croatian

Mentor:

Associate Professor dr. sc. Jurica Perić

Committee:

Professor dr. sc. Milica Klaričić Bakula

Assisstant Professor dr. sc. Goran Erceg

This thesis was approved by a Thesis commettee on *18th December 2020*.