Dokaz iz semantike

Uroš Ševkušić
indeks: 2007/2024

Univerzitet u Beogradu,
Matematički fakultet

Glava 1

Denotaciona semantika jezika While

U knjizi [1] je definisan jezik While kao jezik čiji su literali logičkog tipa **bool** (jedine vrednosti su **True** i **False**) ili tipa prirodnih brojeva (skup $\mathbb{N}$ je ovde skup brojeva $0, 1, \ldots$), promenljive mogu imati samo vrednosti iz skupa $\mathbb{N}$, a jedine naredbe su dodela vrednosti promenljive ($x := a$), naredba **skip**, sekvencijalno izvršavanje dve naredbe (S_1 ; S_2), i naredbe **if-then-else** i **while**.

Postoji više načina za definisanje semantike jezika While: operativna semantika, denotaciona semantika i aksiomatska semantika. Ovde ćemo prikazati samo tabelu sa pravilima za denotacionu semantiku. Pravila denotacione semantike nalaze se na Slici 1.1, i preuzeta su iz knjige [1]. Funkcija S_{ds} je semantička funkcija koja određuje kako se stanje programa s menja kada se izvrši neka naredba (primetimo da su argumenti funkcije zapisani u stilu jezika Haskell). Podrazumeva se da su pravila za računanje aritmetičkog izraza a i logičkog izraza b već poznata i data funkcijama $\mathcal{A}$ i $\mathcal{B}$.

$$
\begin{array}{rcl}
\mathcal{S}_{\mathrm{ds}}[\![x := a]\!]s & = & s[x \mapsto \mathcal{A}[\![a]\!]s] \\
\mathcal{S}_{\mathrm{ds}}[\![\texttt{skip}]\!] & = & \mathrm{id} \\
\mathcal{S}_{\mathrm{ds}}[\![S_1 \ ; \ S_2]\!] & = & \mathcal{S}_{\mathrm{ds}}[\![S_2]\!] \circ \mathcal{S}_{\mathrm{ds}}[\![S_1]\!] \\
\mathcal{S}_{\mathrm{ds}}[\![\texttt{if } b \texttt{ then } S_1 \texttt{ else } S_2]\!] & = & \mathsf{cond}(\mathcal{B}[\![b]\!], \mathcal{S}_{\mathrm{ds}}[\![S_1]\!], \mathcal{S}_{\mathrm{ds}}[\![S_2]\!]) \\
\mathcal{S}_{\mathrm{ds}}[\![\texttt{while } b \texttt{ do } S]\!] & = & \mathsf{FIX}\ F \\
\text{where } F\ g & = & \mathsf{cond}(\mathcal{B}[\![b]\!], g \circ \mathcal{S}_{\mathrm{ds}}[\![S]\!], \mathrm{id})
\end{array}
$$

Slika 1.1: Denotaciona semantika jezika While

Ukratko, pravila su definisana na sledeći način:

- dodela promenljive: nakon izračunavanja vrednosti aritmetičkog izraza, stanje programa se menja tako da promenljiva x sadrži vrednost tog izraza;
- **skip**: stanje programa se ne menja;
- sekvencijalno izvršavanje: najpre se stanje promeni izvršavanjem prve naredbe, a onda se menja izvršavanjem druge;
- **if-then-else**: funkcija *cond* bira da li će da se izvrši naredba u **then** ili **else** grani u zavisnosti od vrednosti aritmetičkog izraza b;
- **while** se definiše kao najmanja fiksna tačka operatora F, odnosno to je najmanja (u odnosu na neki pogodno definisani poredak) funkcija g takva da je $Fg = g$.

Za detaljniju analizu ove semantike, posebno za problem nalaženja najmanje fiksne tačke, referišemo na knjigu [1].

Glava 2

Primeri dokaza ispravnosti programa

U ovoj glavi ćemo dati više primera dokazivanja ispravnosti programa korišćenjem denotacione semantike jezika While. Najpre ćemo pokazati kako se može dokazati ispravnost nekih jednostavnijih programa, a zatim ćemo pokazati dokaz ispravnosti složenije **while** petlje.

2.1 Dokaz ispravnosti naredbe dodele

Pretpostavimo da nam je dat program:

x := 1

Ovaj program se smatra ispravnim ako nakon izvršavanja iz bilo kojeg početnog stanja s, vrednost promenljive x postane 1, a ostalim promenljivama se ne promeni vrednost. Dakle, ispravnost programa se može zapisati na sledeći način: $S_{ds}[\![x := 1]\!]s = s'$, $s'(x) = 1$ i $s'(y) = s(y)$, za sve $y \neq x$. Prvi iskaz kaže da, kada izvršavamo program $x := 1$ u početnom stanju s, postoji stanje s' u kojem će taj program završiti, odnosno naredba $x := 1$ će se izvršiti. Drugi iskaz kaže da će stanje s' imati iste vrednosti sa s na promenljivama koje nisu x, a u x će njegova vrednost biti 1. Kraći zapis za ova tri iskaza je $S_{ds}[\![x := 1]\!]s = s[x \mapsto 1]$, gde $s[x \mapsto 1]$ odgovara stanju s'.

Dokaz tvrđenja $S_{ds}[\![x := 1]\!]s = s[x \mapsto 1]$ nije težak: on sledi iz prvog pravila denotacione semantike jezika While, koje kaže:

$$S_{ds}[\![x := a]\!]s = s[x \mapsto \mathcal{A}[\![a]\!]s].$$

U našem slučaju, instanca pravila je $S_{ds}[\![x := 1]\!]s = s[x \mapsto A[\![1]\!]s]$. Vrednost aritmetičkog izraza $A[\![1]\!]s$ je 1, jer a je konstanta koja ne zavisi od stanja s. Dakle, dobili smo $S_{ds}[\![x := 1]\!]s = s[x \mapsto 1]$, čime je dokaz ispravnosti završen.

2.2 Dokaz ispravnosti dve naredbe dodele

Pređimo na nešto složeniji primer: sekvencu dve naredbe dodele. Neka je program čiju ispravnost dokazujemo sledeći:

```
x := 1; y := x - 1.
```

Program je ispravan ako nakon izvršavanja iz bilo kojeg početnog stanja s, vrednost promenljive x postane 1, a vrednost promenljive y postane 0 (jer $x = 1$ nakon izvršavanja prve naredbe, pa je $x - 1 = 0$). Dakle, program je ispravan ako važi iskaz:

$$S_{ds}[\![x := 1; y := x - 1]\!]s = s[x \mapsto 1][y \mapsto 0].$$

Primetimo sekvencijalnu primenu $[\cdot]$: to je isto kao da kažemo da postoje stanja s' i s'' takva da je $s'(t) = \begin{cases} s(t), & t \neq x \\ 1, & t = x \end{cases}$ i $s''(t) = \begin{cases} s'(t), & t \neq y \\ 0, & t = y \end{cases}$, odnosno $s''(t) = \begin{cases} s(t), & t \neq x, y \\ 1, & t = x \\ 0, & t = y \end{cases}$.

Dokaz ispravnosti sekvencijalne naredbe svodi se primenu pravila semantike za sekvencijalne naredbe. U našem slučaju, dobijamo jednakost:

$$S_{ds}[\![x := 1; y := x - 1]\!]s = S_{ds}[\![y := x - 1]\!](S_{ds}[\![x := 1]\!]s).$$

Na osnovu prethodnog primera, dobijamo da je $S_{ds}[\![x := 1]\!]s = s[x \mapsto 1]$.

Dakle, potrebno je dokazati da važi:

$$S_{ds}[\![y := x - 1]\!](s[x \mapsto 1]) = s[x \mapsto 1][y \mapsto 0].$$

Ovime je zapravo problem sveden na ispravnost naredbe dodele $y := x - 1$. Možemo primeniti pravilo naredbe dodele, čime dobijamo:

$$S_{ds}[\![y := x - 1]\!](s[x \mapsto 1]) = s[x \mapsto 1][y \mapsto \mathcal{A}[\![x - 1]\!](s[x \mapsto 1])].$$

Pošto je u $s[x \mapsto 1]$ vrednost od x jednaka 1, dobijamo da je:

$$\mathcal{A}[\![x - 1]\!](s[x \mapsto 1]) = 0,$$

na osnovu čega sledi da je:

$$S_{ds}[\![y := x - 1]\!](s[x \mapsto 1]) = s[x \mapsto 1][y \mapsto 0].$$

Odavde sledi tvrđenje koje je trebalo pokazati.

2.3 Dokaz ispravnosti if-then-else naredbe

Neka je dat program:

```
if x % 2 = 0
then y := 0
else y := 1
```

Tvrdimo da će, nakon izvršavanja programa u bilo kom stanju s, promenljiva y imati vrednost 1 ako je $s(x)$ neparan broj, a vrednost 0 ako je $s(x)$ paran broj. Označimo ceo ovaj program sa S. Tvrđenje koje želimo da dokažemo je:

$$S_{ds}[\![S]\!]s = s',$$
$$s'(t) = \begin{cases} s(t), & t \neq y \\ 1, & (t = y) \wedge (s(x) \text{ je neparan}) \\ 0, & (t = y) \wedge (s(x) \text{ je paran}) \end{cases}$$

Ovde možemo upotrebiti pravilo semantike za **if-then-else** naredbu. Na našem primeru, kada razmotamo definiciju funkcije cond, dobijamo jednakost:

$$S_{ds}[\![S]\!]s = \begin{cases} S_{ds}[\![y := 0]\!]s, & \mathcal{B}[\![\text{x \% 2} = 0]\!]s = true \\ S_{ds}[\![y := 1]\!]s, & \mathcal{B}[\![\text{x \% 2} = 0]\!]s = false. \end{cases}$$

Ovime se dokaz svodi na dokaz ispravnosti naredbe dodele i izračunavanja izraza x % 2 = 0. Na osnovu prvog primera, možemo $S_{ds}[\![y := 0]\!]s$ zameniti sa $s[y \mapsto 0]$, a $S_{ds}[\![y := 1]\!]s$ sa $s[y \mapsto 1]$. Izraz $\mathcal{B}[\![\text{x \% 2}]\!]s$ se svodi na evaluaciju aritmetičkog izraza x % 2, odnosno evaluaciju izraza $a = \mathcal{A}[\![\text{x \% 2}]\!]s$, i logičkog izraza $a = 0$. Kada evaluiramo aritmetički izraz dobijamo:

$$a = \mathcal{A}[\![\text{x \% 2}]\!]s = \begin{cases} 1, & s(x) \text{ je neparan} \\ 0, & s(x) \text{ je paran.} \end{cases}$$

Odavde dobijamo:

$$\mathcal{B}[\![\text{x \% 2} = 0]\!]s = \begin{cases} true, & s(x) \text{ je paran} \\ false, & s(x) \text{ je neparan.} \end{cases}$$

Kada to ubacimo u jednakost za evaluaciju programa S, dobijamo:

$$S_{ds}[\![S]\!]s = \begin{cases} s[y \mapsto 0], & s(x) \text{ je paran} \\ s[y \mapsto 1], & s(x) \text{ je neparan.} \end{cases}$$

Izraz sa desne strane jednakosti je neko stanje s'' u koje program dospeva nakon izvršavanja. Ostaje da se pokaže da je $s'(t) = s''(t)$, za sve promenljive t. Ovo se svodi na tri slučaja:

1. $t \neq y$: tada je $s'(t) = s(t)$. Primetimo da je $s[y \mapsto 0](t) = s(t)$, kao i $s[y \mapsto 1](t) = s(t)$. Odavde je $s'(t) = s''(t)$;

2. $t = y$ i $s(x)$ je paran: u ovom slučaju je $s'(t) = 0$ i $s''(t) = s[y \mapsto 0](t) = 0$, pa je $s'(t) = s''(t)$;

3. $t = y$ i $s(x)$ je neparan: u ovom slučaju je $s'(t) = 1$ i $s''(t) = s[y \mapsto 1](t) = 1$, pa je $s'(t) = s''(t)$.

Ovime je dokazana ispravnost ovog programa.

2.4 Dokaz ispravnosti while programa

Neka je dat naredni program:

```
x := 0
while x < 3
do (x := x + 1)
```

Ovaj program računa sumu $0 + 1 + 1 + 1 = 3$. Cilj je da to dokažemo formalno upotrebom denotacione semantike. Ceo program označimo sa S, naredbu x := 0 sa S_1, a **while** naredbu sa S_2. Primećujemo prvo da je $S = S_1; S_2$. Tvrdimo da važi:

$$S_{ds}[\![S_1; S_2]\!]s = s[x \mapsto 0 + 1 + 2].$$

Najpre upotrebljavamo pravilo sekvencijalne naredbe:

$$S_{ds}[\![S_1; S_2]\!]s = S_{ds}[\![S_2]\!](S_{ds}[\![S_1]\!]s).$$

Pošto je S_1 naredba dodele, možemo izraz $S_{ds}[\![S_1]\!]s$ zameniti sa $s[x \mapsto 0]$. Tako dobijamo jednakost:

$$S_{ds}[\![S_1; S_2]\!]s = S_{ds}[\![S_2]\!](s[x \mapsto 0]).$$

S_2 je **while** naredba. Semantike **while** naredbe je definisana preko najmanje fiksne tačke operatora F, definisanog ispod:

$$Fgs' = \begin{cases} g(S_{ds}[\![x := x+1]\!]s'), & \mathcal{B}[\![x < 3]\!]s' = true, \\ s', & \mathcal{B}[\![x < 3]\!]s' = false. \end{cases}$$

Primetimo prvo da izraz $S_{ds}[\![x := x+1]\!]s$ možemo zameniti izrazom $s[x \mapsto s(x)+1]$. Takođe, možemo $\mathcal{B}[\![x < 3]\!]s = true$ zapisati kao $s(x) < 3$, a $\mathcal{B}[\![x < 3]\!]s = false$ kao $s(x) \geq 3$. Tako dobijamo jednostaviji izraz za F:

$$Fgs' = \begin{cases} g(s'[x \mapsto s'(x)+1]), & s'(x) < 3 \\ s', & s'(x) \geq 3. \end{cases}$$

Sada možemo analizirati celu **while** petlju. Kao što je pokazano u knjizi [1], najmanja fiksna tačka se može odrediti iterativnom procedurom izračunavanja $F^n \perp = F(F^{n-1} \perp)$, gde je $F^0 \perp = \perp$, a sa $\perp$ označavamo funkciju koja svakom stanju dodeljuje vrednost $undef$. Vrednost $undef$ je oznaka za nedefinisano stanje programa. Dakle, $\perp$ je program koji nema jasno definisano stanje posle izvršavanja.

Ovde je potrebno izračunati $F^n \perp$ u stanju u kojem ulazimo u petlju. Ovde je to stanje $s[x \mapsto 0]$, jer je to rezultat naredbe S_1. Izračunajmo prvo $F \perp s'$. Kada je $s'(x) < 3$, primenjujemo operator $g = \perp$ na stanje $s'[x \mapsto s(x)+1]$, ali $\perp$ za svako stanje vraća vrednost $undef$. Ako je $s(x) \geq 3$, vraća se stanje $s[x \mapsto s(x)+1]$. Dakle, dobili smo naredni izraz za $F \perp s'$:

$$F \perp s' = \begin{cases} undef, & s'(x) < 3 \\ s', & s'(x) \geq 3. \end{cases}$$

U stanju $s[x \mapsto 0]$, vrednost od $F \perp$ je $undef$. Nastavljamo dalje. Sada izračunavamo $F(F \perp)s = F^2 \perp s'$. Kada primenimo F na $(F \perp)s'$, dobijamo izraz:

$$F(F \perp)s' = \begin{cases} F \perp s'[x \mapsto s'(x)+1], & s'(x) < 3 \\ s', & s'(x) \geq 3 \end{cases}$$

Sada možemo da razmotamo definiciju od $F \perp s'$ da dobijemo:

$$F^2 \perp s' = \begin{cases} undef, & s'(x) + 1 < 3 \\ s'[x \mapsto s'(x) + 1], & s'(x) + 1 \geq 3 \\ s', & s'(x) \geq 3. \end{cases}$$

Ovo se može dalje pojednostaviti:

$$F^2 \perp s' = \begin{cases} undef, & s'(x) < 2 \\ s'[x \mapsto s'(x) + 1], & s'(x) \geq 2 \\ s', & s'(x) \geq 3. \end{cases}$$

Kada primenimo $F^2 \perp s'$ na $s[x \mapsto 0]$, takođe dobijamo $undef$. Nastavljamo dalje: izračunavamo $F^3 \perp s'$. Dobijamo jednakost:

$$F^3 \perp s' = \begin{cases} F^2 \perp s'[x \mapsto s'(x) + 1], & s'(x) < 3, \\ s', & s'(x) \geq 3. \end{cases}$$

Slično kao u prethodnoj iteraciji, možemo da razmotamo definiciju od $F^2 \perp$ da dobijemo jednostavniji izraz:

$$F^3 \perp s' = \begin{cases} undef, & s'(x) + 1 < 2 \\ (s'[x \mapsto s'(x) + 1])[x \mapsto s'[x \mapsto s'(x) + 1](x) + 1], & s'(x) + 1 = 2 \\ s'[x \mapsto s'(x) + 1], & (s'(x) + 1 \geq 3) \wedge s'(x) = 3 \\ s', & s'(x) \geq 3. \end{cases}$$

Ovaj komplikovan izraz na drugoj grani se može pojednostaviti na $s'[x \mapsto s'(x) + 2]$. Nakon pojednostavljivanja izraza na granama, dobijamo:

$$F^3 \perp s' = \begin{cases} undef, & s'(x) < 1 \\ (s'[x \mapsto s'(x) + 1])[x \mapsto s'[x \mapsto s'(x) + 1](x) + 1], & s'(x) = 1 \\ s'[x \mapsto s'(x) + 1], & s'(x) = 2 \\ s', & s'(x) \geq 3. \end{cases}$$

Ako nastavimo dalje, u sledećoj iteraciji dobijamo:

$$F^4 \perp s' = \begin{cases} undef, & s'(x) < 0, \\ s'[x \mapsto 3], & s'(x) = 0, \\ s'[x \mapsto 2], & s'(x) = 1, \\ s'[x \mapsto 1], & s'(x) = 2, \\ s', & s'(x) \geq 3, \end{cases}$$

Ovde ako ubacimo $s' = s[x \mapsto 0]$, dobijamo $s[x \mapsto 0][x \mapsto 3] = s[x \mapsto 3]$. Možemo primetiti da dalje iteracije ne utiču na slučaj $s'(x) = 0$, odavde dobijamo da je $F^n \perp (s[x \mapsto 0]) = s[x \mapsto 3]$, pa zaključujemo da važi:

$$S_{ds}[\![S_2]\!](s[x \mapsto 0]) = s[x \mapsto 3],$$

čime je tvrđenje koje smo hteli da pokažemo dokazano.

2.5 Dokaz ispravnosti složenije while petlje

U ovoj sekciji ćemo dokazati ispravnost programa za računanje sume parnih brojeva od 0 do n. Program definišemo kao niz naredbi na sledeći na sledeći način:

```
y := 0;
while x > 0
do (
     (if x % 2 = 0
     then y := y + x
     else skip) ;
     x := x - 1
)
```

Treba dokazati da, za sve $n \in \mathbb{N}$, ako ovaj program počne u stanju s u kom je $x = n$, nakon što se ovaj program izvrši, vrednost promenljive y će

biti jednaka $0 + 2 + \ldots + n$ ako je n paran broj, a ako je n neparan broj, onda će vrednost od y biti jednaka $0 + 2 + \ldots + n - 1$.

Označimo ceo program sa S i neka je početno stanje s u kojem je $x = n$. Ono što možemo na početku primetiti jeste da je ova naredba zapravo naredba sekvencijalnog izvršavanja naredbe dodele $y := 0$ i **while** naredbe. Označimo ova dva potprograma sa S_1 i S_2. Na osnovu pravila semantike, važi:

$$S_{ds}[\![S]\!]s = S_{ds}[\![S_1 \circ S_2]\!]s = S_{ds}[\![S_2]\!](S_{ds}[\![S_1]\!]s).$$

Sada možemo sa s_1 označiti vrednost $S_{ds}[\![S_1]\!]s = S_{ds}[\![y := 0]\!]s = \{x \mapsto n, y \mapsto 0\}$. U poslednjem koraku smo primenili pravilo dodele vrednosti na promenljivu y i aritmetički izraz 0. Dakle $s_1 = \{x \mapsto n, y \mapsto 0\}$. Potrebno je dalje dokazati da će, nakon izvršavanja naredbe S_2 na stanju s_1, promenljiva y dobiti vrednost $0 + \ldots + n$, ako je n paran, a inače $0 + \ldots + n - 1$, ako je n neparan broj.

Program S_2 se sastoji od jedne **while** naredbe. Označimo sve ono posle **do** u zagradama sa S_3. Na osnovu pravila semantike, definišemo operator F izrazom:

$$Fgs = \begin{cases} g(S_{ds}[\![S_3]\!]s), & s(x) > 0 \\ s, & x = 0 \end{cases}$$

Ovaj izraz važi za sve g i s.

Program S_3 se sastoji od naredbe sekvencijalnog izvršavanja **if-then-else** naredbe i naredbe dodelje promenljive. Na osnovu tog zapažanja i pravila semantike, dobijamo naredni izraz:

$$S_{ds}[\![S_3]\!]s = \begin{cases} s[y \mapsto s(y) + s(x)][x \mapsto s(x) - 1], & s(x) \equiv_2 0 \\ s[x \mapsto s(x) - 1], & s(x) \equiv_2 1 \end{cases}$$

Ovaj izraz važi za sve s. Dakle, ako je vrednost promenljive x paran broj, onda se izvršava **then** grana i menja se vrednost promenljive y, pa se zatim izvršava naredba dodele $x := x - 1$ i menja se vrednost promenljive x; ako je vrednost promenljive x neparan broj, onda se izvršava **else** grana (ona

je **skip**, pa se ne menja stanje programa), nakon čega se izvršava naredba dodele $x := x - 1$ i menja se vrednost promenljive x.

Sada možemo analizirati izvršavanje **while** naredbe. Kao što smo ranije spomenuli, ovo se svodi na izračunavanje izraza $F^n \perp$. U našem slučaju, potrebno je iterativno primenjivati funkciju F sa argumentima $g = \perp$ i $s = s_1$. Dobijamo sledeći niz jednakosti:

$$F^0 \perp s_1 = undef,$$

$$F^1 \perp s_1 = \begin{cases} undef, & s(x) > 0 \\ s_1, & s(x) = 0 \end{cases}$$

$$F^2 \perp s_1 = \begin{cases} F^1 \perp (s_1[y \mapsto s_1(y) + s_1(x)][x \mapsto s_1(x) - 1]), & s_1(x) \equiv_2 0 \wedge s_1(x) > 0 \\ F^1 \perp (s_1[x \mapsto s_1(x) - 1]), & s_1(x) \equiv_2 1 \wedge s_1(x) > 0 \\ s_1, & s_1(x) = 0 \end{cases}$$

Sada ćemo analizirati svaku od grana u $F^2 \perp s_1$. Primetimo da je $F_1 \perp s$ jednako $undef$ ako i samo ako je $s(x) > 0$. U prvoj grani, stanje od s_1 se menja tako da se x smanji. Međutim, iz uslova $s_1(x) \equiv_2 0$ i $s_1(x) > 0$, zaključujemo da je $s_1(x) \geq 2$, pa će i posle smanjivanja vrednost promenljive x biti veća od 0, zbog čega se u prvoj grani vraća vrednost $undef$. U drugoj grani, ako je $s_1(x) = 1$, onda će nakon smanjivanja vrednost promenljive x biti 0 i zbog toga je u ovom slučaju $F^2 \perp s_1 = s_1[x \mapsto 0]$. U suprotnom, ako je $s_1(x) \equiv_2 1$ i $s_1(x) > 1$, rezultat će biti $undef$. U trećoj grani se samo prosleđuje vrednost od s_1. Dakle, dobijamo jednakost:

$$F^2 \perp s_1 \begin{cases} undef, & s_1(x) \geq 2, \\ s_1[x \mapsto 0], & s_1(x) = 1 \\ s_1, & s_1(x) = 0 \end{cases}$$

Nastavljamo dalje sa iteracijom: računamo $F^3 \perp s_1$. Sličnom analizom kao za slučaj $F^2 \perp s_1$, dobijamo jednakost:

$$F^3 \perp s_1 = \begin{cases} undef, & s_1(x) > 2 \\ s_1[y \mapsto 0 + 2][x \mapsto 1][x \mapsto 0], & s_1(x) = 2 \\ s_1[x \mapsto 0], & s_1(x) = 1 \\ s_1, & s_1(x) = 0 \end{cases}$$

Izraz na drugoj grani se može pojednostaviti: $s_1[y \mapsto 0+2][x \mapsto 1][x \mapsto 0] = s_1[y \mapsto 0+2][x \mapsto 0]$.

Na sličan način se dobija izraz i za $F_3 \perp s_1$:

$$F^4 \perp s_1 = \begin{cases} undef, & s_1(x) > 3 \\ s_1[x \mapsto 2][y \mapsto 0+2][x \mapsto 0], & s_1(x) = 3 \\ s_1[y \mapsto 0+2][x \mapsto 1][x \mapsto 0], & s_1(x) = 2 \\ s_1[x \mapsto 0], & s_1(x) = 1 \\ s_1, & s_1(x) = 0 \end{cases}$$

Moguće je drugu granu pojednostaviti na $s_1[y \mapsto 0+2][x \mapsto 0]$ i treću granu pojednostaviti na $s_1[y \mapsto 0+2][x \mapsto 0]$.

Ovde već možemo uočiti pravilo. Ako nastavimo istu proceduru, indukcijom možemo zaključiti sledeće:

$$F^n \perp s_1 = \begin{cases} undef, & s_1(x) > n-1 \\ s_1[y \mapsto 0+2+\ldots+2k][x \mapsto 0], & s_1(x) = 2k \wedge 2k \leq n \\ s_1[y \mapsto 0+2+\ldots+2k][x \mapsto 0], & s_1(x) = 2k+1 \wedge 2k+1 < n \end{cases}$$

Odavde, kada pustimo n da teži ∞, dobijamo izraz za $FIX \perp s_1$:

$$FIXF \perp s_1 = \begin{cases} s_1[y \mapsto 0+2+\ldots+2k][x \mapsto 0], & s_1(x) = 2k \\ s_1[y \mapsto 0+2+\ldots+2k][x \mapsto 0], & s_1(x) = 2k+1 \end{cases}$$

Sada možemo ovo iskoristiti tako što primetimo da je $S_{ds}[\![S]\!]s = S_{ds}[\![S_2]\!]s_1 = FIXF \perp s_1$. Dakle, kada je $s(x) = s_1(x) = n = 2k+1$, onda je $(S_{ds}[\![S]\!]s)(y) = 0+2+\ldots+(n-1)$; kada je $s(x) = s_1(x) = n = 2k$, onda je $(S_{ds}[\![S]\!]s)(y) = 0+2+\ldots+n$. Ovime je dokazana ispravnost naredbe S.

Bibliografija

[1] Hanne Riis Nielson and Flemming Nielson. *Semantics with Applications: An Appetizer (Undergraduate Topics in Computer Science)*. Springer-Verlag, Berlin, Heidelberg, 2007.