FAKULTA MATEMATIKY, FYZIKY
A INFORMATIKY
UNIVERZITY KOMENSKÉHO
KATEDRA INFORMATIKY

Matematická teória programovania

Zbierka riešených príkladov

(diplomová práca)

ONDREJ JOMBÍK

release 0.9.9 build 2007-08-11

Vedúci: RNDr. Dušan Guller, PhD.

Bratislava, 2003–2007

Čestne vyhlasujem, že predkladanú diplomovú prácu som vypracoval samostatne s využitím uvedenej literatúry.

Ondrej Jombík

Ďakujem môjmu diplomovému vedúcemu RNDr. Dušanovi Gullerovi, PhD. za cenné rady a odbornú pomoc pri písaní diplomovej práce.

Ďalej chcem poďakovať mojim rodičom, priateľom a spolužiakom za ich veľmi silnú podporu, trpezlivosť a toleranciu.

Moje osobitné poďakovanie patrí Mgr. Ľubomírovi Hostovi za vytvorenie skvelého pracovného a zostavovacieho rámca pre prácu so systémami LaTeX a pdfTeX.

Abstrakt

Ondrej Jombík: *Matematická teória programovania. Zbierka riešených úloh.* Diplomová práca. Univerzita Komenského. Fakulta matematiky, fyziky a informatiky.

Školiteľ: RNDr. Dušan Guller, PhD.
Obhajoba: Bratislava, august 2007

V našej práci sme sa snažili zozbierať reprezentatívnu vzorku úloh tématicky patriacu k základným okruhom matematickej teórie programovania, ako sú schémy, programy, interpretácie, (ne)rozhodnuteľnosť vlastností, správnosť programov, dokazovanie správnosti, sémantika programov a pod. Medzi úlohami sú zastúpené rôzne stupne obtiažnosti od ľahkých, cez stredne ťažké až po náročné úlohy. K všetkým úlohám sme poskytli stručné návody a taktiež vzorové riešenia. Okrem praktických úloh zbierka obsahuje vždy aj prehľad najdôležitejších definícií a tvrdení v danej oblasti. Tvrdenia sú uvádzané bez dôkazov.

Kľúčové slová: matematická teória programovania, základy teórie programovania, cvičenia, príklady, úlohy, riešenia, zbierka úloh, ZTP.

Predhovor

Prvýkrát som sa s teóriou programovania stretol ako študent druhého ročníka informatiky na FMFI UK. Vtedy sa škola ešte volala Matematicko fyzikálna fakulta. Od starších spolužiakov som sa dopočul, že je to vraj ťažký predmet. Po jeho absolvovaní s nimi môžem súhlasiť, ale len čiastočne. Ľahký určite nebol, no aj napriek tomu sa mi stal v porovnaní s ostatnými predmetmi spadajúcimi do nášho odboru bližší.

Matematická teória programovania prednášaná ako predmet Základy teórie programovania pánom doktorom RNDr. Igorom Prívarom, CSc. ma zaskočila nepripraveného. Od základnej školy som vedel programovať a na strednej škole som si túto zručnosť výrazne zlepšil. Veľmi ma to bavilo, problematika ma zaujímala, ale absentovali mi teoretické znalosti. Dnes už viem ľahko posúdiť, že moj vtedajší spôsob uvažovania o programovaní bol veľmi povrchný a nekomplexný. Samozrejme s takýmto štýlom sa predmet úspešne dokončiť nedal. A tak som ho musel opakovať. Pre iných študentov by toto predstavovalo len ďalšiu obligátnu povinnosť, ktorú treba mať čo najrýchlejšie z krku. U mňa to však bolo zdrojom veľkej motivácie. Sám pre seba som si povedal, že chcem do toho vidieť. Keď nám navyše náš vtedajší cvičiaci Mgr. Rastislav Graus položartom povedal, že v tomto predmete správne riešenia neexistujú a vraj sú len správne zadania, začal som s prepisovaním cvičených príkladov do LaTeXového dokumentu, ktorý mal nám študentom s tými správnymi riešeniami pomôcť.

Od roku 2003 sa teda začali rodiť základy mojej terajšej diplomovej práce. Pôvodný cieľ samozrejme nebol až taký smelý, chcel som len spísať riešenia pre známe príklady z cvičení Teórie programovania. Postupne som v zbierke vychytával chyby, dopĺňal príklady, vylepšoval riešenia. Ako moje vedomosti a schopnosti rástli, zväčšovala sa aj kvalita samotného materiálu. Predmet som napokon úspešne absolvoval a na samotnej skúške sa mi už ani nezdal taký ťažký.

Teoretické poznatky a prístupy ktoré mi matematická teória programovania odovzdala do mojho programátorského života sú k nezaplateniu. Okrem toho, že som si značne rozšíril okruh svojich vedomostí, som si uvedomil aj fakt, že sa signifikantne zmenil spôsob mojho uvažovania. Nie som viac obyčajný programátor, ale architekt. Kód, ktorý vytvorím, je menej náchylný na chyby. Viac sa zamýšlam nad kritickými stavmi programu, analyzujem dosiahnuteľnosť výsledkov, skúmam časovú zložitosť algoritmov. Vďaka poznatkom z oblasti sémantiky viem podrobne skúmať úseky programov a v krátkom čase o nich povedať vlastnosti, ku ktorým by som inak dospel len dlhou analýzou. Jednoducho – vidím veci, ktoré som kedysi nevidel.

Zbierku som sa snažil koncipovať tak, aby pomáhala študentom všetkých kategórií. Príklady sú formulované jasne a zrozumiteľne, aby nebolo žiadnych pochýb o tom, čo je úlohou študenta. Riešenia sú podrobné, niekedy trošku dlhé, ale snažia sa dosiahnuť kompletné pochopenie daného problému u riešiteľa. Lepší študenti môžu čítať rýchlejšie a preskakovať jednoduché pasáže, tí zvyšní majú všetko podrobne vysvetlené. K niektorým príkladom sú uvedené aj viaceré riešenia, čo má ukázať široké možnosti matematickej teórie programovania.

Odozvy mojich mladších spolužiakov z nižších ročníkov ma prekvapili. Zbierku vítajú a vnímajú ju ako užitočnú pomôcku pri štúdiu. Som tomu veľmi rád, pretože sa mi týmyo splnil môj sen, ktorý som mal od začiatku štúdia na vysokej škole. Aby moja diplomová práca nebola užitočná len mne, ale aby pomáhala aj ďalším ľuďom a neostala zapadnutá prachom v školskom archíve.

Obsah

Kapitola 1

Programové schémy

1.1 Štandardné programové schémy

Definícia 1 (Symboly) Symbolmi sú individuálne premenné, funkčné symboly, predikátové symboly a špeciálne symboly:

- Individuálne premenné sú $X = \{x, y, z, \ldots\}$
  - vstupné premenné sú $X_x - \overline{x} = \{x_1, \ldots, x_k\}$
  - výstupné premenné sú $X_z - \overline{z} = \{z_1, \ldots, z_m\}$
  - pracovné premenné sú $X_y - \overline{y} = \{y_1, \ldots, y_n\}$
- Funkčné symboly sú $F = \bigcup_{i=0}^{\infty} F^i$
  - pre $f \in F^n$ platí $arity(f) = n$
  - $F^n = \{f, g, h, \ldots\}$ sú n-árne symboly
  - $F^0 = \{a, b, \ldots\}$ sú konštanty
- Predikátové symboly sú $B = \bigcup_{i=0}^{\infty} B^i$
  - pre $p \in B^n$ platí $arity(p) = n$
  - $B^0 = \{0, 1\}$ sú logické konštantnty $true$ a $false$
- Špeciálne symboly sú $[,], (,), :=, :$

Definícia 2 (Výrazy a príkazy) Základnými syntaktickými objektami sú výrazy a predikáty (tzv. booleovské výrazy). Z nich sa konštruujú príkazy.

- Termy sú $T = t, t_i, \ldots$
  1. $F^0 \subseteq T, X \subseteq T$
  2. ak $f \in F^n, t_1, \ldots, t_n \in T$ potom $f(t_1, \ldots, t_n) \in T$
- Predikáty sú $P = \{q\}$
  1. $B^0 \subseteq P$
  2. ak $p \in B^n, t_1, \ldots, t_n \in T$ potom $(p(t_1, \ldots, t_n) \in P$
- Príkazy sú $C = \{st, st_i, \ldots\}$
  1. počiatočný príkaz je **begin** $[\overline{y}] := [t_1(\overline{(x)}, \ldots, t_n(\overline{x})]$
  2. koncový príkaz je **end** $[\overline{z}] := [t_1(\overline{x}, \overline{y}), \ldots, t_n(\overline{x}, \overline{y})]$
  3. priraďovací príkaz je $[\overline{y}] := [t_1(\overline{x}, \overline{y}), \ldots, t_n(\overline{x}, \overline{y})]$
  4. príkaz skoku je **goto** i
  5. podmienkový príkaz je **if** $p(\overline{x}, \overline{y})$ **then** st, kde st je priradenie alebo príkaz skoku
  6. príkaz s návestím je $i : st$, kde st je buď priradenie, podmienka alebo skok

Poznámka 1 Definícia syntaxe podmienkového príkazu určuje, že podmienku reprezentuje predikátový symbol. Takže podmienkou nemôže byť formula vytvorená pomocou logických spojok resp. operácií not, and, or, pretože tým by sme do schém zaviedli niektoré fixne interpretované symboly.

Definícia 3 (Štandardná programová schéma) Štandardnú programovú schému tvorí konečná postupnosť príkazov $S = \{st_0, st_1, \ldots, st_n, st_{n+1}\}$, kde

1. st_0 je počiatočný príkaz
2. st_{n+1} je koncový príkaz
3. st_1 $(1 \leq i \leq n)$ je podminekový, priraďovací alebo skokový príkaz s návestím i
4. skok mieri na návestie z intervalu $\langle 1, n \rangle$, alebo na návestie **end**

Poznámka 2 Programová schéma nemusí obsahovať koncový príkaz **end**, keď podľa terminológie teórie grafov neexistuje žiadna hrana, ktorá do neho vedie.

Definícia 4 (Interpretácia štandardnej programovej schémy) Interpretáciou programovej schémy S nazývame dvojicu $I = (D, i)$, kde

- D je obor interpretácie
- i je interpretačný morfizmus
  - $\forall a \in F^0 : i(a) \in D$
  - $\forall f \in F^n : i(f) \in D^n \mapsto D$
  - $\forall c \in B^0 : i(c) \in \{0, 1\}$
  - $\forall p \in B^n : i(p) \in D^n \mapsto \{0, 1\}$

Pojem interpretácie sa dá primočiaro rozšíriť aj na termy a predikáty (k premenných v terme):

- $i(t) \in D^k \mapsto D$
- $i(q) \in D^k \mapsto \{0, 1\}$

Definícia 5 (Program) Programom P nazývame dvojicu $P = (S, I)$, kde S je programová schéma a I je interpretácia programovej schémy.

Príklad 1 Máme program P_1. Zistite, čo program počíta a napíšte jeho schému.

$$
\begin{array}{rll}
P_1: & \textbf{begin} & [y_1, y_2] := [0, 0] \\
& 1: & \textbf{if } y_2 \geq x \textbf{ then goto end} \\
& 2: & [y_1, y_2] := [y_1 + 1, (y_1 + 1)^2] \\
& 3: & \textbf{goto } 1 \\
& \textbf{end} & [z] := [y_1]
\end{array}
$$

Riešenie 1 Po krátkej analýze je zrejmé, že program počíta $\lceil \sqrt{x} \rceil$ (hornú celú časť odmocniny x).

Schéma S, abstrakcia programu vzhľadom na riadiace štruktúry, vyzerá takto:

$$
\begin{array}{rll}
S: & \textbf{begin} & [y_1, y_2] := [a_1, a_2] \\
& 1: & \textbf{if } p(y_2, x) \textbf{ then goto end} \\
& 2: & [y_1, y_2] := [f_1(y_1), f_2(y_1)] \\
& 3: & \textbf{goto } 1 \\
& \textbf{end} & [z] := [y_1]
\end{array}
$$

Príklad 2 Napíšte interpretáciu I_1 tak, aby sme pomocou nej a predchádzajúcej schémy S dostali pôvodný program P_1, tj. aby platilo $P_1 = (S,\ I_1)$.

Riešenie 2 Interpretácia $I_1 = (D_1, i_1)$:

$$\begin{array}{rrcl} I_1: & i_1(p(y,x)) & = & y \geq x \\ & i_1(f_1(y)) & = & y+1 \\ & i_1(f_2(y)) & = & (y+1)^2 \\ & i_1(a_1) & = & 0 \\ & i_1(a_2) & = & 0 \\ & D_1 & = & \mathbb{N} \end{array}$$

Príklad 3 Nájdite interpretáciu I_2, ktorá spolu s predchádzajúcou schémou S vytvorí program, ktorý bude počítať $\lceil log_2 x \rceil$.

Riešenie 3 Interpretácia $I_2 = (D_2, i_2)$:

$$\begin{array}{rrcl} I_2: & i_2(p(y,x)) & = & y \geq x \\ & i_2(f_1(y)) & = & y+1 \\ & i_2(f_2(y)) & = & 2^{y+1} \\ & i_2(a_1) & = & 0 \\ & i_2(a_2) & = & 1 \\ & D_2 & = & \mathbb{N} \end{array}$$

Pomocou príkladu sme si ukázali, že nad jednou schémou S môžu byť postavené viaceré programy $(S, I_1), (S, I_2) \ldots (S, I_n)$ riešiace odlišné úlohy.

Príklad 4 Je daná programová schéma S.

$$\begin{array}{rrl} S: & \textbf{begin} & [y_1, y_2] := [x, a] \\ & 1: & \textbf{if } p(y_1) \textbf{ then goto end} \\ & 2: & [y_1, y_2] := [f(y_1), g(y_1, y_2)] \\ & 3: & \textbf{goto } 1 \\ & \textbf{end} & [z] := [y_2] \end{array}$$

Nájdite interpretácie:

- I_1 takú, že program (S, I_1) bude počítať $x!$ (faktoriál x)
- I_2 takú, že program (S, I_2) bude počítať $\sum_{i=1}^{x} i$ (suma od 1 po x)

Riešenie 4 Hľadané interpretácie sú zobrazené v nasledujúcich tabuľkách.

I_1	$\mathbb{N}$
a	1
p	$y_1 = 0$
f	$y_1 - 1$
g	$y_1 y_2$

I_2	$\mathbb{N}$
a	0
p	$y_1 = 0$
f	$y_1 - 1$
g	$y_1 + y_2$

Príklad 5 Napíšte históriu výpočtu pre program $P_2 = (S,\ I_2)$ s hodnotou vstupnej premennej $x = 7$. Formálne sa ohodnotenie vstupných premenných zapisuje ako $v[x \leftarrow 7]$.

Riešenie 5 Je nutné si uvedomiť, že históriu výpočtu môžeme zapísať vzhľadom na stavy výpočtu, ale taktiež vzhľadom na konfigurácie. V našom riešení použijeme druhú možnosť, tj. históriu výpočtu vzhľadom na konfigurácie.

$$
\begin{array}{l}
[1,1]_{begin} \\
[1,1]_1 \\
[2,4]_2 \\
[2,4]_3 \\
[2,4]_1 \\
[3,9]_2 \\
[3,9]_3 \\
[3,9]_1 \\
[3]_{end}
\end{array}
$$

Definícia 6 (Herbrandovo univerzum) Herbrandovo univerzum $\mathcal{H}$ pozostáva z reťazcov symbolov, zostrojených zo vstupných premenných a funkčných symbolov:

- ak $x_i \in X_x$ potom "x_i" $\in \mathcal{H}$
- ak $a \in F^0$ potom ä" $\in \mathcal{H}$

- ak $f \in F^n$;"t_1", ..., "t_n" $\in \mathcal{H}$ potom "$f(t_1, \ldots, t_n)$" $\in \mathcal{H}$

Definícia 7 (Herbrandova voľná interpretácia) Herbrandovou interpretáciou štandardnej programovej schémy S nazývame dvojicu $I_H = (\mathcal{H}, i_H)$, kde:

- $\mathcal{H}$ je Herbrandovo univerzum
- i_H je interpretačný morfizumus:
  - $\forall x \in X_x : i_H(x) =$ "x"
  - $\forall a \in F^0 : i_H(a) =$ "a"
  - $\forall f \in F^n : i_H(f) \in \mathcal{H}^n \mapsto \mathcal{H}$ n-tici "t_1", ..., "t_n" priradí "$f(t_1, \ldots, t_n)$"
- Interpretácia premenných a funkčných symbolov je "pevná", voľná je interpretácia predikátov.
- Existuje nekonečne veľa Herbrandových interpretácií danej netriviálnej schémy, odlišujúcich sa interpretáciou predikátových symbolov.
- Hebrandove interpretácie "zastupujú"triedu všetkých interpretácií.

Definícia 8 (Zladenie výpočtov a symbolických výpočtov) Interpretácia I s ohodnotením v je zladená s voľnou interpretáciou I_H s ohodnotením predikátov práve vtedy, keď pre každé $p \in B^n$ platí

$$i_H(p)("t_1", \ldots, "t_n") \iff i(p)(i^v(t_1), \ldots, i^v(t_n))$$

Príklad 6 Zostrojte Herbrandové univerzum pre predchádzajúcu schému S.

Riešenie 6 Herbrandovo univerzum je množina reťazcov symbolov zostrojených zo vstupných premenných a funkčných symbolov.

Riešenie začneme tým, že zoberieme všetky vstupné premenné schémy (v našom prípade vstupnú premennú x) a všetky použité konštanty (v našom prípade a_1 a a_2).

$$"x", "a_1", "a_2"$$

Ďalej zoberieme funkcie f_1 a f_2 a aplikujeme na všetky dostupné termy, ktoré doteraz reprezentujú konštanty a_1, a_2 a vstupná premenná x.

$$"f_1(x)", "f_1(a_1)", "f_1(a_2)"$$
$$"f_2(x)", "f_2(a_1)", "f_2(a_2)"$$

Týmto krokom sa nám teraz rozšírila množina termov, takže uvedeným spôsobom aplikovania funkcií f_1 a f_2 na termy pokračujeme ďalej.

$$"f_1(f_1(x))", "f_1(f_1(a_1))", "f_1(f_1(a_2))"$$
$$"f_1(f_2(x))", "f_1(f_2(a_1))", \ldots$$
$$"f_2(f_1(x))", "f_2(f_1(a_1))", \ldots$$
$$"f_2(f_2(x))", \ldots$$

Takto je možné pokračovať do nekonečna. Uvedeným postupom sa teda dá zostaviť množina reťazcov Herbrandového univerza vzhľadom na príslušnú schému. V našom prípade je táto množina spojená so schémou S.

Keď bližšie preskúmame schému S, zistíme, že napríklad term $"f_1(f_2(a_2))"$ nemôže nikdy počas behu výpočtu vzniknúť. Napriek tomu sa však v množine Herbrandového univerza nachádza.

Príklad 7 Schéma S sa zastaví práve vtedy, keď sa zastaví výpočet pre každú Herbrandovú interpretáciu schémy S.

Riešenie 7

$\underline{\Longrightarrow}$: Z definície vieme, že schéma S sa zastaví, ak pre každú interpretáciu I sa zastaví program $(S,\ I)$. Program $P = (S,\ I)$ sa zastaví, ak pre každé ohodnotenie v vstupných premenných $\overline{x}$ je hodnota $val(S,\ I,\ v)$ definovaná. Keďže sme predpokladali, že schéma S sa zastaví, tj. všetky výpočty na nej sa zastavia, zastavia sa aj všetky výpočty s Herbrandovými interpretáciami.

$\underline{\Longleftarrow}$: Musíme dokázať, že výpočet sa zastaví pre ľubovoľnú interpretáciu I a ľubovolné ohodnotenie v vstupných premenných $\overline{x}$. Ku každej dvojici I a v existuje s nimi zladená Herbrandová interpretácia I_H. Z predpokladu sa ale výpočet pre I_H zastaví, takže sa musí zastaviť aj výpočet (S, I, v). Ak sa zastavia všetky výpočty na schéme S, zastavia sa aj všetky programy $(S,\ I)$. Ak sa zastavia všetky programy, potom sa aj schéma S zastaví.

1.2 Nerozhodnuteľnosť vlastností schém

Definícia 9 (Rozhodnuteľnosť) Problém je *rozhodnuteľný* ak existuje všeobecný algoritmus (musí sa zastaviť), ktorý pre ľubovoľnú schému z danej triedy schém odpovie, či schéma spĺňa danú vlastnosť alebo nespĺňa.

Problém je *nerozhodnuteľný*, ak takýto všeobecný algoritmus pre danú vlastnosť neexistuje.

Definícia 10 (Čiastočná rozhodnuteľnosť) Problém je *čiastočne rozhodnuteľný* ak existuje procedúra (nemusí sa zastaviť), ktorá pre ľubovoľnú schému z danej triedy schém odpovie kladne, keď schéma spĺňa danú vlastnosť a odpovie záporne alebo sa neskončí, keď schéma danú vlasnosť nespĺňa.

Poznámka 3 Pre triedu štandardných programových schém $\mathcal{S}$ sa skúmajú štyri hlavné rozhodovacie problémy:

- problém zastavenia
- problém divergencie
- problém ekvivalencie
- problém izomorfizmu

Príklad 8 Máme danú schému S.

$$
\begin{array}{rrl}
S: & \textbf{begin} & [y_1, y_2] := [a, a] \\
& 1: & \textbf{if } p(y_1) \textbf{ then goto end} \\
& 2: & [y_1] := [f(y_1)] \\
& 3: & \textbf{if } p(y_1) \textbf{ then goto end} \\
& 4: & [y_1, y_2] := [f(y_1), f(y_1)] \\
& 5: & \textbf{if } p(y_1) \textbf{ then goto } 7 \\
& 6: & \textbf{goto end} \\
& 7: & \textbf{if } p(y_2) \textbf{ then goto } 4 \\
& 8: & \textbf{goto } 2 \\
& \textbf{end} & [z] := [a]
\end{array}
$$

Nájdite interpretáciu I_1 takú, že program $P_1 = (S, I_1)$ diverguje.

Riešenie 8 Aby program divergoval, potrebujeme vytvoriť večný cyklus, čiže zabezpečiť beh programu bez dosiahnutia príkazu na návestí **end**. Smer behu programu ovplyvňujú predikáty. Pre náš zámer bude vhodné, ak predikát $p(x)$ bude dávať napríklad takéto výsledky.

$$\begin{aligned} p(a) &= false \\ p(f(a)) &= false \\ p(f(f(a))) &= true \end{aligned}$$

Vždy po vykonaní príkazu na riadku 4 obsahujú premenné y_1 a y_2 rovnaké hodnoty. Preto ak výsledok testu na riadku 5 bude *true*, potom bude *true* aj výsledok testu na riadku 7. Pre večný cyklus teda stačí, aby ešte platila nasledujúca podmienka.

$$\forall n \geq 2 : \ p(f^n(a)) = true$$

Hľadaná interpretácia $I_1 = (D_1, \ i_1)$ môže potom vyzerať takto:

$$\begin{aligned} I_1 : \quad i_1(p(x)) &= x \geq 2 \\ i_1(f(x)) &= x + 1 \\ i_1(a) &= 0 \\ D_1 &= \mathbb{N} \end{aligned}$$

Príklad 9 Dokážte alebo vyvráťte tvrdenie, že pre schému S z predchádzajúceho príkladu, pre každú konečnú doménu D_2 a pre každý interpretačný morfizmus i platí, že program $P_2 = (S, \ (D_2, \ i_2))$ sa zastaví.

Riešenie 9 Tvrdenie neplatí. Ak na konečnej doméne $D_2 = \{0, \ 1, \ 2\}$ upravíme funkciu f tak, že bude vstupný parameter inkrementovať najnajvýš po hodnotu 2, dostávame dokonca divergentný program P_2.

$$\begin{aligned} i_2(p(x)) &= x \geq 2 \\ i_2(f(x)) &= min(x + 1, 2) \\ i_2(a) &= 0 \end{aligned}$$

Príklad 10 Rozhodnite, či je problém dosiahnuteľnosti príkazu v štandardnej schéme rozhodnuteľný. Svoje tvrdenie dokážte.

Riešenie 10 Problém je čiastočne rozhodnuteľný.

1. Nie je (úplne) rozhodnuteľný.

   Sporom. Predpokladajme, že je problém rozhodnuteľný. Potom vieme rozhodnúť aj to, či je dosiahnuteľný príkaz **end**. Takto by sme ale vedeli rozhodovať divergenciu schémy a to nasledovným spôsobom:

   – ak je **end** dosiahnuteľný, tak schéma nie je divergentná,
   – ak **end** dosiahnuteľný nie je, tak schéma je divergentná.

2. Je čiastočne rozhodnuteľný.

   Zostrojíme procedúru, ktorá povie, že je príkaz dosiahnuteľný ak je príkaz dosiahnuteľný. Ak je príkaz nedosiahnuteľný procedúra povie, že je príkaz nedosiahnuteľný alebo nepovie nič (bude bežať donekonečna).

   Procedúra simuluje výpočty programov na schéme reprezentovanej stromom. Pri predikátoch existujú dve hrany, inde je hrana len jedna. Čiže vrcholmi stromu s dvomi hranami sú miesta v schéme, kde sa testujú predikáty (**if** ... **then** ...). Koreňom stromu je návestie **begin**.

   Strom prehľadávame do šírky, tj. po rozdelení sledujeme všetky vetvy stromu súčasne. Výpočet sa rozdelí na n ciest, ale n je vždy konečné číslo. Koniec behu procedúry môže nastať v dvoch možných prípadoch.

   a. V niektorej z ciest prídeme na príkaz, ktorého dosiahnuteľnosť sme chceli zistiť. Odpovieme, že príkaz je dosiahnuteľný (aspoň jednou interpretáciou a vstupným ohodnotením).
   b. Všetkých n ciest sa dostane do príkazu **end** pričom ani jedna neprešla hľadaným príkazom. Odpovieme, že príkaz nie je dosiahnuteľný ani jednou interpretáciou s ľubovoľnými vstupnými hodnotami.

   V prípade, že daný príkaz nie je dosiahnuteľný a v programe sa vyskytuje cyklus, naša procedúra nikdy neskončí.

Definícia 11 (Syntakticky ohraničené podtriedy schém)

- *Stromové schémy* sú také, ktorých graf neobsahuje cyklus.
- *Janovove schémy* sú také, ktoré obsahujú monadické symboly nad jedinou pracovnou premennou.
- *Konzervatívne schémy* sú také, v ktorých sa premenná z ľavej strany priradenia vyskytuje aj na pravej strane priraďovacieho príkazu.

- *Progresívne schémy* sú také, v ktorých premenná z ľavej strany priraďovacieho príkazu je použitá na pravej strane nasledujúceho príkazu priradenia.

Definícia 12 (Sémanticky ohraničené podtriedy schém)

- *Liberálne schémy* sú také, kde sa pri ľubovoľnej interpretácii I_H ten istý výraz počíta nanajvýš jeden krát.
- *Voľné schémy* sú také, keď pre každú cestu vedúcu z počiatočného príkazu existujú interpretácia a ohodnotenie vstupných premenných také, že výpočet sleduje túto cestu.
- *Dosiahnuteľné schémy* sú také, ktoré obsahujú len dosiahnuteľné príkazy, tj. pre každý píkaz v schéme existuje interpretácia, pri ktorej sa príkaz vykoná.
- *Priechodné schémy* sú také, ktoré obsahujú len dosiahnuteľné hrany, tj. pre každú hranu v schéme existuje interpretácia, pri ktorej sa hranou prejde.

Príklad 11 Uvažujme triedu dosiahnuteľných schém $\mathcal{D}$. Je príslušnosť schémy k triede $\mathcal{D}$ rozhodnuteľný problém? Svoje tvrdenie zdôvodnite.

Riešenie 11 Problém je čiastočne rozhodnuteľný.

Z definície vieme, že dosiahnuteľná schéma obsahuje iba dosiahnuteľné príkazy. Pre každý príkaz v dosiahnuteľnej schéme existuje interpretácia, pri ktorej sa príkaz vykoná.

V predchádzajúcom príklade sme dokázali, že problém dosiahnuteľnosti príkazu je čiastočne rozhodnuteľný. V našej procedúre, ktorá bude skúmať príslušnosť schémy k triede $\mathcal{D}$, sa rovnakým spôsobom súčasne opýtame na dosiahnuteľnosť všetkých príkazov schémy.

Množina príkazov schémy je konečná, takže sa v konečnom čase dozvieme, že:

- buď všetky príkazy schémy sú dosiahnuteľné, potom je aj schéma dosiahnuteľná a teda patrí do triedy $\mathcal{D}$,
- alebo existuje príkaz, ktorý nie je dosiahnuteľný, potom aj schéma nie je dosiahnuteľná a teda nepatrí do triedy $\mathcal{D}$,

– alebo sa beh procedúry neskončí a v tomto prípade schéma taktiež nepatrí do triedy $\mathcal{D}$.

Príklad 12 Uvažujme triedu štrukturovaných schém $\mathcal{W}$. Je problém divergencie pre štruktúrované schémy rozhodnuteľný? Svoje tvrdenie zdôvodnite.

Riešenie 12 Problém je rozhodnuteľný.

Štruktúrované schémy obsahujú iba riadiace štruktúry **if** a **while** a neobsahujú riadiacu štruktúru **goto**. Pre každú štruktúrovanú schému sa dá vytvoriť interpretácia taká, že pokiaľ návestie **end** existuje, tak bude dosiahnuté. Konštrukcia interpretácie je triviálna – výsledkom každého predikátu použitého v riadiacej štruktúre **while** musí byť $false$.

Z toho ale vyplýva, že neexistuje divergentná štruktúrovaná schéma taká, že obsahuje návestie **end**. Naopak, ak schéma návestie **end** neobsahuje, tak je určite divergentná. Preto je problém divergencie na $\mathcal{W}$ rozhodnuteľný. Odpoveďou pre každú štruktúrovanú schému je, že schéma je divergentná ak neobsahuje návestie **end**, inak nie je divergentná.

1.3 Porovnávanie tried programových schém

Definícia 13 Uvažujme triedy programových schém $\mathcal{S}_1$ a $\mathcal{S}_2$. Hovoríme, že:

- trieda $\mathcal{S}_1$ je **silnejšia** ako trieda $\mathcal{S}_2$ (označenie $\mathcal{S}_2 \sqsubseteq \mathcal{S}_1$), ak ku každej schéme $S_2 \in \mathcal{S}_2$ existuje schéma $S_1 \in \mathcal{S}_1$ taká, že $S_1 \equiv S_2$ (trieda $\mathcal{S}_2$ je **preložiteľná** do $\mathcal{S}_1$)
- trieda $\mathcal{S}_1$ je **ostro silnejšia** ako trieda $\mathcal{S}_2$ (označenie $\mathcal{S}_2 \sqsubset \mathcal{S}_2$), ak $\mathcal{S}_1$ je silnejšia ako $\mathcal{S}_2$ a existuje schéma $S_1 \in \mathcal{S}_1$, ku ktorej neexistuje ekvivalentná schéma $S_2 \in \mathcal{S}_2$
- triedy $\mathcal{S}_1$ a $\mathcal{S}_2$ sú **rovnocenné**, ak $\mathcal{S}_1 \sqsubset \mathcal{S}_2$ a $\mathcal{S}_2 \sqsubset \mathcal{S}_1$
- trieda $\mathcal{S}_1$ je **efektívne preložiteľná** do $\mathcal{S}_2$ ak existuje algoritmus, ktorý transformuje ľubovoľnú schému $S_1 \in \mathcal{S}_1$ do ekvivalentnej schémy $S_2 \in \mathcal{S}_2$
- triedy $\mathcal{S}_1$ a $\mathcal{S}_2$ sú **efektívne rovnocenné**, ak $\mathcal{S}_1$ je efektívne preložiteľná do $\mathcal{S}_2$ a $\mathcal{S}_2$ je efektívne preložiteľná do $\mathcal{S}_1$

Príklad 13 Rozhodnite, či je schéma S voľná.

$$
\begin{array}{rrl}
S: & \textbf{begin} & [y_1, y_2] := [a, a] \\
& 1: & \textbf{if } p(y_1) \textbf{ then goto } 4 \\
& 2: & [y_1] := [f(y_1)] \\
& 3: & \textbf{goto } 1 \\
& 4: & \textbf{if } p(y_2) \textbf{ then goto end} \\
& 5: & [y_1, y_2] := [g(y_1), f(y_2)] \\
& 6: & \textbf{goto } 4 \\
& \textbf{end} & [z] := [y_1]
\end{array}
$$

Riešenie 13 Z definície vieme, že schéma S je voľná, keď pre každú cestu vedúcu zo začiatočného príkazu existuje interpretácia I a ohodnotenie vstupných premenných v také, že výpočet (S, I, v) sleduje túto cestu.

Schéma S reprezentuje dva cykly. Na začiatku sa premenné y_1 a y_2 inicializujú na rovnakú hodnotu. V prvom cykle sa iteruje podľa y_1, v druhom cykle sa iteruje podľa y_2. V oboch prípadoch testuje ukončenie cyklu predikát p aplikovaný na iteračnú premennú. Takže oba cykly budú mať vždy rovnaký počet opakovaní.

To je ale v rozpore s voľnosťou schémy, pretože nie sú možné výpočty, kde počet opakovaní prvého cyklu nie je rovnaký ako pri druhom cykle. Takže sa nedá spraviť napríklad 3-násobné opakovanie prvého cyklu nasledované 2-násobným opakovaním cyklu druhého. Schéma S teda nie je voľná.

Príklad 14 Máme danú Janovovu schému S_1.

$$
\begin{array}{rrl}
S_1: & \textbf{begin} & [y] := [x] \\
& 1: & [y] := [f(y)] \\
& 2: & \textbf{if } p(y) \textbf{ then goto } 6 \\
& 3: & [y] := [f(y)] \\
& 4: & \textbf{if } p(y) \textbf{ then goto } 6 \\
& 5: & \textbf{goto } 2 \\
& 6: & \textbf{if } q(y) \textbf{ then goto } 8 \\
& 7: & \textbf{goto } 4 \\
& 8: & [y] := [f(y)] \\
& \textbf{end} & [z] := [y]
\end{array}
$$

Nájdite ku schéme S_1 ekvivalentnú voľnú Janovovu schému S_{1_v}. Schému napíšte a stručne zdôvodnite, prečo je schéma S_{1_v} voľná a ekvivalentná so schémou S_1.

Riešenie 14 Riešením je schéma S_{1_v}.

$$
\begin{array}{rll}
S_{1_v}: & \textbf{begin} & [y] := [x] \\
& 1: & [y] := [f(y)] \\
& 2: & \textbf{if } p(y) \textbf{ then goto } 4 \\
& 3: & \textbf{goto } 1 \\
& 4: & \textbf{if } q(y) \textbf{ then goto } 6 \\
& 5: & \textbf{goto } 5 \\
& 6: & [y] := [f(y)] \\
& \textbf{end} & [z] := [y]
\end{array}
$$

Voľnosť: Ak predikát $p(y)$ platí, tak sa už ďalej netestuje. Ak neplatí, tak pred ďalším testom toho istého predikátu sa zmení premenná y. Predikát $q(y)$ sa testuje len raz. Pre každú cestu existuje interpretácia I_{1_v} a valuácia v_{1_v} taká, že výpočet $(S_{1_v},\ I_{1_v},\ v_{1_v})$ sleduje túto cestu, takže schéma je voľná.

Ekvivalencia: Oproti pôvodnej schéme sme zmenili príkaz 7 : **goto** 4 na nový príkaz 5 : **goto** 5. V pôvodnej schéme bol tento príkaz dosiahnuteľný iba ak na riadku 4 platil predikát $p(y)$ a na riadku 6 neplatil predikát $q(y)$, čoho dôsledkom bol opäť skok na riadok 4 a rovnaké testy s rovnakými hodnotami, čiže večný cyklus. Cyklusom 5 : **goto** 5 sme teda dosiahli ekvivalentnú schému.

Taktiež sme vynechali podmienku na riadku 4, pretože môže byť nahradená ekvivalentnou podmienkou na riadku 2 pôvodnej schémy. Nakoniec sme zredukovali príkazy na riadkoch 1 a 3 pôvodnej schémy, pretože sa po malých úpravách novej schémy dajú nahradiť jedným príkazom.

Príklad 15 Daná je štandardná schéma S_2.

$$
\begin{array}{rll}
S_2: & \textbf{begin} & [y] := [x] \\
1: & & \textbf{if } p(y) \textbf{ then goto end} \\
2: & & \textbf{if } q(y) \textbf{ then goto } 6 \\
3: & & [y] := [f_1(y)] \\
4: & & \textbf{if } p(y) \textbf{ then goto } 2 \\
5: & & \textbf{goto end} \\
6: & & \textbf{if } p(y) \textbf{ then goto } 10 \\
7: & & [y] := [f_2(y)] \\
8: & & \textbf{if } q(y) \textbf{ then goto end} \\
9: & & \textbf{goto } 1 \\
10: & & [y] := [f_3(y)] \\
11: & & \textbf{goto } 8 \\
& \textbf{end} & [z] := [y]
\end{array}
$$

Nájdite ku schéme S_2 ekvivalentnú voľnú schému S_{2_v}.

Riešenie 15 Riešením je schéma S_{2_v}.

$$
\begin{array}{rll}
S_{2_v}: & \textbf{begin} & [y] := [x] \\
1: & & \textbf{if } p(y) \textbf{ then goto end} \\
2: & & \textbf{if } q(y) \textbf{ then goto } 6 \\
3: & & [y] := [f_1(y)] \\
4: & & \textbf{if } p(y) \textbf{ then goto } 10 \\
5: & & \textbf{goto end} \\
6: & & [y] := [f_2(y)] \\
7: & & \textbf{if } q(y) \textbf{ then goto end} \\
8: & & \textbf{if } p(y) \textbf{ then goto end} \\
9: & & \textbf{goto } 3 \\
10: & & \textbf{if } q(y) \textbf{ then goto } 12 \\
11: & & \textbf{goto } 3 \\
12: & & [y] := [f_3(y)] \\
13: & & \textbf{goto } 7 \\
& \textbf{end} & [z] := [y]
\end{array}
$$

Schému S_{2_v} sme našli pomocou vytvorenia vývojového diagramu pôvodnej schémy S_2 a jeho následných modifikácií vedúcich k zvoľneniu pri zachovaní ekvivalencie. Tento postup je štandardný. Nedoporučuje sa písať programový kód výslednej schémy priamo[1].

[1] pretože to ide naozaj len veľmi ťažko

Príklad 16 Daná je štandardná schéma S_3.

$$\begin{array}{rrl} S_3: & \textbf{begin} & [y] := [x] \\ & 1: & \textbf{if } p(y) \textbf{ then goto } 9 \\ & 2: & \textbf{if } q(y) \textbf{ then goto } 5 \\ & 3: & \textbf{if } p(y) \textbf{ then goto } 8 \\ & 4: & \textbf{goto } 2 \\ & 5: & [y] := [f_1(y)] \\ & 6: & \textbf{if } q(y) \textbf{ then goto } 1 \\ & 7: & \textbf{goto end} \\ & 8: & [y] := [f_2(y)] \\ & 9: & [y] := [f_3(y)] \\ & \textbf{end} & [z] := [y] \end{array}$$

Nájdite ku schéme S_3 ekvivalentnú voľnú schému S_{3_v}.

Riešenie 16 Riešením úlohy je teda schéma S_{3_v}.

$$\begin{array}{rrl} S_{3_v}: & \textbf{begin} & [y] := [x] \\ & 1: & \textbf{if } q(y) \textbf{ then goto } 4 \\ & 2: & \textbf{if } p(y) \textbf{ then goto } 8 \\ & 3: & \textbf{goto } 3 \\ & 4: & \textbf{if } p(y) \textbf{ then goto } 8 \\ & 5: & [y] := [f_1(y)] \\ & 6: & \textbf{if } q(y) \textbf{ then goto } 1 \\ & 7: & \textbf{goto end} \\ & 8: & [y] := [f_3(y)] \\ & \textbf{end} & [z] := [y] \end{array}$$

Príklad 17 Máme danú štandardnú schému S.

$$\begin{array}{rrl} S: & \textbf{begin} & [y_1, y_2] := [x, a] \\ & 1: & \textbf{if } p(y_1) \textbf{ then goto end} \\ & 2: & [y_1, y_2] := [f(y_1), g(y_1, y_2)] \\ & 3: & \textbf{goto } 1 \\ & \textbf{end} & [z] := [y_2] \end{array}$$

Nájdite ku schéme S ekvivalentnú rekurzívnu schému R.

Riešenie 17 Ekvivalentnú rekurzívnu schému R zostrojíme pomocou štandardného postupu. Návestia štandardnej schémy prepisujeme pomocou funkčných premenných ϕ_i (simulácia toku riadenia) tak, aby v ich rekurzívnych definíciach vektory vstupných argumentov $\overline{y}$ zodpovedali vektorom pracovných premenných (simulácia zmenu stavu výpočtu).

$$\begin{array}{lcl}
\phi_b(y_1, y_2) & = & z = \phi_1(x, a) \\
\phi_1(y_1, y_2) & = & \textbf{if } p(y_1) \textbf{ then } \phi_e(y_1, y_2) \textbf{ else } \phi_2(y_1, y_2) \\
\phi_2(y_1, y_2) & = & \phi_3(f(y_1), g(y_1, y_2)) \\
\phi_3(y_1, y_2) & = & \phi_1(y_1, y_2) \\
\phi_e(y_1, y_2) & = & y_2
\end{array}$$

Jednoduchým dosadením do funkčných premenných ϕ_i dosiahneme zjednodušenie systému rekurzívnych definícií a výslednú schému R.

$$\begin{array}{lll}
R: & \textbf{begin} & [y_1, y_2] := [x, a] \\
 & & \phi_1(y_1, y_2) \Longleftarrow \textbf{if } p(y_1) \textbf{ then } y_2 \textbf{ else } \phi_1(f(y_1), g(y_1, y_2)) \\
 & \textbf{end} & [z] := [\phi_1(x, a)]
\end{array}$$

Príklad 18 Daná je štandardná schéma S.

$$\begin{array}{lrl}
S: & \textbf{begin} & [y] := [x] \\
 & 1: & \textbf{if } p(y) \textbf{ then goto end} \\
 & 2: & [y] := [f(y)] \\
 & 3: & \textbf{if } q(y) \textbf{ then } [y] := [g(y)] \\
 & 4: & \textbf{if } p(y) \textbf{ then goto } 2 \\
 & 5: & \textbf{goto } 1 \\
 & \textbf{end} & [z] := [y]
\end{array}$$

Nájdite rekurzívnu schému R, ktorá je ekvivalentná so schémou S. Upravte nájdenú schému tak, aby mala minimálny počet funkčných premenných.

Riešenie 18 Keďže ide o úlohu rovnakého typu ako v predchádzajúcom príklade, aj náš postup bude obdobný. Najskôr vytvoríme základnú sústavu rekurzívnych definícií.

$$
\begin{array}{lcl}
\phi_b(y) & = & z = \phi_1(x) \\
\phi_1(y) & = & \textbf{if } p(y) \textbf{ then } \phi_e(y) \textbf{ else } \phi_3(f(y)) \\
\phi_2(y) & = & \phi_3(f(y)) \\
\phi_3(y) & = & \textbf{if } q(y) \textbf{ then } \phi_4(g(y)) \textbf{ else } \phi_4(y) \\
\phi_4(y) & = & \textbf{if } p(y) \textbf{ then } \phi_2(y) \textbf{ else } \phi_5(y) \\
\phi_5(y) & = & \phi_1(y) \\
\phi_e(y) & = & y
\end{array}
$$

Minimálny počet funčných premenných dosiahneme nasledovnými krokmi:

- Zrušením ϕ_2 a dosadením ϕ_3 na príslušné miesta vo ϕ_1 a ϕ_4.
- Zrušením ϕ_e a dosadením y na príslušné miesto vo ϕ_1.
- Zrušením ϕ_5 a dosadením ϕ_1 na príslušné miesto vo ϕ_4.
- Zrušením ϕ_4 a dosadením príkazu **if** na príslušné miesta vo ϕ_3.

Po týchto úpravách dostávame výslednú rekurzívnu schému R, ktorá je ekvivalentá so schémou S.

$$
\begin{array}{lll}
R: & \textbf{begin} & [y] := [x] \\
 & & \phi_1(y) \Longleftarrow \textbf{if } p(y) \textbf{ then } y \\
 & & \qquad\qquad\qquad \textbf{else } \phi_3(f(y)) \\
 & & \phi_3(y) \Longleftarrow \textbf{if } q(y) \textbf{ then if } p(g(y)) \textbf{ then } \phi_3(f(g(y))) \\
 & & \qquad\qquad\qquad\qquad\qquad\qquad\quad \textbf{else } \phi_1(g(y)) \\
 & & \qquad\qquad\qquad \textbf{else if } p(y) \textbf{ then } \phi_3(f(y)) \\
 & & \qquad\qquad\qquad\qquad\quad \textbf{else } \phi_1(y) \\
 & \textbf{end} & [z] := [\phi_1(x)]
\end{array}
$$

Príklad 19 Máme danú štandardnú schému S.

$$
\begin{array}{lrl}
S: & \textbf{begin} & [y] := [x] \\
 & 1: & [y] := [f(y)] \\
 & 2: & \textbf{if } p(y) \textbf{ then goto } 5 \\
 & 3: & [y] := [g(y)] \\
 & 4: & \textbf{goto } 1 \\
 & 5: & \textbf{if } p(y) \textbf{ then } [y] := [h(y)] \\
 & 6: & \textbf{if } p(y) \textbf{ then goto end} \\
 & 7: & \textbf{goto } 5 \\
 & \textbf{end} & [z] := [y]
\end{array}
$$

Nájdite ku schéme S ekvivalentnú rekurzívnu schému R.

Riešenie 19 Do tretice je uvedený príklad rovnakého typu, tj. úloha na prevod štandardnej schémy do rekurzívnej. Tentoraz však iba s výslednou podobou rekurzívnej schémy. Čitateľ si tak môže aspoň porovnať výsledok.

$$
\begin{array}{lll}
R: & \textbf{begin} & [y] := [x] \\
 & & \phi_1(y) \Longleftarrow \textbf{if } p(f(y)) \textbf{ then } \phi_5(f(y)) \\
 & & \qquad\qquad\qquad\qquad \textbf{else } \phi_1(g(f(y))) \\
 & & \phi_5(y) \Longleftarrow \textbf{if } p(y) \textbf{ then if } q(h(y)) \textbf{ then } h(y) \\
 & & \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \textbf{else } \phi_5(h(y)) \\
 & & \qquad\qquad\qquad\qquad\quad \textbf{else if } q(y) \textbf{ then } y \\
 & & \qquad\qquad\qquad\qquad\qquad\qquad\quad \textbf{else } \phi_5(y) \\
 & \textbf{end} & [z] := [\phi_1(x)]
\end{array}
$$

Príklad 20 Máme danú rekurzívnu schému R.

$$
\begin{array}{lll}
R: & \textbf{begin} & [\ldots] := [\ldots] \\
 & & \phi(y) \Longleftarrow \textbf{if } p(y) \textbf{ then } f(y) \textbf{ else } h(\phi(g(y))) \\
 & \textbf{end} & [z] := [\phi(a)]
\end{array}
$$

Zistite, či existuje k tejto schéme štandardná schéma S. V prípade, že existuje, nájdite ju.

Riešenie 20 Štandardným postupom hľadania ekvivalentnej rekurzívnej schémy k štandardnej schéme je:

1. Zistiť akého tvaru je výstupná premenná rekurzívnej schémy a rozhodnúť, či môže existovať štandardná schéma dávajúca rovnaké výsledky.
2. V prípade kladnej odpovede v predchádzajúcom bode už ostáva iba túto schému nájsť. V mnohých prípadoch to ide, nie však vo všeobecnosti.

Výstupná premenná z schémy R je tvaru $h^n f g^n(a)$, kde hodnota n vyjadruje hĺbku rekurzívneho vnorenia. V triede štandardných schém $\mathcal{S}$ existuje schéma S ekvivalentná s R generujúca výstupné premenné uvedeného tvaru.

$$
\begin{array}{rll}
S: & \textbf{begin} & [y_1, y_2] := [a, a] \\
& 1: & \textbf{if } p_1(y_1) \textbf{ then goto } 4 \\
& 2: & [y_1] := [g(y_1)] \\
& 3: & \textbf{goto } 1 \\
& 4: & [y_1] := [f(y_1)] \\
& 5: & \textbf{if } p(y_2) \textbf{ then goto end} \\
& 6: & [y_1, y_2] := [h(y_1), g(y_2)] \\
& 7: & \textbf{goto } 5 \\
& \textbf{end} & [z] := [y_1]
\end{array}
$$

Obsah pracovných premenných $[y_1,\ y_2]$ v príkaze **begin** bol $[a,\ a]$, pred riadkom 4 bol $[g^n(a),\ a]$, po riadku 4 bol $[fg^n(a),\ a]$ a nakoniec v príkaze **end** bol obsah pracovných premenných $[h^n fg^n(a),\ g^n(a)]$. Výstupnej premennej z sa priraďuje hodnota y_1, ktorej obsah je v žiadanom tvare.

Príklad 21 Uvažujme triedu štruktúrovaných schém $\mathcal{W}^b$. Trieda $\mathcal{W}^b$ je obohatená o booleovské premenné a dobre otypované priradenie do booleovských premenných. Ktorú z uvedených konštrukcií je alebo nie je možné preložiť do ekvivalentnej schémy z triedy $\mathcal{W}^b$? Zdôvodnite prečo.

$$
\begin{array}{ll}
W_1: & \textbf{while } b_1 \wedge b_2 \textbf{ do } S \textbf{ od} \\
W_2: & \textbf{while } b_1 \vee b_2 \textbf{ do } S \textbf{ od} \\
W_3: & \textbf{while } \neg b \textbf{ do } S \textbf{ od}
\end{array}
$$

Riešenie 21 Začneme konštrukciou W_2, ktorá ide preložiť do triedy $\mathcal{W}$. Z toho vyplýva, že určite pôjde preložiť aj do triedy $\mathcal{W}^b$. Použitie booleovských premenných a priradení je však v tomto prípade nepotrebné.

$$
\begin{array}{ll}
W_2': & \textbf{while } b_1 \textbf{ do } S \textbf{ od} \\
& \textbf{while } b_2 \textbf{ do} \\
& \quad S; \\
& \quad \textbf{while } b_1 \textbf{ do } S; \\
& \textbf{od}
\end{array}
$$

Pre nasledujúce programové konštrukcie však už bude použitie booleovskej premennej nevyhnutné. Máme teda možnosť použitia premennej *bool*, do ktorej môžeme priraďovať hodnoty iných booleovských premenných, ako je napr. b_1 alebo b_2. Tiež je možné priamo priraďovať booleovské konštanty $true$ a $false$. Je nutné si ale uvedomiť, že trieda $\mathcal{W}^b$ nie je čiastočne interpretovaná vzhľadom na použitie logických operátorov $\wedge$ (AND), $\vee$ (OR) alebo $\neg$ (NOT).

Nájdime teda konštrukciu $W_1' \in \mathcal{W}^b$ ekvivalentnú s W_1.

$$
\begin{array}{ll}
W_1': & [bool] := false; \\
& \textbf{if } b_1 \textbf{ then} \\
& \quad \textbf{if } b_2 \textbf{ then } [bool] := true; \\
& \textbf{while } bool \textbf{ do} \\
& \quad S; \\
& \quad [bool] := false; \\
& \quad \textbf{if } b_1 \textbf{ then} \\
& \quad\quad \textbf{if } b_2 \textbf{ then } [bool] := true; \\
& \textbf{od}
\end{array}
$$

Viac elegancie v riešení je možné získať nahradením konštrukcií

$$
\begin{array}{l}
[bool] := false; \\
\textbf{if } b_1 \textbf{ then} \\
\quad \textbf{if } b_2 \textbf{ then } [bool] := true;
\end{array}
$$

za jednoduchšie

$$
\begin{array}{l}
\textbf{if } b_1 \textbf{ then } [bool] := [b_2] \\
\quad\quad \textbf{else } [bool] := [b_1]
\end{array}
$$

Uvedeným spôsobom sa dá dokonca vyhnúť použitiu konštánt $true$ a $false$. To už však nie je možné pri zostávajúcej konštrukcii $W_3' \in \mathcal{W}^b$, ktorá je ekvivalentná s W_3.

$$
\begin{array}{ll}
W_3': & [bool] := true; \\
& \textbf{if } b \textbf{ then } [bool] := false; \\
& \textbf{while } bool \textbf{ do} \\
& \quad S; \\
& \quad [bool] := true; \\
& \quad \textbf{if } b \textbf{ then } [bool] := false; \\
& \textbf{od}
\end{array}
$$

Príklad 22 Uvažujme triedu voľných schém $\mathcal{V}$, triedu rekurzívnych schém $\mathcal{R}$ a triedu dosiahnuteľných schém $\mathcal{D}$. Sformulujte a zdôvodnite vzťahy medzi uvedenými triedami schém a triedou štandardných schém $\mathcal{S}$ na základe relácií podtrieda $\subseteq$, preložiteľná trieda $\sqsubseteq$ a efektívne preložiteľná trieda $\trianglelefteq$.

Riešenie 22

$\mathcal{S}$, $\mathcal{V}$ – porovnanie tried štandardných a voľných schém

$\mathcal{S} \not\subseteq \mathcal{V}$ Existuje štandardná schéma, ktorá nie je voľná.

$\mathcal{S} \not\sqsubseteq \mathcal{V}$ Vo všeobecnosti neexistuje postup, pomocou ktorého sa dá spraviť ekvivalentná voľná schéma ku každej štandardnej schéme, aj keď v mnohých prípadoch to ide. Pre kontrapríklad pozri tvrdenie $\mathcal{D} \not\sqsubseteq \mathcal{V}$.

$\mathcal{S} \not\trianglelefteq \mathcal{V}$ Priamo vyplýva z tvrdenia $\mathcal{S} \not\sqsubseteq \mathcal{V}$.

$\mathcal{V} \subseteq \mathcal{S},\ \mathcal{V} \sqsubseteq \mathcal{S},\ \mathcal{V} \trianglelefteq \mathcal{S}$
Tvrdenia sú zrejmé, vyplývajú priamo z definícií.

$\mathcal{S}, \mathcal{R}$ – porovnanie tried štandardných a rekurzívnych schém

$\mathcal{S} \not\subseteq \mathcal{R}$ Ide o syntakticky odlišné schémy, takže principiálne nemôžu byť navzájom podtriedami.

$\mathcal{S} \sqsubseteq \mathcal{R}$ Štandardná schéma sa dá previesť na ekvivalentnú rekurzívnu schému.

$\mathcal{S} \trianglelefteq \mathcal{R}$ Existuje štandardný postup, pomocou ktorého sa dá previesť štandardná schéma na ekvivalentú rekurzívnu. Niekoľko ukážok sa nachádza aj v tejto zbierke príkladov.

$\mathcal{R} \not\subseteq \mathcal{S}$ Ide o syntakticky odlišné schémy, takže principiálne nemôžu byť navzájom podtriedami.

$\mathcal{R} \not\sqsubseteq \mathcal{S}$ Existuje rekurzívna schéma ku ktorej neexistuje ekvivalentná štandardná schéma. V niektorých prípadoch sa však rekurzívna schéma dá previesť na ekvivalentú štandardnú (viz. napríklad úlohu v tejto zbierke).

$\mathcal{R} \not\trianglelefteq \mathcal{S}$ Priamo vyplýva z tvrdenia $\mathcal{R} \not\sqsubseteq \mathcal{S}$.

$\mathcal{S}, \mathcal{D}$ – porovnanie tried štandardných a dosiahnuteľných schém

$\mathcal{S} \not\subseteq \mathcal{D}$ Existuje štandardná schém, ktorá nie je dosiahnuteľná. Kontrapríkladom je schéma obsahujúca jeden alebo viac komponetov nesúvislosti (tzv. ostrovčeky).

$\mathcal{S} \sqsubseteq \mathcal{D}$ Odstránením komponentov nesúvislosti zo štandardnej schémy sa stane schéma dosiahnuteľnou.

$\mathcal{S} \not\trianglelefteq \mathcal{D}$ Odstraňovanie komponetov nesúvislosti však nemôže ísť spravit' efektívne. Ak by to išlo, vedeli by sme rozhodovať divergenciu štandardných schém. Každú schému z triedy $\mathcal{S}$ by sme previedli na ekvivalentnú schému z triedy $\mathcal{D}$ a spýtali sa na dosiahnuteľnosť návestia **end**.

$\mathcal{D} \subseteq \mathcal{S},\ \mathcal{D} \sqsubseteq \mathcal{S},\ \mathcal{D} \trianglelefteq \mathcal{S}$
Tvrdenia sú zrejmé, vyplývajú priamo z definícií.

Príklad 23 Uvažujme triedu dosiahnuteľných schém $\mathcal{D}$, triedu priechodných schém $\mathcal{P}$ a triedu Janovových schém $\mathcal{J}$. Sformulujte a zdôvodnite vzťahy medzi uvedenými triedami schém a triedou voľných schém $\mathcal{V}$ na základe relácií podtrieda $\subseteq$, preložiteľná trieda $\sqsubseteq$ a efektívne preložiteľná trieda $\trianglelefteq$.

Riešenie 23

$\mathcal{V}$, $\mathcal{D}$ – porovnanie tried voľných a dosiahnuteľných schém

$\mathcal{V} \not\sqsubseteq \mathcal{D}$ Komponenty nesúvislosti neodporujú voľnosti, nakoľko do nich nevedie cesta zo začiatku schémy. Odporujú však dosiahnuteľnosti schémy. Preto existuje schéma, ktorá je voľná, ale nie je dosiahnuteľná.

$\mathcal{V} \sqsubseteq \mathcal{D}$ Vyplýva z tvrdení $\mathcal{V} \subseteq \mathcal{S} \wedge \mathcal{S} \sqsubseteq \mathcal{D}$.

$\mathcal{V} \trianglelefteq \mathcal{D}$ Keďže voľné schémy sú orientované grafy a na orientovaných grafoch existuje algoritmus odstraňujúci komponenty nesúvislosti, potom sa dá každá voľná schéma efektívne preložit' do ekvivalentnej dosiahnuteľnej schémy. Algoritmus je lineárny vzhľadom na počet hrán a kvadratický vzhľadom na počet príkazov schémy.

$\mathcal{D} \not\sqsubseteq \mathcal{V}$ Existuje dosiahnuteľná schéma, ktorá nie je voľná.

$$\begin{aligned} i: &\ \textbf{if } p(y) \textbf{ then goto } i+2 \\ i+1: &\ \textbf{if } p(y) \textbf{ then goto } i+3 \\ i+2: &\ [y] := [y] \\ i+3: &\ \ldots \end{aligned}$$

Všetky príkazy v načrtnutej časti schémy sú dosiahnuteľné, ale cesta z $i+1$ do $i+3$ nie je voľná.

$\mathcal{D} \not\sqsubseteq \mathcal{V}$ Existuje dosiahnuteľná schéma, ktorá sa nedá previesť na voľnú. Príkladom môže byť napríklad nasledujúca schéma s dvojitým cyklusom dávajúca na výstupe $f^n g^n(a)$, kde n je počet opakovaní prvého a druhého cyklu.

begin $[y_1, y_2] := [a, a]$
1 : **if** $p(y_1)$ **then goto** 4
2 : $[y_1] := [f(y_1)]$
3 : **goto** 1
4 : **if** $p(y_2)$ **then goto end**
5 : $[y_1, y_2] := [g(y_1), f(y_2)]$
6 : **goto** 4
end $[z] := [y_1]$

$\mathcal{D} \not\trianglelefteq \mathcal{V}$ Priamo vyplýva z tvrdenia $\mathcal{D} \not\sqsubseteq \mathcal{V}$.

$\mathcal{V}$, $\mathcal{P}$ – porovnanie tried voľných a priechodných schém

$\mathcal{V} \not\subseteq \mathcal{P}$ Existuje voľná schéma, ktorá nie je priechodná. Inkriminovaná schéma obsahuje také komponenty nesúvislosti, ktoré neodporujú voľnosti, ale odporujú priechodnosti schémy.

$\mathcal{V} \sqsubseteq \mathcal{P}$ Odstránením komponentov nesúvislosti voľnej schémy dostávame ekvivalentnú priechodnú schému.

$\mathcal{V} \trianglelefteq \mathcal{P}$ Analogicky ako dôkaz tvrdenia $\mathcal{V} \trianglelefteq \mathcal{D}$. Je nutné najdenie komponenty súvislosti orientovaného grafu s vrcholom **begin** reprezentujúceho voľnú schému a odstránenie ostatných komponentov nesúvislosti.

$\mathcal{P} \not\subseteq \mathcal{V}$ Existuje priechodná schéma, ktorá nie je voľná. Pre kontrapríklad pozri tvrdenie $\mathcal{D} \not\sqsubseteq \mathcal{V}$.

$\mathcal{P} \not\sqsubseteq \mathcal{V}$ Existuje priechodná schéma, ktorá sa nedá preložiť do ekvivalentnej voľnej schémy. Pre kontrapríklad pozri tvrdenie $\mathcal{D} \not\sqsubseteq \mathcal{V}$.

$\mathcal{P} \not\trianglelefteq \mathcal{V}$ Priamo vyplýva z tvrdenia $\mathcal{P} \not\sqsubseteq \mathcal{V}$.

$\mathcal{V}$, $\mathcal{J}$ – porovnanie tried voľných a Janovových schém

$\mathcal{V} \not\subseteq \mathcal{J}$ Existuje voľná schéma, ktorá nie je Janovova. Je to jednoducho taká, ktorá obsahuje viac ako jednu pracovnú premennú.

$\mathcal{V} \not\sqsubseteq \mathcal{J}$ Existuje voľná schéma, ktorá sa nedá preložiť do ekvivalentnej Janovovej schémy. Ako kontrapríklad poslúži voľná schéma, ktorá obsahuje dve pracovné premenné, pričom obe sú v schéme využívané tak, že sa nedajú nahradiť jednou pracovnou premennou.

$\mathcal{V} \not\trianglelefteq \mathcal{J}$ Priamo vyplýva z tvrdenia $\mathcal{V} \not\sqsubseteq \mathcal{J}$.

$\mathcal{J} \not\subseteq \mathcal{V}$ Existuje Janovova schéma, ktorá nie je voľná.

$\mathcal{J} \sqsubseteq \mathcal{V}$ Z každej Janovovej schémy sa dá spraviť ekvivalentná voľná Janovova schéma. Dôkaz tvrdenia sa nachádza v skriptách.

$\mathcal{J} \trianglelefteq \mathcal{V}$ Každá Janovova schéma sa dá efektívne preložiť do ekvivalentnej voľnej Janovovej schémy. Popis postupu sa opäť nachádza v skriptách.

Kapitola 2

Správnosť programov

2.1 Metódy dokazovania správnosti

Príklad 24 Uvažujme nasledujúci štandardný program P, ktorý počíta $\lceil\sqrt{x}\rceil$ (hornú celú časť odmocniny x).

$$\begin{array}{rll} P: & \textbf{begin} & [y_1, y_2] := [1, 1] \\ & 1: & \textbf{if } y_2 \geq x \textbf{ then goto end} \\ & 2: & [y_1, y_2] := [y_1 + 1, (y_1 + 1)^2] \\ & 3: & \textbf{goto } 1 \\ & \textbf{end} & [z] := [y_1] \end{array}$$

Definujte vstupnú podmienku, výstupnú podmienku a invarianty. Floydovou metódou dokážte čiastočnú správnosť programu vzhľadom na vstupnú a výstupnú podmienku.

Riešenie 24 Vstupná podmienka presne vymedzuje vstupné hodnoty pre ktoré dáva program žiadaný výsledok. V našom prípade ide o všetky kladné hodnoty. Program by sa dal modifikovať aj tak, aby dával zmysluplné výsledky pre všetky nezáporné hodnoty, ale prinieslo by nám to isté množstvo ďalších komplikácií. Takže vstupná podmienka p vyzerá nasledovne.

$$p(x): \ x > 0$$

Výstupná podmienka q popisuje $z = \lceil\sqrt{x}\rceil$.

$$\begin{array}{lccccc} q(x,z): & & & \lceil\sqrt{x}\rceil & = & z \\ & (z-1) & < & \sqrt{x} & \leq & z \\ & (z-1)^2 & < & x & \leq & z^2 \end{array}$$

Invariantu v príkaze **begin** zodpovedá vstupná podmienka a invariantu v príkaze **end** zodpovedá výstupná podmienka.

$$\begin{array}{rcl} I_B & = & p \\ I_E & = & q \end{array}$$

Program obsahuje jeden cyklus. Ideálne miesto pre jeho deliaci bod je tam, kde sa z cyklu vychádza. V programe P je to rovnaké miesto ako to, kde sa do cyklu vchádza. Ako deliaci bod teda zvolíme riadok 1. Invariant I_1 v tomto bode vyzerá nasledovne.

$$I_1: \ (y_1 - 1)^2 < x \ \wedge \ y_2 = y_1^2$$

Prvá časť invariantu I_1 reprezentuje riadiacu podmienku cyklu. V druhej časti sa definuje závislosť medzi premennými y_1 a y_2.

Program P obsahuje tri deliace body medzi ktorými sú tri konečné cesty.

- cesta B1: **begin** $\rightarrow$ riadok 1
- cesta 11: riadok 1 $\rightarrow$ riadok 1
- cesta 1E: riadok 1 $\rightarrow$ **end**

Z definície vieme, že pre každú cestu musíme dokázať verifikačnú podmienku odvodenú z nasledujúceho všeobecného tvaru.

$$\forall \overline{x}, \overline{y} \ \ I_A(\overline{x}, \overline{y}) \ \wedge \ R_{AB}(\overline{x}, \overline{y}) \implies I_B(\overline{x}, r_{AB}(\overline{x}, \overline{y}))$$

cesta B1: Použitím spätnej substitúcie odvodíme podmienku prechodu a modifikáciu pracovných premenných na ceste B1 a dosadíme do príslušnej verifikačnej podmienky.

$$\begin{array}{rcl} R_{B1}(y_1,\ y_2) & = & true \\ r_{B1}(y_1,\ y_2) & = & (1,\ 1) \end{array}$$

$$\begin{array}{rcl} I_B \ \wedge \ true & \Rightarrow & I_1(1,1) \\ x > 0 \ \wedge \ true & \Rightarrow & (1-1)^2 < x \ \wedge \ 1^2 = 1 \\ x > 0 \ \wedge \ true & \Rightarrow & 0 < x \ \wedge \ 1 = 1 \\ x > 0 \ \wedge \ true & \Rightarrow & x > 0 \end{array}$$

cesta 11: Podobne ako v predchádzajúcom prípade odvodíme R_{11} a r_{11} a dosadíme do verifikačnej podmienky pre cestu 11.

$$\begin{aligned} R_{11}(y_1,\ y_2) &= true \ \wedge\ \neg(y_2 \geq x) \ =\ y_2 < x \\ r_{11}(y_1,\ y_2) &= (y_1 + 1, (y_1 + 1)^2) \end{aligned}$$

$$\begin{aligned} I_1(y_1, y_2) \ \wedge\ y_2 < x &\ \Rightarrow\ I_1(y_1 + 1, (y_1 + 1)^2) \\ (y_1 - 1)^2 < x \ \wedge\ y_1^2 = y_2 \ \wedge\ y_2 < x &\ \Rightarrow\ (y_1 + 1 - 1)^2 < x \ \wedge\ (y_1 + 1)^2 = (y_1 + 1)^2 \\ y_1^2 = y_2 \ \wedge\ y_2 < x &\ \Rightarrow\ y_1^2 < x \ \wedge\ true \\ y_1^2 < x &\ \Rightarrow\ y_1^2 < x \end{aligned}$$

cesta 1E: A do tretice aj pre poslednú cestu odvodíme spätnou substitúciou R_{1E} a r_{1E} a dosadíme do prislúchajúcej verifikačnej podmienky.

$$\begin{aligned} R_{1E}(y_1,\ y_2) &= true \ \wedge\ y_2 \geq x \\ r_{1E}(z) &= y_1 \end{aligned}$$

$$\begin{aligned} I_1(y_1, y_2) \ \wedge\ y_2 \geq x &\ \Rightarrow\ I_E \\ (y_1 - 1)^2 < x \ \wedge\ y_2 = y_1^2 \ \wedge\ y_2 \geq x &\ \Rightarrow\ (y_1 - 1)^2 < x \leq y_1^2 \\ (y_1 - 1)^2 < x \ \wedge\ y_1^2 \geq x &\ \Rightarrow\ (y_1 - 1)^2 < x \ \wedge\ y_1^2 \geq x \end{aligned}$$

Pre všetky cesty v programe P sme overili platnosť príslušných odvodených verifikačných podmienok a tým sme dokázali aj čiastočnú správnosť programu P vzhľadom na vstupnú podmienku p a výstupnú podmienku q.

Príklad 25 Daný je štandardný program P.

$$\begin{aligned} P:\ \ \textbf{begin}\ & [y_1, y_2] := [0, x_1] \\ 1:\ & \textbf{if}\ y_2 < x_2\ \textbf{then goto end} \\ 2:\ & [y_1, y_2] := [y_1 + 1, y_2 - x_2] \\ 3:\ & \textbf{goto}\ 1 \\ \textbf{end}\ & [z_1, z_2] := [y_1, y_2] \end{aligned}$$

Floydovou metódou formálne dokážte čiastočnú správnosť programu P vzhľadom na nasledujúce podmienky:

- vstupná podmienka – $p(x_1, x_2) :\ x_1 \geq 0 \ \wedge\ x_2 > 0$
- výstupná podmienka – $q(x_1, x_2, z_1, z_2) :\ z_1 x_2 + z_2 = x_1 \ \wedge\ 0 \leq z_2 < x_2$

Určte deliace body, nájdite k nim prislúchajúce invarianty, zostrojte verifikačné podmienky a ukážte, že platia.

Riešenie 25 Po krátkej analýze zistíme, že program delí hodnotu vstupnej premennej x_1 hodnotou x_2. Deliteľ je uložený do výstupnej premennej z_1 a zvyšok po delení do premennej z_2. Výsledok je tak tvaru $x_1 = z_1 x_2 + z_2$, kde $z_2 < x_2$ (zvyšok je menší ako hodnota, ktorou delíme).

Máme dva implicitné deliace body v návestiach **begin** a **end**. Invariantom v týchto bodoch zodpovedajú vstupná podmienka p a výstupná podmienka q. Takže platí $I_B = p$ a $I_E = q$.

Keďže program opäť obsahuje jeden cyklus, ako jeho deliaci bod zvolíme riadok 1. Je to miesto kde sa do cyklu vchádza i vychádza. Konštrukcia invariantu k tomuto deliacemu bodu programu je nerozhodnuteľným problémom. Preto sa snažíme vyčítať z vlastností programu i cyklu čo najviac užitočných informácií a vložiť ich do podmienok invariantu I_1.

$$I_1 : \; x_1 = y_1 x_2 + y_2 \;\wedge\; x_2 > 0 \;\wedge\; y_1 \geq 0 \;\wedge\; y_2 \geq 0$$

Pre každú z troch ciest odvodíme a dokážeme verifikačnú podmienku.

cesta B1:

$$\begin{aligned} R_{B1}(y_1,\, y_2) &= true \\ r_{B1}(y_1,\, y_2) &= (0,\, x_1) \end{aligned}$$

$$\begin{aligned} I_B \;\wedge\; true &\;\Rightarrow\; I_1(0, x_1) \\ x_1 \geq 0 \;\wedge\; x_2 > 0 &\;\Rightarrow\; x_1 = 0 x_2 + x_1 \;\wedge\; x_2 > 0 \;\wedge\; 0 \geq 0 \;\wedge\; x_1 \geq 0 \\ x_1 \geq 0 \;\wedge\; x_2 > 0 &\;\Rightarrow\; x_1 = x_1 \;\wedge\; x_2 > 0 \;\wedge\; x_1 \geq 0 \\ x_1 \geq 0 \;\wedge\; x_2 > 0 &\;\Rightarrow\; x_1 \geq 0 \;\wedge\; x_2 > 0 \end{aligned}$$

cesta 11:

$$\begin{aligned} R_{11}(y_1,\, y_2) &= true \;\wedge\; \neg(y_2 < x_2) \;=\; y_2 \geq x_2 \\ r_{11}(y_1,\, y_2) &= (y_1 + 1, y_2 - x_2) \end{aligned}$$

$$I_1(y_1, y_2) \;\wedge\; y_2 \geq x_2 \;\Rightarrow\; I_1(y_1 + 1, y_2 - x_2)$$

$$\begin{aligned} &x_1 = y_1 x_2 + y_2 \;\wedge\; x_2 > 0 \;\wedge\; y_1 \geq 0 \;\wedge\; y_2 \geq 0 \;\wedge\; y_2 \geq x_2 \;\Rightarrow \\ &\Rightarrow\; x_1 = (y_1 + 1) x_2 + (y_2 - x_2) \;\wedge\; x_2 > 0 \;\wedge\; y_1 + 1 \geq 0 \;\wedge\; y_2 - x_2 \geq 0 \\ &\qquad\qquad \text{(zrejme } y_2 \geq 0 \wedge x_2 > 0 \wedge y_2 \geq x_2 \Rightarrow y_2 - x_2 \geq 0\text{)} \\ &x_1 = y_1 x_2 + y_2 \;\wedge\; y_1 \geq 0 \;\wedge\; y_2 - x_2 \geq 0 \;\Rightarrow \\ &\qquad\qquad \Rightarrow\; x_1 = y_1 x_2 + y_2 \;\wedge\; y_1 + 1 \geq 0 \;\wedge\; y_2 - x_2 \geq 0 \\ &\qquad\qquad\qquad \text{(zrejme platí } y_1 \geq 0 \Rightarrow y_1 + 1 \geq 0\text{)} \end{aligned}$$

cesta 1E:

$$\begin{aligned} R_{1E}(y_1,\, y_2) &= true \;\wedge\; y_2 < x_2 \\ r_{1E}(z_1, z_2) &= (y_1, y_2) \end{aligned}$$

$$I_1(y_1, y_2) \ \wedge \ y_2 < x_2 \ \Rightarrow \ I_E$$

$$x_1 = y_1 x_2 + y_2 \ \wedge \ x_2 > 0 \ \wedge \ y_1 \geq 0 \ \wedge \ y_2 \geq 0 \ \wedge \ y_2 < x_2 \ \Rightarrow$$
$$\Rightarrow \ x_1 = y_1 x_2 + y_2 \ \wedge \ 0 \leq y_2 < x_2$$
$$x_1 = y_1 x_2 + y_2 \ \wedge \ y_2 \geq 0 \ \wedge \ y_2 < x_2 \ \Rightarrow$$
$$\Rightarrow \ x_1 = y_1 x_2 + y_2 \ \wedge \ y_2 \geq 0 \ \wedge y_2 < x_2$$

Po overení verifikačných podmienok pre všetky cesty je čiastočná správnosť programu P dokázaná.

Príklad 26 Daný je štruktúrovaný program P.

$$\begin{array}{lll} P: & \textbf{begin} & [y_1, y_2] := [1, 1] \\ & & \textbf{while } y_2 < x \textbf{ do} \\ & & \quad [y_1, y_2] := [y_1 + 1, (y_1 + 1)^2] \\ & & \textbf{od} \\ & \textbf{end} & [z] := [y_1] \end{array}$$

Hoareovou metódou formálne dokážte čiastočnú správnosť programu P vzhľadom na nasledujúce podmienky:

- vstupná podmienka – $p(x): \ x > 0$
- výstupná podmienka – $q(x, y): \ (z-1)^2 < x \leq z^2$

Riešenie 26 Hoareova metóda dokazovania čiastočnej správnosti štruktúrovaných programov sa opiera o logický systém založený na jazyku induktívnych formúl $\{p\}\ P\ \{q\}$. Pri dokazovaní sa používajú všetky platné formuly špecifikačného jazyka, axióma priradenia a inferenčné resp. odvodzovacie pravidlá Hoareovského kalkulu.

Pre dôkaz čiastočnej správnosti programu P musíme dokázať platnosť nasledujúcej induktívnej formuly.

$$\{p\}\ P\ \{q\}$$

Krátkou analýzou zistíme, že program P sa skladá z troch častí. Z dostupných inferenčných pravidiel teda aplikujeme *pravidlo kompozície* a vyriešime induktívne formuly zodpovedajúce jednotlivým častiam programu P.

$$\{p\}\ P_1\ \{r\} \quad \{r\}\ P_2\ \{s\} \quad \{s\}\ P_3\ \{q\}$$

Ešte pred samotným riešením jednotlivých induktívnych formúl sa pokúsime prispôsobiť si podmienky r a s. Časť P_2 je **while** cyklus s podmienkou b, preto uhádneme, ako by mohla vyzerať podmienka s, ktorá platí po jeho ukončení.

$$s \equiv r \wedge \neg b$$

Je nutné podotknúť, že v našom prípade tento postup povedie k úspechu. Rozhodne však neplatí vo všeobecnosti na všetky štruktúrované programy.

- Induktívna formula $\{p\}\ P_1\ \{r\}$: Časť P_1 programu P obsahuje jediné priradenie v návestí **begin**. Použitím *axiómy priradenia* nahradíme v podmienke r obe premenné y_1 a y_2 hodnotou 1. Určite platí

  $$\{r[y_1/1, y_2/1]\}\ P_1\ \{r\}$$

  Ak sa podarí dokázať nasledujúcu implikáciu, bude možné použiť *pravidlo následku* a tým bude induktívna formula $\{p\}\ P_1\ \{r\}$ dokázaná.

  $$p \Rightarrow r[y_1/1, y_2/1]$$

- Induktívna formula $\{r\}\ P_2\ \{r \wedge \neg b\}$: Časť P_2 programu P je reprezentovaná cyklusom **while**. Preto použijeme *pravidlo iterácie*.

  $$\{r \wedge b\}\ P_{21}\ \{r\}$$

  Pre priradenie nachádzajúce sa v cykle **while** použijeme *axiómu priradenia*. Určite teda platí

  $$\{r[y_1/y_1 + 1, y_2/(y_1 + 1)^2]\}\ P_{21}\ \{r\}$$

  Dokázaním nasledujúcej implikácie dokážeme platnosť celej induktívnej formuly $\{r\}\ P_2\ \{r \wedge \neg b\}$.

  $$r \wedge b \Rightarrow r[y_1/y_1 + 1, y_2/(y_1 + 1)^2]$$

- Induktívna formula $\{r \wedge \neg b\}\ P_3\ \{q\}$: Podobne ako časť P_1 aj časť P_3 programu P obsahuje len jedno priradenie, tentokrát v návestí **end**. Aplikujeme *axiómu priradenia*. Potom určite platí

$$\{q[z/y_1]\}\ P_3\ \{q\}$$

Ak sa podarí dokázať nasledujúcu implikáciu, bude možné použiť *pravidlo následku* a tým bude induktívna formula $\{r \wedge \neg b\}\ P_3\ \{q\}$ dokázaná.

$$r \wedge \neg b\ \Rightarrow\ q[z/y_1]$$

Z troch hlavných častí programu teda dostávame tri implikácie, ktorých platnosť je nutné dokázať.

$$\begin{aligned} p(x)\ \wedge\ true\ &\Rightarrow\ r[y_1/1, y_2/1] \\ r\ \wedge\ b\ &\Rightarrow\ r[y_1/y_1+1, y_2/(y_1+1)^2] \\ r\ \wedge\ \neg b\ &\Rightarrow\ q[z/y_1] \end{aligned}$$

Musíme sformulovať podmienku r. Podobne ako hľadanie invariantu vo Floydovej metóde, je nájdenie tejto podmienky netriviálny a nedeterministický proces. Je vhodné a vo väčšine prípadov aj úspešné vychádzať z poslednej implikácie a odvodiť hľadanú podmienku r od výstupnej podmienky q.

V našom prípade je podmienka r rovnaká, ako prislúchajúci invariant v ekvivalentnom štandardnom programe dokazovanom Floydovou metódou.

$$r\ \equiv\ y_1^2 = y_2\ \wedge\ (y_1-1)^2 < x$$

Poznáme podmienku b cyklu **while** v časti P_2.

$$b\ \equiv\ y_2 < x$$

Výsledné implikácie teda vyzerajú nasledovne.

$$\begin{aligned} x > 0\ \wedge\ true\ &\Rightarrow\ 1^2 = 1\ \wedge\ (1-1)^2 < x \\ y_1^2 = y_2\ \wedge\ (y_1-1)^2 < x\ \wedge\ y_2 < x\ &\Rightarrow\ (y_1+1)^2 = (y_1+1)^2\ \wedge\ (y_1+1-1)^2 < x \\ y_1^2 = y_2\ \wedge\ (y_1-1)^2 < x\ \wedge\ y_2 \geq x\ &\Rightarrow\ (y_1-1)^2 < x \leq y_1^2 \end{aligned}$$

Samotné dôkazy implikácií sú priamočiare a jednoduché.

Príklad 27 Daný je štruktúrovaný program P.

$$
\begin{array}{lll}
P: & \textbf{begin} & [y_1, y_2] := [0, x_1] \\
 & & \textbf{while } y_2 \geq x_2 \textbf{ do} \\
 & & \quad [y_1, y_2] := [y_1 + 1, y_2 - x_2] \\
 & & \textbf{od} \\
 & \textbf{end} & [z_1, z_2] := [y_1, y_2]
\end{array}
$$

Hoareovou metódou formálne dokážte čiastočnú správnosť programu P vzhľadom na nasledujúce podmienky:

- vstupná podmienka – $p(x_1, x_2) : \ x_1 \geq 0 \ \wedge \ x_2 > 0$
- výstupná podmienka – $q(x_1, x_2, z_1, z_2) : \ z_1 x_2 + z_2 = x_1 \ \wedge \ 0 \leq z_2 < x_2$

Riešenie 27 V predchádzajúcom príklade na dokazovanie čiastočnej správnosti programu P Hoareovou metódou sme použili tzv. postup *zhora dole*. Vychádzali sme z induktívnej formuly $\{p\}P\{q\}$, na ktorú sme postupne aplikovali inferenčné odvodzovacie pravidlá a axiómu priradenia. Týmto spôsobom sme dospeli k niekoľkým implikáciam, ktoré sme dokázali.

Akosi implicitne sme predpokladali, ze dokázaním týchto implikácií sa dokáže aj induktívna formula $\{p\}P\{q\}$ a teda aj čiastočná správnosť programu P. To je samozrejme pravda, no v skutočnosti má táto induktívna formula stáť na konci celého procesu odvodzovania a dokazovania a nie na jeho začiatku. Preto existuje aj spôsob, ako zapísať Hoareovu metódu formálnejšie.

Je nutné zdôrazniť, že oba zápisy sú dobré. Prvý je názornejší, keďže nezačíname ničnehovoriacim tvrdením, ale známou všeobecnou induktívnou formulou. Druhý zápis je zase formálnejší. Nasledujúci dôkaz čiastočnej správnosti Hoareovou metódou zapíšeme formálnejším spôsobom. Tento spôsob je používaný taktiež v skriptách.

Zvolíme si invariant. $R(x_1, x_2, y_1, y_2) : \ y_1 x_2 + y_2 = x_1 \wedge 0 \leq y_2 < x_2$

1. $x_1 \geq 0 \ \wedge \ x_2 > 0 \ \Rightarrow \ R(x_1, x_2, 0, x_1)$
   $0x_2 + x_1 = x_1 \ \wedge \ x_1 \geq 0$

2. $\{R(x_1, x_2, 0, x_1)\}$
   $\quad [y_1, y_2] := [0, x_1]$
   $\{R(x_1, x_2, y_1, y_2)\}$
   axióma priradenia

3. $\{x_1 \geq 0 \ \wedge \ x_2 > 0\}$
   $[y_1, y_2] := [0, x_1]$
   $\{R(x_1, x_2, y_1, y_2)\}$
   pravidlo následku pre 1. a 2.

4. $R(x_1, x_2, y_1, y_2) \ \wedge \ y_2 \geq x_2 \ \Rightarrow \ R(x_1, x_2, y_1 + 1, y_2 - x_2)$
   $y_1 x_2 + y_2 = x_1 \ \wedge \ y_2 \geq 0 \ \wedge \ y_2 \geq x_2 \ \Rightarrow$
   $\Rightarrow \ (y_1 + 1)x_2 + y_2 - x_2 = x_1 \ \wedge \ y_2 - x_2 \geq 0$
   $y_1 x_2 + y_2 = x_1 \ \wedge \ y_2 \geq 0 \ \wedge \ y_2 \geq x_2 \ \Rightarrow \ y_1 x_2 + y_2 = x_1 \ \wedge \ y_2 \geq x_2$

5. $\{R(x_1, x_2, y_1 + 1, y_2 - x_2)\}$
   $[y_1, y_2] := [y_1 + 1, y_2 - x_2]$
   $\{R(x_1, x_2, y_1, y_2)\}$
   axióma priradenia

6. $\{R(x_1, x_2, y_1, y_2) \ \wedge \ y_2 \geq x_2\}$
   $[y_1, y_2] := [y_1 + 1, y_2 - x_2]$
   $\{R(x_1, x_2, y_1, y_2)\}$
   pravidlo následku pre 4. a 5.

7. $\{R(x_1, x_2, y_1, y_2)\}$
   while $y_2 \geq x_2$ **do**
   $[y_1, y_2] := [y_1 + 1, y_2 - x_2]$
   od
   $\{R(x_1, x_2, y_1, y_2) \ \wedge \ y_2 < x_2\}$
   pravidlo iterácie pre 6.

8. $\{x_1 \geq 0 \ \wedge \ x_2 > 0\}$
   $[y_1, y_2] := [0, x_1]$
   while $y_2 \geq x_2$ **do**
   $[y_1, y_2] := [y_1 + 1, y_2 - x_2]$
   od
   $\{R(x_1, x_2, y_1, y_2) \ \wedge \ y_2 < x_2\}$
   pravidlo kompozície pre 3. a 7.

9. $R(x_1, x_2, y_1, y_2) \ \wedge \ y_2 < x_2 \ \Rightarrow \ y_1 x_2 + y_2 = x_1 \ \wedge \ 0 \leq y_2 < x_2$
   $y_1 x_2 + y_2 = x_1 \ \wedge \ y_2 \geq 0 \ \wedge \ y_2 < x_2 \ \Rightarrow \ y_1 x_2 + y_2 = x_1 \ \wedge \ 0 \leq y_2 < x_2$

10. $\{y_1 x_2 + y_2 = x_1 \ \wedge \ 0 \leq y_2 < x_2\}$
    $[z_1, z_2] := [y_1, y_2]$
    $\{z_1 x_2 + z_2 = x_1 \ \wedge \ 0 \leq z_2 < x_2\}$
    axióma priradenia

11. $\{R(x_1, x_2, y_1, y_2) \ \wedge \ y_2 < x_2\}$
    $[z_1, z_2] := [y_1, y_2]$
    $\{z_1 x_2 + z_2 = x_1 \ \wedge \ 0 \leq z_2 < x_2\}$
    pravidlo následku pre 9. a 10.

12. $\{x_1 \geq 0 \ \wedge \ x_2 > 0\}$
    $[y_1, y_2] := [0, x_1]$
    while $y_2 \geq x_2$ **do**
    $[y_1, y_2] := [y_1 + 1, y_2 - x_2]$
    od
    $[z_1, z_2] := [y_1, y_2]$
    $\{z_1 x_2 + z_2 = x_1 \ \wedge \ 0 \leq z_2 < x_2\}$
    pravidlo kompozície pre 8. a 11.

Príklad 28 Uvažujme nasledujúci štruktúrovaný program P.

```
P :  begin  [y1, y2, y3] := [1, 0, 0]
            while y1 ≤ n do
               [y3] := [a[y1]];
               if y3 < 0 then
                  [y3] := [−y3]
               fi;
               [y1, y2] := [y1 + 1, y2 + y3]
            od
     end    [z] := [y2]
```

Hoareovou metódou formálne dokážte čiastočnú správnosť programu P vzhľadom na vstupnú a výstupnú podmienku:

- vstupná podmienka – $p(a, n) : \ n \geq 0$
- výstupná podmienka – $q(a, n, z) : \ z = \sum_{i=1}^{n} |a[i]|$

Riešenie 28 Začíname použitím *pravidla kompozície*.

$$\frac{\{p\} \ P_1 \ \{r\} \quad \{r\} \ P_2 \ \{s\} \quad \{s\} \ P_3 \ \{q\}}{\{p\} \ P \ \{q\}}$$

Postupne dokážeme všetky tri induktívne formuly. Prvú, tretiu a nakoniec druhú, ktorá je najobsiahlejšia.

- Induktívna formula $\{p\}\ P_1\ \{r\}$: Vieme, že podľa *axiómy priradenia* platí tvrdenie $\{r[y_1/1, y_2/0, y_3/0]\}P_1\{r\}$. Musíme teda dokázať nasledujúcu implikáciu.

$$p \ \Rightarrow \ r[y_1/1, y_2/0, y_3/0]$$

- Induktívna formula $\{s\}\ P_3\ \{q\}$: Určite platí $\{q[z/y_2]\}P_3\{q\}$ a tak dokazujeme nasledujúcu implikáciu.

$$s \ \Rightarrow \ q[z/y_2]$$

- Induktívna formula $\{r\}\ P_2\ \{s\}$: Časť P_2 programu P obsahuje cyklus **while**, pre ktorý je možné použiť *pravidlo iterácie*. Ešte pred tým si však musíme upraviť induktívnu formulu *pravidlom následku*.

$$\frac{\{r\}\ \textbf{while}\ b\ \textbf{do}\ P_2'\ \textbf{od}\ \{r \wedge \neg b\} \quad (r \wedge \neg b \Rightarrow s)}{\{r\}\ \textbf{while}\ b\ \textbf{do}\ P_2'\ \textbf{od}\ \{s\}}$$

$$\frac{\{r \wedge b\}\ P_2'\ \{r\}}{\{r\}\ \textbf{while}\ b\ \textbf{do}\ P_2'\ \textbf{od}\ \{r \wedge \neg b\}}$$

Pre vnútorný príkaz P_2' cyklusu **while** použijeme *pravidlo kompozície*, pretože sa skladá z troch osobitných častí.

$$\frac{\{r \wedge b\}\ P_{21}\ \{f\} \quad \{f\}\ P_{22}\ \{g\} \quad \{g\}\ P_{23}\ \{r\}}{\{r \wedge b\}\ P_2'\ \{r\}}$$

Časti P_{21} a P_{23} sú reprezentované priradeniami. Určite platia tvrdenia $\{f[y_3/a[y_1]]\}P_{21}\{f\}$ a $\{r[y_1/y_1+1, y_2/y_2+y_3]\}P_{23}\{r\}$, preto musíme dokázať nasledujúce implikácie.

$$\begin{aligned} r \wedge b &\Rightarrow f[y_3/a[y_1]] \\ g &\Rightarrow r[y_1/y_1+1, y_2/y_2+y_3] \end{aligned}$$

Zostávajúcu časť P_{22} tvorí riadiaca štruktúra **if** neobsahujúca vetvu **else**. Pre tento účel sa používa upravené *pravidlo alternatívy*, tzv. *pravidlo pol alternatívy*.

$$\frac{\{f \wedge c\}\ P_{221}\ \{g\} \quad (f \wedge \neg c \Rightarrow g)}{\{f\}\ \textbf{if}\ c\ \textbf{then}\ P_{221}\ \textbf{fi}\ \{g\}}$$

Nakoniec *axiómou priradenia* aplikovanou na časť P_{221} dostávame poslednú implikáciu.

$$f \wedge c \;\Rightarrow\; g[y_3/-y_3]$$

Pre dokázanie čiastočnej správnosti programu P je teda nutné sformulovať podmienky r, s, f a g v nasledujúcich implikáciach a tieto implikácie dokázať.

$$\begin{aligned} p &\Rightarrow r[y_1/1, y_2/0, y_3/0] \\ s &\Rightarrow q[z/y_2] \\ r \wedge \neg b &\Rightarrow s \\ r \wedge b &\Rightarrow f[y_3/a[y_1]] \\ g &\Rightarrow r[y_1/y_1+1, y_2/y_2+y_3] \\ f \wedge \neg c &\Rightarrow g \\ f \wedge c &\Rightarrow g[y_3/-y_3] \end{aligned}$$

Po sformulovaní podmienok r, s, f a g a následnom dokázaní všetkých uvedených implikácií je čiastočná správnosť programu P vzhľadom na vstupnú podmienku p a výstupnú podmienku q dokázaná. Kompletný dôkaz je prenechaný ako cvičenie pre čitateľa.

2.2 Rozširovanie Hoareovských kalkulov

Príklad 29 Sformulujte inferenčné pravidlo Hoareovského kalkulu pre riadiacu štruktúru **repeat** definovanú nasledujúcim vzťahom.

$$(\textbf{repeat } S \textbf{ until } b) \;\equiv\; (S;\ \textbf{while } \neg b \textbf{ do } S \textbf{ od})$$

Dokážte, že navrhnuté inferenčné pravidlo je zdravé.

Riešenie 29 Riadiaca štruktúra **repeat**, dobre známa napríklad z programovacieho jazyka Pascal, sa dá jednoducho prepísať pomocou riadiacej štruktúry **while** tak, ako je to zobrazené v zadaní úlohy.

$$\frac{\{p\}\ S;\ \textbf{while } \neg b \textbf{ do } S \textbf{ od}\ \{q\}}{\{p\}\ \textbf{repeat } S \textbf{ until } b\ \{q\}} \qquad \begin{matrix}(1)\\(2)\end{matrix}$$

Ak dokážeme tvrdenie (1), budeme mať dokázané aj tvrdenie (2). Obdobný postup budeme aplikovať aj v ďalších odvodzovaniach rozširovaní Hoareovských kalkulov za pomoci štyroch inferenčných pravidiel.

Tvrdenie (1) sa skladá z dvoch častí. Použijeme *pravidlo kompozície*.

$$\frac{\{p\}\ S\ \{r\} \quad \{r\}\ \textbf{while}\ \neg b\ \textbf{do}\ S\ \textbf{od}\ \{q\}}{\{p\}\ S;\ \textbf{while}\ \neg b\ \textbf{do}\ S\ \textbf{od}\ \{q\}}$$

Pre cyklus **while** existuje *pravidlo iterácie*. Pravidlo však vyžaduje výstupnú podmienku v konkrétnom tvare. Na dosiahnutie žiadaného tvaru použijeme *pravidlo následku*.

$$\frac{\{r\}\ \textbf{while}\ \neg b\ \textbf{do}\ S\ \textbf{od}\ \{r \wedge b\} \quad (r \wedge b \Rightarrow q)}{\{r\}\ \textbf{while}\ \neg b\ \textbf{do}\ S\ \textbf{od}\ \{q\}}$$

Teraz je už možné použiť zmieňované *pravidlo iterácie* pre cyklus **while**.

$$\frac{\{r \wedge \neg b\}\ S\ \{r\}}{\{r\}\ \textbf{while}\ \neg b\ \textbf{do}\ S\ \textbf{od}\ \{r \wedge b\}}$$

Nakoniec sformulujeme inferenčné pravidlo pre riadiacu štruktúru **repeat**.

$$\frac{\{p\}\ S\ \{r\} \quad \{r \wedge \neg b\}\ S\ \{r\} \quad (r \wedge b \Rightarrow q)}{\{p\}\ \textbf{repeat}\ S\ \textbf{until}\ b\ \{q\}}$$

Výsledné inferenčné pravidlo je zdravé, pretože pri jeho odvodzovaní sme používali už existujúce inferenčné pravidlá Hoareovského dokazovacieho systému, ktoré sú zdravé.

Príklad 30 Navrhnite inferenčné pravidlo Hoareovho dokazovacieho systému pre riadiacu štruktúru reprezentujúcu tzv. jeden a pol cyklus.

do
  S_1;
  exit when b;
  S_2
od

resp.

$$(\textbf{loop}\ S_1;\ \textbf{when}\ b\ \textbf{exit};\ S_2\ \textbf{pool}) \equiv (S_1;\ \textbf{while}\ \neg b\ \textbf{do}\ S_2;\ S_1\ \textbf{od})$$

Riešenie 30 Podobne ako v predchádzajúcom príklade budeme používať už existujúce inferenčné pravidlá Hoareovského kalkulu pre dokázanie žiadaného tvrdenia.

$$\frac{\{p\}\ S_1;\ \textbf{while}\ \neg b\ \textbf{do}\ S_2;\ S_1\ \textbf{od}\ \{q\}}{\{p\}\ \textbf{loop}\ S_1;\ \textbf{when}\ b\ \textbf{exit};\ S_2\ \textbf{pool}\ \{q\}}$$

- *pravidlo kompozície*

$$\frac{\{p\}\ S_1\ \{r\} \quad \{r\}\ \textbf{while}\ \neg b\ \textbf{do}\ S_2;\ S_1\ \textbf{od}\ \{q\}}{\{p\}\ S_1;\ \textbf{while}\ \neg b\ \textbf{do}\ S_2;\ S_1\ \textbf{od}\ \{q\}}$$

- *pravidlo následku*

$$\frac{\{r\}\ \textbf{while}\ \neg b\ \textbf{do}\ S_2;\ S_1\ \textbf{od}\ \{r \wedge b\} \quad (r \wedge b \Rightarrow q)}{\{r\}\ \textbf{while}\ \neg b\ \textbf{do}\ S_2;\ S_1\ \textbf{od}\ \{q\}}$$

- *pravidlo iterácie*

$$\frac{\{r \wedge \neg b\}\ S_2;\ S_1\ \{r\}}{\{r\}\ \textbf{while}\ \neg b\ \textbf{do}\ S_2;\ S_1\ \textbf{od}\ \{r \wedge b\}}$$

- *pravidlo kompozície*

$$\frac{\{r \wedge \neg b\}\ S_2\ \{s\} \quad \{s\}\ S_1\ \{r\}}{\{r \wedge \neg b\}\ S_2;\ S_1\ \{r\}}$$

Na záver sformulujeme výsledné inferenčné pravidlo.

$$\frac{\{p\}\ S_1\ \{r\} \quad \{r \wedge \neg b\}\ S_2\ \{s\} \quad \{s\}\ S_1\ \{r\} \quad (r \wedge b \Rightarrow q)}{\{p\}\ \textbf{loop}\ S_1;\ \textbf{when}\ b\ \textbf{exit};\ S_2\ \textbf{pool}\ \{q\}}$$

Príklad 31 Sformulujte inferenčné pravidlo Hoareovského kalkulu pre programovú konštrukciu P_K.

$$\begin{array}{ll} P_K: & \textbf{while}\ c\ \textbf{do}\ S\ \textbf{od};\\ & \textbf{while}\ b\ \textbf{do}\\ & \quad S;\\ & \quad \textbf{while}\ c\ \textbf{do}\ S\ \textbf{od};\\ & \textbf{od} \end{array}$$

Riešenie 31 Inferenčné pravidlo pre programovú konštrukciu P_K vyzerá nasledovne. Podrobné odvodenie je prenechané ako cvičenie pre čitateľa.

$$\frac{\{p \wedge c\}\ S\ \{p\}\quad \{s \wedge b\}\ S\ \{r\}\quad \{r \wedge c\}\ S\ \{r\}\quad (p \wedge \neg c \Rightarrow s)\quad (r \wedge \neg c \Rightarrow s)\quad (s \wedge \neg b \Rightarrow q)}{\{p\}\ \textbf{while}\ c\ \textbf{do}\ S\ \textbf{od};\ \textbf{while}\ b\ \textbf{do}\ S;\ \textbf{while}\ c\ \textbf{do}\ S\ \textbf{od};\ \textbf{od}\ \{q\}}$$

Príklad 32 Predpokladajme, že platí nasledujúca formula.

$$\{p\ \wedge\ (b\ \vee\ c)\}\ S\ \{p\}$$

Dokážte Hoareovou metódou čiastočnú správnosť programovej konštrukcie P_K z predchádzajúceho príkladu vzhľadom na vstupnú podmienku $\{p\}$ a výstupnú podmienku $\{p \wedge (\neg b \wedge \neg c)\}$.

Riešenie 32 Pre vyriešenie úlohy stačí dokázať nasledujúci vzťah.[1]

$$\frac{\{p\ \wedge\ (b\ \vee\ c)\}\ S\ \{p\}}{\{p\}\ P_K\ \{p\ \wedge\ \neg b\ \wedge\ \neg c\}}$$

[1] V skutočnosti je to však dosť zložité. Komplexné riešenie úlohy je vítané a bude zaradené do *Zbierky riešených úloh zo ZTP*.

Kapitola 3

Sémantika programov

3.1 Základy sémantiky

Príklad 33 Uvažujme hypotetický iteratívny príkaz **loop** (b, S_1, S_2) definovaný sémantickou rovnicou

$$\mathcal{M}[\![\textbf{loop } (b, S_1, S_2)]\!] \;=\; \bigsqcup_0^\infty \{\phi_i\}$$

kde

$$\begin{array}{rcl} \phi_0 & = & \lambda\sigma \cdot \bot \\ \phi_{i+1} & = & \lambda\sigma \cdot \textbf{if } \mathcal{W}[\![b]\!]\sigma \textbf{ then } \phi_i(\mathcal{M}[\![S_1]\!]\sigma) \\ & & \qquad\qquad \textbf{else } \mathcal{M}[\![S_2]\!]\sigma \end{array}$$

Na základe základných príkazov (priradenie, kompozícia, vetvenie a cyklus) definujte programový segment S taký, že platí

$$\mathcal{M}[\![\textbf{loop } (b, S_1, S_2)]\!] \;=\; \mathcal{M}[\![S]\!]$$

Tvrdenie dokážte.

Riešenie 33 Analýzou sémantickej rovnice v zadaní vieme vytvoriť programový segment S skladajúci sa z častí S_0 a S_2.

$$S: \left.\begin{array}{l} \textbf{while } b \textbf{ do} \\ \quad S_1 \\ \textbf{od} \end{array}\right\} S_0 \\ S_2;$$

Teda pre programový subsegment S_0 platí

$$\mathcal{M}[\![\textbf{while } b \textbf{ do } S_1 \textbf{ od}]\!] \;=\; \bigsqcup_0^\infty\{\psi_i\}$$

kde

$$\begin{array}{rcl} \psi_0 & = & \lambda\sigma \cdot \bot \\ \psi_{i+1} & = & \lambda\sigma \cdot \textbf{if } \mathcal{W}[\![b]\!]\sigma \textbf{ then } \psi_i(\mathcal{M}[\![S_1]\!]\sigma) \\ & & \qquad\qquad \textbf{else } \sigma \end{array}$$

Keďže $\mathcal{M}[\![S_2]\!]\sigma'$ potom pre programový segment S platí

$$\mathcal{M}[\![S_0; S_2]\!]\sigma \;=\; \lambda\sigma \;\cdot\; \mathcal{M}[\![S_2]\!](\mathcal{M}[\![S_0]\!]\sigma)$$

Našli sme teda programový segment S. Teraz musíme dokázať ekvivalenciu s iteratívnym príkazom **loop** (b, S_1, S_2).

$$\begin{array}{rcl} \mathcal{M}[\![S]\!] & = & \lambda\sigma \;\cdot\; \mathcal{M}[\![S_2]\!](\mathcal{M}[\![S_0]\!]\sigma) \\ & = & \lambda\sigma \;\cdot\; \mathcal{M}[\![S_2]\!](\mathcal{M}[\![\textbf{while } b \textbf{ do } S_1 \textbf{ od}]\!]\sigma) \\ & = & \lambda\sigma \;\cdot\; \mathcal{M}[\![S_2]\!](\sqcup_0^\infty\{\psi_i\}\sigma) \end{array}$$

Čiže musíme dokázať nasledujúce tvrdenie.

$$\mathcal{M}[\![S_2]\!](\bigsqcup_0^\infty\{\psi_i\}) \;=\; \bigsqcup_0^\infty\{\phi_i\}$$

Rovnosť rozložíme na dva možné prípady.

1. Nech $\sqcup_0^\infty\{\phi_i\} = \bot$. To znamená, že $\forall\phi_i = \bot$. Potom $\mathcal{M}[\![S_2]\!](\sqcup_0^\infty\{\psi_i\}) = \bot$. Kedy bude ϕ_i $\bot$? Ak $i = 0$ alebo $\forall i > 0 : \mathcal{W}[\![b]\!]\sigma = true$. Rovnako to platí aj pri ψ. Takže $\mathcal{M}[\![S_2]\!]$ $\bot = \bot$.

2. Nech $\sqcup_0^\infty\{\phi_i\} \neq \bot$. Potom $i \neq 0$ resp. $i > 0$. Určite $\exists k : k < i$ také, že k-krát bolo $\mathcal{W}[\![b]\!]$ $true$ a na $k+1$ bolo $false$. Teda $\sqcup_0^\infty$ bola $\mathcal{M}[\![S_2]\!]\sigma$.

   Pozrieme sa na ψ. Vieme, že k-krát bude $true$, potom $false$. Vykonal sa teda rovnaký kód ako pri ϕ. $\sqcup_0^\infty\{\psi_i\}$ bola σ. Takže $\sqcup_0^\infty$ ľavej strany priradenia je $\mathcal{M}[\![S_2]\!]\sigma$.

3.2 Sémantika rekuzívnych programov

Definícia 14 Definujme množinu termov $E_{xp} = \{t, t_i, \ldots\}$ resp. $B_{exp} = \{b, b_i, \ldots\}$, d'alej $X \subseteq E_{xp}$, $F^0 \subseteq E_{xp}$, $B^0 \subseteq B_{exp}$.

- ak $f_F \in F^0, t_1 ... t_n \in E_{xp}$, potom $f_F(t_1 ... t_n) \in E_{xp}$
- ak $p \in B^n, t_1 ... t_n \in E_{xp}$, potom $p(t_1 ... t_n) \in B_{exp}$
- ak $b \in B_{exp}, t_1, t_2 \in E_{xp}$, potom **if** b **then** t_1 **else** t_2 **fi** $\in E_{xp}$
- ak $\phi \in \Phi^n, t_1 ... t_n \in E_{xp}$, potom $\phi(t_1 ... t_n) \in E_{xp}$

Definícia 15 Definujme syntax rekurzívnej definície:

$$\phi(x_1 ... x_n) \Longleftarrow t[\phi](x_1 ... x_n)$$

Definícia 16 Definícia programu so systémom rekurzívnych definícií:

$$\begin{array}{rcl} \phi_1(x) & \Longleftarrow & t_1[\phi_1 ... \phi_n](\overline{x}) \\ \phi_2(x) & \Longleftarrow & t_2[\phi_1 ... \phi_n](\overline{x}) \\ & \ldots & \\ \phi_n(x) & \Longleftarrow & t_n[\phi_1 ... \phi_n](\overline{x}) \end{array}$$

Veta 1 (Kleene) Každý spojitý funkcionál τ má jediný najmenší pevný bod

$$f_\tau = \sqcup_0^\infty \tau^i[\Omega]$$

kde $\tau^0[\Omega] = \Omega$ a taktiež $\tau^{i+1}[\Omega] = \tau[\tau^i[\Omega]]$.

Dôsledok 1.1 Kleeneho veta nám teda dáva návod ako nájst' riešenie rekurzívnej rovnice resp. pevný bod rekurzívneho programu.

Konštrukcia riešenia je nasledovná:

1. zadefinujeme postupnost' aproximácií $\tau^i[\Omega]$ pre všetky $i \geq 0$
2. zostrojíme najmenšiu hornú hranicu ret'azca $\sqcup_0^\infty \tau^i[\Omega]$

Príklad 34 Majme program:

```
int f(int x)
{
    if (x == 0) then return 0;
    else return x + f(x − 1);
}
```

Nájdite jeho pevný bod.

Riešenie 34 Program počíta prvých x čísiel. Prepíšeme ho do systému rekurzívnej definície:

$$\tau[\phi](x) : \textbf{if } x = 0 \textbf{ then } 0 \textbf{ else } x + \phi(x-1) \textbf{ fi}$$

Nájdime jeho pevný bod. Keďže je funkcionál spojitý, môžeme použiť Kleeneho vetu:

$$\begin{aligned}
\tau^0[\Omega] &\equiv \Omega \\
\tau^1[\Omega] &\equiv \textbf{if } x = 0 \textbf{ then } 0 \textbf{ else } x + \tau^0[\Omega](x-1) \textbf{ fi} \equiv \\
&\equiv \textbf{if } x = 0 \textbf{ then } 0 \textbf{ else } \bot \textbf{ fi} \\
\tau^2[\Omega] &\equiv \textbf{if } x = 0 \textbf{ then } 0 \textbf{ else } x + \tau^1[\Omega](x-1) \textbf{ fi} \equiv \\
&\equiv \textbf{if } x = 0 \textbf{ then } 0 \textbf{ else } x + (\textbf{if } x - 1 = 0 \textbf{ then } 0 \textbf{ else } \bot \textbf{ fi}) \textbf{ fi} \\
\tau^3[\Omega] &\equiv \textbf{if } x = 0 \textbf{ then } 0 \textbf{ else } x + \tau^2[\Omega](x-1) \textbf{ fi} \equiv \\
&\equiv \textbf{if } x = 0 \textbf{ then } 0 \\
&\quad \textbf{else } x + (\textbf{if } x - 1 = 0 \textbf{ then } 0 \textbf{ else } x - 1 + (\textbf{if } x - 1 = 1 \textbf{ then } 0 \textbf{ else } \bot \textbf{ fi}) \textbf{ fi}) \textbf{ fi} \equiv \\
&\equiv \textbf{if } x = 0 \textbf{ then } 0 \\
&\quad \textbf{else } x + (\textbf{if } x = 1 \textbf{ then } 0 \textbf{ else } x - 1 + (\textbf{if } x = 2 \textbf{ then } 0 \textbf{ else } \bot \textbf{ fi}) \textbf{ fi}) \textbf{ fi} \\
&\dots
\end{aligned}$$

Nultý krok je podľa definície. V prvom kroku sme len upravili podmienku $x-1 = 0$ na $x = 1$, čo je to isté. Nič iné nie je možné v tomto kroku vykonať.

V druhom kroku je jednoduché dostať sa k výsledku, ale napriek tomu sa nad ním zamyslíme, čo to znamená. Je to vlastne náš program definovaný pre $x = 0$ a $x = 1$. Pre ostatné hodnoty je nedefinovaný.

Kroky tri je obdobný ako krok dva, pričom náš program je už samozrejme definovaný aj pre $x = 2$.

Čo sa vlastne pri aproximácii deje? Pri k-tej aproximácii by sme mohli s výsledným funkcionálom vypočítať sumu prvých i čísiel, pre také $i < k$. Teraz už vlastne stačí zostrojiť najmenšiu hornú hranicu tých aproximácií. A tou je pevný bod

$$f_\tau = \frac{x(x+1)}{2}$$

Príklad 35 Majme nasledovnú jednoduchú fukciu:

$$\begin{aligned} f_a(0) &= 1 \\ f_a(x) &= a \cdot f_a(x-1) \end{aligned}$$

Nájdite pevný bod.

Riešenie 35 Najskôr to prepíšeme na funkcionál:

$$\phi(x) \equiv \textbf{if } x = 0 \textbf{ then } 1 \textbf{ else } a \cdot \phi(x-1) \textbf{ fi}$$

Teraz hľadajme pevný bod:

$$\begin{aligned} \tau^0[\Omega] &\equiv \Omega \\ \tau^1[\Omega] &\equiv \textbf{if } x = 0 \textbf{ then } 1 \\ &\quad\ \textbf{else } \bot \textbf{ fi} \\ \tau^2[\Omega] &\equiv \textbf{if } x = 0 \textbf{ then } 1 \\ &\quad\ \textbf{else } a \cdot (\textbf{if } x = 1 \textbf{ then } 1 \textbf{ else } \bot \textbf{ fi}) \textbf{ fi} \\ \tau^3[\Omega] &\equiv \textbf{if } x = 0 \textbf{ then } 1 \\ &\quad\ \textbf{else } a \cdot (\textbf{if } x = 1 \textbf{ then } 1 \textbf{ else } a \cdot (\textbf{if } x = 2 \textbf{ then } 1 \textbf{ else } \bot \textbf{ fi}) \textbf{ fi}) \textbf{ fi} \\ &\ldots \end{aligned}$$

Po rozpísaní štvrtého člena už môžeme vidieť ako to pekne aproximuje pevný bod. Ten zapíšeme nasledovne:

$$f_p(x) \equiv \textbf{if } x = 0 \textbf{ then } 1 \textbf{ else } a^x \textbf{ fi}$$

Príklad 36 Máme funkciu:

$$\begin{aligned} f_0 &= 0 \\ f(n) &= f(f(n-1)) \end{aligned}$$

Nájdite pevný bod tejto funkcie.

Riešenie 36 Funkciu si najskôr rozpíšeme:

$$\begin{aligned} f(1) &= f(f(0)) = f(0) = 0 \\ f(2) &= f(f(1)) = f(f(f(0))) = f(f(0)) = f(0) = 0 \\ f(3) &= f(f(2)) = f(f(f(1))) = f(f(f(f(0)))) = f(f(f(0))) = f(f(0)) = f(0) = 0 \end{aligned}$$

Teraz ju zapíšeme v tvare:

$$\phi(x) : \textbf{if } x = 0 \textbf{ then } 0 \textbf{ else } \phi(\phi(x-1)) \textbf{ fi}$$

Hľadajme pevný bod:

$$\begin{aligned} \tau^0 &\equiv \Omega \\ \tau^1 &\equiv \textbf{if } x = 0 \textbf{ then } 0 \textbf{ else } \bot \textbf{ fi} \\ \tau^2 &\equiv \textbf{if } x = 0 \textbf{ then } 0 \textbf{ else } (\textbf{if } [\textbf{if } x = 1 \textbf{ then } 0 \textbf{ else } \bot \textbf{ fi}] = 0 \textbf{ then } 0 \textbf{ else } \bot \textbf{ fi}) \textbf{ fi} \\ \tau^3 &\equiv \textbf{if } x = 0 \\ &\quad \textbf{then } 0 \\ &\quad \textbf{else if } [\textbf{if } x = 1 \textbf{ then } 0 \textbf{ else } (\textbf{if } x = 2 \textbf{ then } 0 \textbf{ else } \bot \textbf{ fi}) \textbf{ fi}] = 0 \\ &\qquad \textbf{then } 0 \\ &\qquad \textbf{else if } \left[\begin{array}{l} \textbf{if } \left[\begin{array}{l} \textbf{if } x = 1 \\ \textbf{then } 0 \\ \textbf{else } (\textbf{if } x = 2 \textbf{ then } 0 \textbf{ else } \bot \textbf{ fi}) \\ \textbf{fi} \end{array}\right] = 1 \\ \textbf{then } 0 \\ \textbf{else } \bot \\ \textbf{fi} \end{array}\right] = 0 \\ &\qquad\quad \textbf{then } 0 \\ &\qquad\quad \textbf{else } \bot \\ &\qquad\quad \textbf{fi} \\ &\qquad \textbf{fi} \\ &\quad \textbf{fi} \end{aligned}$$

Dosad'me do τ^3 hodnotu 1:

$$\begin{array}{rcl}
\tau^3\{x=1\} & \equiv & \textbf{if } x=0 \\
 & & \textbf{then } 0 \\
 & & \textbf{else if } 0=0 \\
 & & \quad \textbf{then } 0 \\
 & & \quad \textbf{else } (\textbf{if } [\textbf{if } \bot = 1 \textbf{ then } 0 \textbf{ else } \bot \textbf{ fi}] = 0 \textbf{ then } 0 \textbf{ else } \bot \textbf{ fi}) \\
 & & \quad \textbf{fi} \\
 & & \textbf{fi} \\
\tau^3\{x=1\} & \equiv & \textbf{if } x=0 \textbf{ then } 0 \textbf{ else } 0 \textbf{ fi}
\end{array}$$

Dosad'me do τ^3 tiež hodnotu 2:

$$\begin{array}{rcl}
\tau^3\{x=2\} & \equiv & \textbf{if } x=0 \\
 & & \textbf{then } 0 \\
 & & \textbf{else if } 0=0 \\
 & & \quad \textbf{then } 0 \\
 & & \quad \textbf{else } (\textbf{if } [\textbf{if } 0 = 1 \textbf{ then } 0 \textbf{ else } \bot \textbf{ fi}] = 0 \textbf{ then } 0 \textbf{ else } \bot \textbf{ fi}) \\
 & & \quad \textbf{fi} \\
 & & \textbf{fi} \\
\tau^3\{x=2\} & \equiv & \textbf{if } x=0 \textbf{ then } 0 \textbf{ else } 0 \textbf{ fi}
\end{array}$$

$$\begin{array}{rcl}
\tau^3\{x>2\} & \equiv & \textbf{if } x=0 \\
 & & \textbf{then } 0 \\
 & & \textbf{else if } \bot = 0 \\
 & & \quad \textbf{then } 0 \\
 & & \quad \textbf{else } (\textbf{if } \bot = 1 \textbf{ then } 0 \textbf{ else } \bot \textbf{ fi}) \\
 & & \quad \textbf{fi} \\
 & & \textbf{fi} \\
\tau^3\{x>2\} & \equiv & \textbf{if } x=0 \textbf{ then } 0 \textbf{ else } \bot \textbf{ fi}
\end{array}$$

Skúsme teraz τ^4:

$$\begin{array}{rcl}
\tau^4 & \equiv & \textbf{if } x=0 \\
 & & \textbf{then } 0 \\
 & & \textbf{else } (\textbf{if } [\textbf{if } x = 1 \textbf{ then } 0 \textbf{ else } 0 \textbf{ fi}] = 0 \textbf{ then } 0 \textbf{ else } 0 \textbf{ fi}) \\
 & & \textbf{fi} \\
\tau^4 & \equiv & \textbf{if } x=0 \textbf{ then } 0 \textbf{ else } 0 \textbf{ fi}
\end{array}$$

Vyzerá to tak, že pre $x \geq 0$ je to 0, inak $\bot$. A to už je pevný bod f_P:

$$f_P \equiv \textbf{if } x \geq 0 \textbf{ then } 0 \textbf{ else } \bot \textbf{ fi}$$

Príklad 37 Uvažujme rovnakú funkciu f ako v predchádzajúcom príklade. Nech $g(x)$ je jej pevný bod. Ukážte, že je silne ekvivalentný s $f_P(x)$.

Riešenie 37 Fixpointovou indukciou dokážeme, že pre $x \geq 0$:

$$f_P(x) \sqsubseteq g(x)$$

Označme

$$g(x) \equiv \textbf{if } x \geq 0 \textbf{ then } 0 \textbf{ else } \bot \textbf{ fi}$$

Podľa fixpointovej indukcie stačí dokázať, že $\tau[g] \sqsubseteq g$, čiže:

$$\left(\begin{array}{l} \textbf{if } x = 0 \\ \textbf{then } 0 \\ \textbf{else } (\textbf{if } [\textbf{if } x - 1 \geq 0 \textbf{ then } 0 \textbf{ else } \bot \textbf{ fi}] \geq 0 \textbf{ then } 0 \textbf{ else } \bot \textbf{ fi}) \\ \textbf{fi} \end{array}\right) \sqsubseteq \left(\begin{array}{l} \textbf{if } x = 0 \\ \textbf{then } 0 \\ \textbf{else } \bot \\ \textbf{fi} \end{array}\right)$$

Rozdelíme to na nasledujúce prípady:

1. $x = 0$

$$\begin{array}{rcl} \tau[g](x) & = & 0 \\ \tau[g](x) & = & g(x) \end{array}$$

2. $x - 1 \geq 0 \equiv x \geq 1$

$$\begin{array}{rcl} \tau[g](x) & = & \textbf{if } x = 0 \\ & & \textbf{then } 0 \\ & & \textbf{else } (\textbf{if } 0 \geq 0 \textbf{ then } 0 \textbf{ else } \bot \textbf{ fi}) \\ & & \textbf{fi} \\ \tau[g](x) & = & \textbf{if } x = 0 \textbf{ then } 0 \textbf{ else } 0 \textbf{ fi} \\ \tau[g](x) & = & g(x) \end{array}$$

Teda platí $f_P \sqsubseteq g$. A navyše pre $x < 0$ platí $g(x) \sqsubseteq f_P(x)$, pretože $g(x) = \bot$. Celkovo platí

$$(g \sqsubseteq f_P) \wedge (f_P \sqsubseteq g) \implies f_P \equiv g$$

Teda g je najmenší pevný bod.

Príklad 38 Máme funkciu:

$$\phi(x) : \textbf{if } x > 0 \textbf{ then } \phi(\phi(x-1)) \textbf{ else } x \textbf{ fi}$$

pre $x \in \mathbb{N}$.

Nájdite pevný bod tejto funkcie.

Riešenie 38 Najskôr si rozpíšeme čo to vlastne robí:

$$\begin{array}{rcl}
\phi(0) & = & \textbf{if } 0 > 0 \textbf{ then } \bot \textbf{ else } 0 \textbf{ fi} \\
& = & 0 \\
\phi(1) & = & \textbf{if } 1 > 0 \textbf{ then } \phi(0) \textbf{ else } 1 \textbf{ fi } = \\
& = & \textbf{if } 1 > 0 \textbf{ then } 0 \textbf{ else } 1 \textbf{ fi } = \\
& = & 0 \\
\phi(2) & = & \textbf{if } 2 > 0 \textbf{ then } \phi(1) \textbf{ else } 2 \textbf{ fi } = \\
& = & \textbf{if } 2 > 0 \textbf{ then } (\textbf{if } 1 > 0 \textbf{ then } 0 \textbf{ else } 1 \textbf{ fi}) \textbf{ else } 2 \textbf{ fi } = \\
& = & \textbf{if } 2 > 0 \textbf{ then } 0 \textbf{ else } 2 \textbf{ fi } = \\
& = & 0
\end{array}$$

Riešenie:

$$\begin{array}{rcl}
\tau^0 & \equiv & \Omega \\
\tau^1 & \equiv & \textbf{if } x > 0 \textbf{ then } \bot \textbf{ else } x \textbf{ fi} \\
\tau^2 & \equiv & \textbf{if } x > 0 \\
 & & \textbf{then if } [\textbf{if } x > 1 \textbf{ then } \bot \textbf{ else } x - 1 \textbf{ fi}] > 0 \\
 & & \quad \textbf{then } \bot \\
 & & \quad \textbf{else } (\textbf{if } x > 1 \textbf{ then } \bot \textbf{ else } x - 1 \textbf{ fi}) \\
 & & \quad \textbf{fi} \\
 & & \textbf{else } ? \\
 & & \textbf{fi} \\
\tau^2\{x > 1\} & \equiv & \textbf{if } x > 0 \textbf{ then } (\textbf{if } \bot > 0 \textbf{ then } \bot \textbf{ else } \bot \textbf{ fi}) \textbf{ else } x \textbf{ fi } \equiv \\
 & \equiv & \textbf{if } x > 0 \textbf{ then } \bot \textbf{ else } x \textbf{ fi } \equiv \\
 & \equiv & \tau^1 \\
\tau^2\{x \leq 1\} & \equiv & \textbf{if } x > 0 \textbf{ then } (\textbf{if } x > 1 \textbf{ then } \bot \textbf{ else } x - 1 \textbf{ fi}) \textbf{ else } x \textbf{ fi } \equiv \\
 & \equiv & \textbf{if } x > 0 \textbf{ then } x - 1 \textbf{ else } x \textbf{ fi}
\end{array}$$

A teraz pre τ^3:

$$
\begin{array}{lcl}
\tau^3 & \equiv & \textbf{if } x > 0 \\
 & & \textbf{then if } [\textbf{if } x > 1 \textbf{ then } x - 2 \textbf{ else } x - 1 \textbf{ fi}] > 0 \\
 & & \quad \textbf{then } [\textbf{if } x > 1 \textbf{ then } x - 2 \textbf{ else } x - 1 \textbf{ fi}] - 1 \\
 & & \quad \textbf{else } (\textbf{if } x > 1 \textbf{ then } x - 2 \textbf{ else } x - 1 \textbf{ fi}) \\
 & & \quad \textbf{fi} \\
 & & \textbf{else } x \\
 & & \textbf{fi} \\
\tau^3\{x > 1\} & \equiv & \textbf{if } x > 0 \textbf{ then } (\textbf{if } x > 2 \textbf{ then } x - 3 \textbf{ else } x - 2 \textbf{ fi}) \textbf{ else } x \textbf{ fi} \\
\tau^3\{x \leq 1\} & \equiv & \textbf{if } x > 0 \textbf{ then } (\textbf{if } x > 1 \textbf{ then } x - 2 \textbf{ else } x - 1 \textbf{ fi}) \textbf{ else } x \textbf{ fi} \equiv \\
 & \equiv & \textbf{if } x > 0 \textbf{ then } x - 2 \textbf{ else } x \textbf{ fi}
\end{array}
$$

Vznikli nám tri možnosti τ_1^3, τ_2^3 a τ_3^3:

$$
\begin{array}{lcl}
\tau_1^3 & \equiv & \textbf{if } x > 0 \textbf{ then } (\textbf{if } x > 2 \textbf{ then } x - 3 \textbf{ else } x - 2 \textbf{ fi}) \textbf{ else } x \textbf{ fi} \\
\tau_2^3\{x > 2\} & \equiv & \textbf{if } x > 0 \textbf{ then } x - 3 \textbf{ else } x \textbf{ fi } \equiv \tau_1^3 \\
\tau_2^3\{x \leq 2\} & \equiv & \textbf{if } x > 0 \textbf{ then } x - 2 \textbf{ else } x \textbf{ fi } \equiv \tau_1^3 \\
\tau_3^3 & \equiv & \textbf{if } x > 0 \textbf{ then } x - 2 \textbf{ else } x \textbf{ fi}
\end{array}
$$

Vidíme, že prípad τ_2^3 je totožný s τ_1^3. Takže ďalej nám stačí rozvinúť len možnosti τ_1^3 a τ_3^3. Začneme s možnosťou $\tau[\tau_3^3]$:

$$
\begin{array}{lcl}
\tau^4 \equiv \tau[\tau_3^3] & \equiv & \textbf{if } x > 0 \\
 & & \textbf{then if } [\textbf{if } x > 1 \textbf{ then } x - 3 \textbf{ else } x - 1 \textbf{ fi}] > 0 \\
 & & \quad \textbf{then } [\textbf{if } x > 1 \textbf{ then } x - 3 \textbf{ else } x - 1 \textbf{ fi}] - 2 \\
 & & \quad \textbf{else } (\textbf{if } x > 1 \textbf{ then } x - 3 \textbf{ else } x - 1 \textbf{ fi}) \\
 & & \quad \textbf{fi} \\
 & & \textbf{else } x \\
 & & \textbf{fi}
\end{array}
$$

Čiže konkrétne vyjadrené:

$$
\begin{array}{lcl}
\tau^4\{x > 1\} & \equiv & \textbf{if } x > 0 \textbf{ then } (\textbf{if } x > 3 \textbf{ then } x - 5 \textbf{ else } x - 3 \textbf{ fi}) \textbf{ else } x \textbf{ fi} \\
\tau^4\{x > 1\}\{x > 3\} & \equiv & \textbf{if } x > 0 \textbf{ then } x - 5 \textbf{ else } x \textbf{ fi} \\
\tau^4\{x > 1\}\{x \leq 3\} & \equiv & \textbf{if } x > 0 \textbf{ then } x - 3 \textbf{ else } x \textbf{ fi} \\
\tau^4\{x \leq 1\} & \equiv & \textbf{if } x > 0 \textbf{ then } (\textbf{if } x > 1 \textbf{ then } x - 3 \textbf{ else } x - 1 \textbf{ fi}) \textbf{ else } x \textbf{ fi} \\
 & \equiv & \textbf{if } x > 0 \textbf{ then } x - 1 \textbf{ else } x \textbf{ fi}
\end{array}
$$

Pokračujeme s možnosťou $\tau[\tau_1^3]$:

$$\begin{array}{rcl}\tau^4 \equiv \tau[\tau_1^3] & \equiv & \textbf{if } x > 0 \\ & & \textbf{then if } [\textbf{if } x > 1 \textbf{ then } x - 4 \textbf{ else } x \textbf{ fi}] > 0 \\ & & \quad \textbf{then } [\textbf{if } x > 1 \textbf{ then } x - 4 \textbf{ else } x \textbf{ fi}] - 3 \\ & & \quad \textbf{else } (\textbf{if } x > 1 \textbf{ then } x - 4 \textbf{ else } x \textbf{ fi}) \\ & & \quad \textbf{fi} \\ & & \textbf{else } x \\ & & \textbf{fi}\end{array}$$

Aj toto konkrétne vyjadríme:

$$\begin{array}{lcl}\tau^4\{x > 1\} & \equiv & \textbf{if } x > 0 \textbf{ then } (\textbf{if } x > 4 \textbf{ then } x - 7 \textbf{ else } x - 4 \textbf{ fi}) \textbf{ else } x \textbf{ fi} \\ \tau^4\{x > 1\}\{x > 4\} & \equiv & \textbf{if } x > 0 \textbf{ then } x - 7 \textbf{ else } x \textbf{ fi} \\ \tau^4\{x > 1\}\{x \leq 4\} & \equiv & \textbf{if } x > 0 \textbf{ then } x - 4 \textbf{ else } x \textbf{ fi} \\ \tau^4\{x \leq 1\} & \equiv & \textbf{if } x > 0 \textbf{ then } (\textbf{if } x > 0 \textbf{ then } x - 3 \textbf{ else } x \textbf{ fi}) \textbf{ else } x \textbf{ fi} \\ & \equiv & \textbf{if } x > 0 \textbf{ then } x - 3 \textbf{ else } x \textbf{ fi}\end{array}$$

Takto postupne konštruujeme aproximácie. Evidentne vyzerajú dobre, v každom kroku nám pribudne +1 v niektorej z nich. Aký je teda pevný bod?

$$f_P \equiv \textbf{if } x > 0 \textbf{ then } 0 \textbf{ else } x \textbf{ fi}$$

Čiže

$$\begin{array}{rl}f_P : & \textbf{if } x > 0 \\ & \textbf{then if } [\textbf{if } x > 1 \textbf{ then } 0 \textbf{ else } x - 1 \textbf{ fi}] > 0 \\ & \quad \textbf{then } 0 \\ & \quad \textbf{else } (\textbf{if } x > 1 \textbf{ then } 0 \textbf{ else } x - 1 \textbf{ fi}) \\ & \quad \textbf{fi} \\ & \textbf{else } x \\ & \textbf{fi}\end{array}$$

sa nám rozloží na tri prípady:

1. $x > 1 \Rightarrow \textbf{if } x > 0 \textbf{ then } (\textbf{if } 0 > 0 \textbf{ then } 0 \textbf{ else } 0 \textbf{ fi}) \textbf{ else } x \textbf{ fi} =$
   $= \textbf{if } x > 0 \textbf{ then } 0 \textbf{ else } x \textbf{ fi}$

2. $x = 1 \;\Rightarrow\; \textbf{if } x > 0 \textbf{ then } (\textbf{if } x > 1 \textbf{ then } 0 \textbf{ else } x - 1 \textbf{ fi}) \textbf{ else } x \textbf{ fi} =$
   $= \;\; \textbf{if } x > 0 \textbf{ then } 0 \textbf{ else } x \textbf{ fi}$

3. $x > 0 \;\Rightarrow\; \textbf{if } x > 0 \textbf{ then } 0 \textbf{ else } x \textbf{ fi} =$
   $= \;\; \textbf{if } x > 0 \textbf{ then } 0 \textbf{ else } x \textbf{ fi}$

Ukázali sme, že f_P je pevný bod.

Príklad 39 Uvažujme rovnakú funkciu f ako v predchádzajúcom príklade. Nech $g(x)$ je jej pevný bod. Ukážte, že je silne ekvivalentný s $f_P(x)$.

Riešenie 39 Fixpointovou indukciou dokážeme, že pre $x \geq 0$:

$$f_P(x) \sqsubseteq g(x)$$

Označme

$$g(x) \equiv \textbf{if } x > 0 \textbf{ then } 0 \textbf{ else } x \textbf{ fi}$$

Podľa fixpointovej indukcie stačí dokázať, že $\tau[g] \sqsubseteq g$, čiže:

$$\left(\begin{array}{l}\textbf{if } x > 0 \\ \textbf{then if } [\textbf{if } x - 1 > 0 \textbf{ then } 0 \textbf{ else } x - 1 \textbf{ fi}] > 0 \\ \qquad \textbf{then } 0 \\ \qquad \textbf{else } (\textbf{if } x - 1 > 0 \textbf{ then } 0 \textbf{ else } x - 1 \textbf{ fi}) \\ \qquad \textbf{fi} \\ \textbf{else } x \\ \textbf{fi}\end{array}\right) \sqsubseteq \left(\begin{array}{l}\textbf{if } x > 0 \\ \textbf{then } 0 \\ \textbf{else } x \\ \textbf{fi}\end{array}\right)$$

Rozdelíme to na nasledujúce prípady:

1. $x = 0 : \tau[g](x) = 0 = g(x)$
2. $x > 0 : \tau[g](x) = 0 = g(x)$

Teda platí $f_P \sqsubseteq g$. A navyše pre $x < 0$ platí $g(x) = x$, potom musí $f(x) = x$ aby platilo $g \sqsubseteq f_P$. Celkovo platí

$$(g \sqsubseteq f_P) \wedge (f_P \sqsubseteq g) \implies f_P \equiv g$$

Teda g je najmenší pevný bod.

Záver

Cieľom našej práce bolo vytvoriť zbierku riešených úloh z matematickej teórie programovania. Táto zbierka bola primárne zameraná na najväčšiu cieľovú skupinu, a to študentov Informatiky na FMFI UK, konkrétne poslucháčov predmetu Základy teórie programovania. Samozrejme sme predpokladali aj širšie možnosti využitia, a to najmä preto, že na Slovensku doteraz podobná zbierka z danej oblasti chýbala.

Myslíme si, že tento cieľ sa nám vo veľkej miere podarilo splniť. Zbierka ponúka širokú škálu tématických oblastí, ako i úrovne náročnosti úloh. Veríme, že bude slúžiť, ako vhodný doplnkový materiál pri štúdiu tejto modernej a vzhľadom na svoje široké možnosti uplatnenia aj veľmi populárnej matematicko-informatickej disciplíny. Pri riešení problémov si čitateľ upevňuje nadobudnuté poznatky, precvičuje si základné operácie so schémami a najmä si zautomatizuje niektoré štandardné postupy, často využívané či už pri dokazovaní (ne)rozhodnuteľnosti vlastností programových schém ako aj dokazovaní čiastočnej správnosti

Vzhľadom na veľmi široký záber disciplíny akou je matematická teória programovania nebolo možné v zbierke obsiahnuť všetky jej skúmané odvetvia. Myslíme si však, že ich výber prezentovaný v tejto zbierke zahŕňa väčšinu z najpodstatnejších, najviac prebádaných a najvyužívanejších tématických okruhov, akými sú schémy, programy, interpretácie, (ne)rozhodnuteľnosť vlastností, správnosť programov, dokazovanie správnosti a úvod do sémantiky programov.

Samozrejme si uvedomujeme, že existuje ešte množstvo ďalších tém, ktoré sa do zbierky nezmestili a tiež ich nemôžeme označiť za nepodstatné. Práve tu vidíme najväčší priestor pre pokračovanie v našej práci. Zbierka by sa mohla rozrásť o ďalšie kapitoly so spomínanými, ale aj ďalšími odvetviami.

Literatúra

[1] Igor Prívara: *Základy teórie programovania*, Fakulta matematiky, fyziky a informatiky UK, Bratislava.

[2] Igor Prívara: *Formálne špecifikácie programov a Algebraické špecifikácie*, Inštitút informatiky a štatistiky, Bratislava.

[3] Zohar Manna: *Mathematical Theory Of Computation*, McGraw-Hill, New York 1974.

[4] Anthony J. Field, Peter G. Harrison: *Functional Programming*, Addison Wesley.