
FAKULTA MATEMATIKY, FYZIKY
A INFORMATIKY
UNIVERZITY KOMENSKÉHO
KATEDRA INFORMATIKY


Základy teórie programovania

Zbierka riešených príkladov

ONDREJ JOMBÍK

release 0.7 build 2003-04-27

Bratislava 2003

Vďaka zmene systému zo známkovacieho na kreditové, bolo možné zapísať si ZÁKLADY TEÓRIE PROGRAMOVANIA v troch po sebe idúcich školských rokoch. Treba mať na pamäti, že autori podieľajúci sa na tvorbe tohto dokumentu, tento predmet aj tri krát zapísaný mali. Z toho vyplýva, že určite nepatria k absolútnym odborníkom na danú problematiku. Tento dokument si vytvorili len pre svoju osobnú potrebu ako užitočnú pomôcku pri štúdiu.

Dokument je len doplnkovým materiálom. V žiadnom prípade nemá slúžiť ako náhrada za dobré skriptá alebo prednášku zo ZÁKLADOV TEÓRIE PROGRAMOVANIA. Autori nenesú žiadnu zodpovednosť za prípadné chyby, preklepy alebo nepresnosti, ale privítajú ich opravy či vylepšenia. Rozšírenie dokumentu o ďalšie príklady a poznámky je taktiež vítané.

Osobitné poďakovanie patrí ĽUBOMÍROVI HOSTOVI za vytvorenie skvelého pracovného a zostavovacieho rámca pre prácu so systémami LaTeX a pdfTeX.

Obsah

Kapitola 1

Programové schémy

1.1 Štandardné programové schémy

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

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

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 $I_1 = (D_1, i_1)$:

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

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 $I_2 = (D_2, i_2)$:

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

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 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 4 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.

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

Príklad 5 Zostrojte Herbrandové univerzum pre predchádzajúcu schému S.

Riešenie 5 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 6 Schéma S sa zastaví práve vtedy, keď sa zastaví výpočet pre každú Herbrandovú interpretáciu schémy S[1].

[1] Ide o zadanie domácej úlohy čislo 1 v roku 2003. Riešenie nie je overené, takže môže byť a pravdepodobne aj je nepresné alebo nesprávne.

Riešenie 6

$\underline{\Longrightarrow}$: 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

Príklad 7 Máme danú schému S.

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

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

Riešenie 7 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:

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

Príklad 8 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 8 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.

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

Príklad 9 Rozhodnite, či je problém dosiahnuteľnosti príkazu v štandardnej schéme rozhodnuteľný. Svoje tvrdenie dokážte [2].

Riešenie 9 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:

[2] Ide o zadanie domácej úlohy číslo 2 v roku 2003. Riešenie nie je overené, takže môže byť a pravdepodobne aj je nepresné alebo nesprávne.

- 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.

Príklad 10 Uvažujme triedu dosiahnuteľných schém $\mathcal{D}$. Je príslušnosť schémy k triede $\mathcal{D}$ rozhodnuteľný problém? Svoje tvrdenie zdôvodnite.

Riešenie 10 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 11 Uvažujme triedu štrukturovaných schém $\mathcal{W}$. Je problém divergencie pre štruktúrované schémy rozhodnuteľný? Svoje tvrdenie zdôvodnite.

Riešenie 11 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

Príklad 12 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 12 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 poďla 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 13 Máme danú Janovovu schému S.

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

Nájdite ku schéme S ekvivalentnú voľnú Janovovu schému S_v. Schému napíšte a stručne zdôvodnite, prečo je schéma S_v voľná a ekvivalentná so schémou S.

Riešenie 13 Riešením je schéma S_v.

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

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_v a valuácia v_v taká, že výpočet $(S_v,\ I_v,\ v_v)$ 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 14 Daná je štandardná schéma S.

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

Nájdite ku schéme S ekvivalentnú voľnú schému S_v[3].

Riešenie 14 Riešením je schéma S_v.

[3]Ide o zadanie domácej úlohy číslo 3 v roku 2003. Riešenie nie je overené, takže môže byť a pravdepodobne aj je nepresné alebo nesprávne.

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

Schému S_v sme našli pomocou vytvorenia vývojového diagramu pôvodnej schémy S a jeho následných modifikácií vedúcich k zvoľneniu pri zachovaní ekvivalencie. Tento postup je štandardný. Nedoporučuje sa písať programový kód výslednej schémy priamo[4].

Príklad 15 Máme danú štandardnú schému S.

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

Nájdite ku schéme S ekvivalentnú rekurzívnu schému R.

Riešenie 15 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).

[4] pretože to ide naozaj len veľmi ťažko

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

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 16 Daná je štandardná schéma S.

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

Nájdite rekurzívnu schému R, ktorá je ekvivalentná so schémou S. Upravte nájdenú schému tak, aby mala minimálny počet funkčných premenných [5].

Riešenie 16 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:

[5] Ide o zadanie domácej úlohy čislo 4 v roku 2003. Riešenie nie je overené, takže môže byť a pravdepodobne aj je nepresné alebo nesprávne.

- 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.

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

Príklad 17 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 17 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.

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

Príklad 18 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 18 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.

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

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 19 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 19

$\mathcal{S}, \mathcal{V}$ – porovnanie tried štruktúrovaných 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 štruktúrovaných 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 štruktúrovaných a dosiahnuteľných schém

$\mathcal{S} \not\subseteq \mathcal{D}$ Existuje štandartná 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 štandartnej schémy sa stane schéma dosiahnuteľnou.

$\mathcal{S} \not\trianglelefteq \mathcal{D}$ Odstraňovanie komponetov nesúvislosti však nemôže ísť spraviť efektívne. Ak by to išlo, vedeli by sme rozhodovať divergenciu štandartných schém. Každú schému z triedy $\mathcal{S}$ by sme previedli na ekvivateltnú 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 20 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 20

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

$\mathcal{V} \not\subseteq \mathcal{D}$ 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 preložiť do ekvivalentnej priechodnej schémy. Algoritmus je lineárny vzhľadom na počet hrán a kvadratický vzhľadom na počet príkazov schémy.

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

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

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.

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

$\mathcal{D} \not\trianglelefteq \mathcal{V}$ 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 21 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 21 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.

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

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.

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

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

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 22 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 22 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:

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

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

cesta 11:

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

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

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

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 23 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 23 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 taktiež sformulovať podmienku r. Jej znenie je rovnaké, ako znenie invariantu v ekvivalentnom štandartnom programe dokazovanom Floydovou metódou[1].

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

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[2].

Príklad 24 Daný je štruktúrovaný program P.

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

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 24 TODO

[1] Od tohto miesta sa mi toto riešenie veľmi nepozdáva. Privítam komentáre a poznámky, týkajúce sa hlavne hľadania podmienky r.

[2] A preto ich prenechávam na čitateľa, ja idem pozerať hokej. . . :-)

2.2 Rozširovanie Hoareovských kalkulov

Príklad 25 Navrhnite inferenčné pravidlo Hoareovho dokazovacieho systému pre riadiacu štruktúru reprezentujúcu tzv. jeden a pol cyklus.

do
  S_1;
  exit when b;
  S_2;
od

Dokážte zdravosť navrhnutého pravidla.

Riešenie 25 TODO

Príklad 26 Sformulujte inferenčné pravidlo Hoareovského kalkulu pre príkaz **repeat** definovaný 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 26 TODO

Kapitola 3

Sémantika programov

Literatúra

[1] Igor Prívara, *Základy teórie programovania*, Fakulta matematiky, fyziky a informatiky UK, Bratislava 19??