FAKULTA MATEMATIKY, FYZIKY
A INFORMATIKY
UNIVERZITY KOMENSKÉHO
KATEDRA INFORMATIKY

# Matematická teória programovania

## Zbierka riešených príkladov

ONDREJ JOMBÍK

release 0.8.4 build 2005-11-23

Bratislava 2003–2005

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 trikrát 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.

Domovská stránka dokumentu je:

`http://nepto.sk/skola/ztp/`

Aktuálnu verziu je možné si prezrieť na:

`http://platon.sk/projects/ztp-zbierka/`

Dokument je vyvíjaný a spravovaný v CVS archíve Platon Group:

`http://platon.sk/cvs/sk/cvs.php/doc/ztp-zbierka/`

Platon Group je slovenská skupina vývojárov otvoreného softvéru. Vyvíjajú, spravujú, dokumentujú a ponúkajú vysokokvalitné softvérové riešenia. Podporovali a sponzorovali tiež vývoj tejto zbierky príkladov a cvičení.

`www.platon.sk`

Document homepage is:

`http://nepto.org/school/ztp/`

Current version can be viewed online at:

`http://platon.sk/projects/ztp-zbierka/`

Document is maintained in the Platon Group CVS repository:

`http://platon.sk/cvs/cvs.php/doc/ztp-zbierka/`

Platon Group is Slovak open source software development group.
They are developing, maintaining, documenting and offering high quality software solutions. They also kindly supported and sponsored this colletion of examples and exercises.

`www.platon.sk`

# Obsah

# Kapitola 1

# Programové schémy

## 1.1 Štandardné programové schémy

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

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

**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:

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

**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 $I_1 = (D_1, i_1)$:

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

**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 $I_2 = (D_2, i_2)$:

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

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 $S$.

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

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.

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

**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.

$$\begin{array}{l} "f_1(f_1(x))", "f_1(f_1(a_1))", "f_1(f_1(a_2))" \\ "f_1(f_2(x))", "f_1(f_2(a_1))", \ldots \\ "f_2(f_1(x))", "f_2(f_1(a_1))", \ldots \\ "f_2(f_2(x))", \ldots \end{array}$$

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

**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.

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

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 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.

**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

**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 $S_1$.

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

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 $S_{1_v}$.

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

<u>Voľnosť:</u> 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.

<u>Ekvivalencia:</u> 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 $S_2$.

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

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

**Riešenie 15** Riešením je schéma $S_{2_v}$.

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

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 $\phi_i$ (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 $\phi_i$ dosiahneme zjednodušenie systému rekurzívnych definícií a výslednú schému $R$.

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

**Príklad 18** Daná je štandardná schéma $S$.

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

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.

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

Minimálny počet funčných premenných dosiahneme nasledovnými krokmi:

– Zrušením $\phi_2$ a dosadením $\phi_3$ na príslušné miesta vo $\phi_1$ a $\phi_4$.

– Zrušením $\phi_e$ a dosadením $y$ na príslušné miesto vo $\phi_1$.

– Zrušením $\phi_5$ a dosadením $\phi_1$ na príslušné miesto vo $\phi_4$.

– Zrušením $\phi_4$ a dosadením príkazu **if** na príslušné miesta vo $\phi_3$.

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

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

**Príklad 19** Máme danú štandardnú schému $S$.

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

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.

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

**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.

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

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.

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

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 $W_1$.

$W_1'$ : $[bool] := false;$
**if** $b_1$ **then**
  **if** $b_2$ **then** $[bool] := true;$
**while** $bool$ **do**
  $S;$
  $[bool] := false;$
  **if** $b_1$ **then**
    **if** $b_2$ **then** $[bool] := true;$
**od**

Viac elegancie v riešení je možné získať nahradením konštrukcií

$[bool] := false;$
**if** $b_1$ **then**
  **if** $b_2$ **then** $[bool] := true;$

za jednoduchšie

**if** $b_1$ **then** $[bool] := [b_2]$
  **else** $[bool] := [b_1]$

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 $W_3$.

$W_3'$ : $[bool] := true;$
**if** $b$ **then** $[bool] := false;$
**while** $bool$ **do**
  $S;$
  $[bool] := true;$
  **if** $b$ **then** $[bool] := false;$
**od**

**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 $\mathcal{D} \not\sqsubseteq \mathcal{V}$.

$\mathcal{S} \ntrianglelefteq \mathcal{V}$ 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.

<u>$\mathcal{S}$, $\mathcal{R}$ – porovnanie tried štandardných a rekurzívnych schém</u>

$\mathcal{S} \not\subseteq \mathcal{R}$ 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).

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

<u>$\mathcal{S}$, $\mathcal{D}$ – porovnanie tried štandardných a dosiahnuteľných schém</u>

$\mathcal{S} \not\subseteq \mathcal{D}$ 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.

$\mathcal{S} \ntrianglelefteq \mathcal{D}$ Odstraňovanie komponetov nesúvislosti však nemôže ísť spraviť 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

$\mathcal{V} \nsubseteq \mathcal{D}$ 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 dosiahnuteľnej schémy. Algoritmus je lineárny vzhľadom na počet hrán a kvadratický vzhľadom na počet príkazov schémy.

$\mathcal{D} \nsubseteq \mathcal{V}$ 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** | $[z] := [y_1]$ |

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

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

$\mathcal{V} \nsubseteq \mathcal{P}$ 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.

$\mathcal{P} \nsubseteq \mathcal{V}$ 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 $\mathcal{D} \not\sqsubseteq \mathcal{V}$.

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

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

$\mathcal{V} \nsubseteq \mathcal{J}$ 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.

$\mathcal{V} \ntrianglelefteq \mathcal{J}$ 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 $x$).

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

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 $z = \lceil \sqrt{x} \rceil$.

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

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** → riadok 1
- cesta 11: riadok 1 → riadok 1
- cesta 1E: riadok 1 → **end**

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

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

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

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

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

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

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

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

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

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

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

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 $P$.

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

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 $x_1 = z_1x_2 + z_2$, 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 $I_1$.

$$I_1 : \; x_1 = y_1x_2 + y_2 \;\wedge\; x_2 > 0 \;\wedge\; y_1 \geq 0 \;\wedge\; y_2 \geq 0$$

Pre každú z troch ciest odvodíme a dokážeme verifikačnú podmienku.

<u>cesta B1:</u>

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

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

<u>cesta 11:</u>

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

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

$$\begin{aligned} & x_1 = y_1x_2 + y_2 \;\wedge\; x_2 > 0 \;\wedge\; y_1 \geq 0 \;\wedge\; y_2 \geq 0 \;\wedge\; y_2 \geq x_2 \;\Rightarrow \\ & \Rightarrow\; x_1 = (y_1 + 1)x_2 + (y_2 - x_2) \;\wedge\; x_2 > 0 \;\wedge\; y_1 + 1 \geq 0 \;\wedge\; y_2 - x_2 \geq 0 \\ & \qquad (\text{zrejme } y_2 \geq 0 \wedge x_2 > 0 \wedge y_2 \geq x_2 \Rightarrow y_2 - x_2 \geq 0) \\ & x_1 = y_1x_2 + y_2 \;\wedge\; y_1 \geq 0 \;\wedge\; y_2 - x_2 \geq 0 \;\Rightarrow \\ & \qquad \Rightarrow\; x_1 = y_1x_2 + y_2 \;\wedge\; y_1 + 1 \geq 0 \;\wedge\; y_2 - x_2 \geq 0 \\ & \qquad (\text{zrejme platí } y_1 \geq 0 \Rightarrow y_1 + 1 \geq 0) \end{aligned}$$

<u>cesta 1E:</u>

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

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

$$\begin{aligned} x_1 = y_1x_2 + y_2 \ \wedge \ x_2 > 0 \ \wedge \ y_1 \geq 0 \ \wedge \ y_2 \geq 0 \ \wedge \ y_2 < x_2 \ \Rightarrow \\ \Rightarrow \ x_1 = y_1x_2 + y_2 \ \wedge \ 0 \leq y_2 < x_2 \\ x_1 = y_1x_2 + y_2 \ \wedge \ y_2 \geq 0 \ \wedge \ y_2 < x_2 \ \Rightarrow \\ \Rightarrow \ x_1 = y_1x_2 + y_2 \ \wedge \ y_2 \geq 0 \ \wedge y_2 < x_2 \end{aligned}$$

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 $P$.

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

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

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** $[y_1, y_2, y_3] := [1, 0, 0]$
**while** $y_1 \leq n$ **do**
$[y_3] := [a[y_1]]$;
**if** $y_3 < 0$ **then**
$[y_3] := [-y_3]$
**fi**;
$[y_1, y_2] := [y_1 + 1, y_2 + y_3]$
**od**
**end** $[z] := [y_2]$

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.

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

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.*

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

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

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

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

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

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

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

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.

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

**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.

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

**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

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

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

kde

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

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

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

Tvrdenie dokážte.

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

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

Teda pre programový subsegment $S_0$ platí

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

kde

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

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

2. Nech $\sqcup_0^\infty\{\phi_i\} \neq\perp$. 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 $\psi$. Vieme, že $k$-krát bude $true$, potom $false$. Vykonal sa teda rovnaký kód ako pri $\phi$. $\sqcup_0^\infty\{\psi_i\}$ bola $\sigma$. Takže $\sqcup_0^\infty$ ľavej strany priradenia je $\mathcal{M}[\![S_2]\!]\sigma$.

# Literatúra

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