ZÁKLADY

TEÓRIE PROGRAMOVANIA

Igor Prívara

Obsah

Programové schémy 5

1 Štandardné programové schémy **7**
- 1.1 Syntax štandardných schém . . . 7
- 1.2 Program – interpretovaná schéma . . . 9
- 1.3 Výpočet interpretovanej schémy . . . 10
- 1.4 Vlastnosti programových schém . . . 11
- 1.5 Herbrandove interpretácie . . . 12
- 1.6 Zladené interpretácie . . . 12

2 Nerozhodnuteľnosť vlastností schém **15**
- 2.1 Rozhodovacie problémy . . . 16
- 2.2 Nerozhodnuteľnosť problému divergencie . . . 17
- 2.3 Nerozhodnuteľnosť problémov zastavenia, ekvivalencie a izomorfizmu . . . 20
- 2.4 Vlastnosti podtried štandardných schém . . . 20
  - 2.4.1 Podtriedy štandardných schém . . . 21
  - 2.4.2 Voľné schémy . . . 21
  - 2.4.3 Janovove schémy . . . 25
  - 2.4.4 Na hranici medzi rozhodnuteľnosťou a nerozhodnuteľnosťou . . . 26

3 Porovnávanie tried programových schém **27**
- 3.1 Štruktúrované schémy . . . 28
- 3.2 Porovnanie štandardných a štruktúrovaných schém . . . 29
- 3.3 Rekurzívne schémy . . . 30
- 3.4 Porovnanie štandardných a rekurzívnych schém . . . 33
- 3.5 Čiastočne interpretované schémy . . . 35

Správnosť programov **37**

4 Indukčné metódy pri dokazovaní správnosti **39**
- 4.1 Výpočtová indukcia . . . 39
- 4.2 Štrukturálna (noetherovská) indukcia . . . 40

5 Floydova metóda **41**
- 5.1 Dôkaz správnosti Floydovou metódou . . . 41
- 5.2 Invariant . . . 42
- 5.3 Konštrukcia verifikačných podmienok . . . 42
- 5.4 Dôkaz zastavenia programu . . . 45

6 Hoareova metóda **47**
- 6.1 Logický systém . . . 47
- 6.2 Induktívna (invariantná) formula . . . 48
- 6.3 Inferenčný systém $\mathcal{H}$ . . . 50
- 6.4 Poznámky o úplnosti Hoareovských kalkulov . . . 54

7 Metóda intermitentov **55**
- 7.1 Intermitent . . . 55
- 7.2 Dôkaz správnosti metódou intermintentov . . . 55

8 Konštrukcia invariantov a správnych programov **58**
- 8.1 Návrh invariantov . . . 58
- 8.2 Systematický vývoj korektných programov . . . 60

Sémantika programov **65**

9 Formálna sémantika programov **67**
- 9.1 Porovnanie sémantických definícií . . . 67
- 9.2 Kompozičná sémantika . . . 68
- 9.3 Základné prístupy pri popise významu programov . . . 68
- 9.4 Ekvivalencia programov . . . 70

10 Algebraická štruktúra sémantických oborov **71**
- 10.1 plné čiastočné usporiadanie . . . 72
- 10.2 Konštrukcia cpo . . . 72

11 Sémantika imperatívnych programov **75**
- 11.1 Syntax jazyka . . . 76
- 11.2 Sémantické obory . . . 77
- 11.3 Sémantika výrazov . . . 78
- 11.4 Operačná vstupno–výstupná sémantika . . . 79
- 11.5 Denotačná sémantika . . . 80
- 11.6 Porovnanie operačnej a denotačnej sémantiky . . . 82

12 Sémantika rekurzívnych funkcionálnych programov **84**
- 12.1 Syntaktické obory . . . 85
- 12.2 Obory interpretácie a sémantické obory . . . 86
- 12.3 Pevné body funkcionálov . . . 89

12.4 Systémy rekurzívnych definícií . . . 91
12.5 Sémantika rekurzívnych programov . . . 91
12.6 Výpočet rekurzívnych programov . . . 92
12.7 Korektnosť výpočtových pravidiel . . . 95
12.8 Kritéria korektnosti pravidiel . . . 96

13 Nedeterministické programy a schémy **100**
13.1 Syntax nedeterministických schém . . . 100
13.2 Interpretácia nedeterministických schém . . . 102
13.3 Význam nedeterministických schém . . . 105
13.4 Správnosť programov v relačnom kalkule . . . 107

Časť I

Programové schémy

Predmetom štúdia teórie programových schém sú riadiace štruktúry programov.

1. Abstrakcia programu: Podstatou abstrakcie je "odfiltrovanie" nepodstatných informácií, ktoré nie sú významné pri analýze problémov súvisiacich s predmetom štúdia, t.j. v tomto prípade pri štúdiu vlastností riadiacich štruktúr programov.

Teória programových schém sa zaoberá štúdiom vlastností riadiacich štruktúr, nezávisle od funkcií a predikátov, použitých v programe. Riadiaca štruktúra schémy zachováva štruktúru riadenia programu. Abstrahuje sa však od funkcií a predikátov použitých v programe tým, že ich nahradíme "formálnymi" symbolmi funkcií a predikátov. Dôsledkom oddelenia špecifických vlastností interpretácie od štruktúry riadenia je možnosť lepšieho pochopenia základných rysov riadiacich štruktúr.

Programová schéma reprezentuje celú triedu programov s rovnakou štruktúrou riadenia. Rôznymi interpretáciami symbolov funkcií a predikátov môžme dostať rôzne programy. Ak teda nejaká vlastnosť, týkajúca sa riadiacej štruktúry, platí pre schému S, platí pre každý program P, ktorý je "inštanciou" schémy S.

2. Triedy programových schém: V závislosti na tom, aké typy riadiacich štruktúr sa študujú uvažujeme rôzne triedy programových schém, napr. štandardné schémy, štrukturované schémy, rekurzívne schémy, atď. Každá z uvedených tried schém zachováva podstatné vlastnosti a zvláštnosti niektorej z "ostro sledovaných" tried programov (fortranovské, pascalovské, resp. funkcionálne programy).

Jednou zo základných otázok teórie programových schém je štúdium výrazovej sily riadiacich štruktúr, t.j. porovnávanie jednotlivých tried programových schém. Dosiahnuté výsledky umožujú hlbšie vniknúť do tajov odstraovania rekurzie pomocou iterácie, do teoretických základov štrukturovaného programovania, atď.

3. Rôzne stupne abstrakcie: Pri definícii programových schém môžme vychádzať z rôznych úrovní abstrakcie. Podstatou je určiť elementárne objekty, t.j. objekty, ktorých vnútorná štruktúra sa ďalej nebude analyzovať. Takýmito elementárnymi objektami môžu byť symboly premenných, funkcií a predikátov alebo symboly elementárnych príkazov. Ak je takýmto elementárnym príkazom priraďovací príkaz, stáva sa vlastne "čiernou skrinkou". Pri takomto stupni abstrakcie nie je potom možné analyzovať napr. vnútorné vlastnosti stavu výpočtu (stav pracovných premenných).

Kapitola 1

Štandardné programové schémy

1.1 Syntax štandardných schém

1. Symboly: Symbolmi sú individuálne premenné, funkčné symboly, predikátové symboly a špeciálne symboly.

Individuálne premenné – $X = \{x, y, z, \cdots\}$

- vstupné premenné – X_x – $\overline{x} = \{x_1, \cdots, x_k\}$
- výstupné premenné – X_z – $\overline{z} = \{z_1, \cdots, z_m\}$
- pracovné premenné – X_y – $\overline{y} = \{y_1, \cdots, y_n\}$

Funkčné symboly – $F = \bigcup_{i=0}^{\infty} F^i$

- pre $f \in F^n$ platí $arity(f) = n$
- $F^n = \{f, g, h, \cdots\}$ – n–arne symboly
- $F^0 = \{a, b, \cdots\}$ – konštanty

Predikátové symboly – $B = \bigcup_{i=0}^{\infty} B^i$

- pre $p \in B^n$ platí $arity(p) = n$
- $B^0 = \{0, 1\}$ – logické konštanty: true, false

Špeciálne symboly – $[,], (,), :=, :$

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

Termy – $T = \{t, t_i, \ldots\}$

1. $F^0 \subseteq T$, $X \subseteq T$,
2. ak $f \in F^n$, $t_1, \cdots, t_n \in T$ potom $f(t_1, \cdots, t_n) \in T$.

Predikáty – $P = \{q\}$

1. $B^0 \subseteq P$,
2. ak $p \in B^n$, $t_1, \cdots, t_n \in T$ potom $p(t_1, \cdots, t_n) \in P$.

Príkazy – $C = \{st, st_i, \cdots\}$

1. počiatočný – **begin** $[\overline{y}] := [t_1(\overline{x}), \cdots, t_n(\overline{x})]$,
2. koncový – **end** $[\overline{z}] := [t_1(\overline{x}, \overline{y}), \cdots, t_n(\overline{x}, \overline{y})]$,
3. priraďovací – $[\overline{y}] := [t_1(\overline{x}, \overline{y}), \cdots, t_n(\overline{x}, \overline{y})]$,
4. príkaz skoku – **goto** i,
5. podmienkový – **if** $p(\overline{x}, \overline{y})$ **then** st , kde st je priradenie alebo príkaz skoku,
6. príkaz s návestím – i : st (st – priradenie, podmienka, skok).

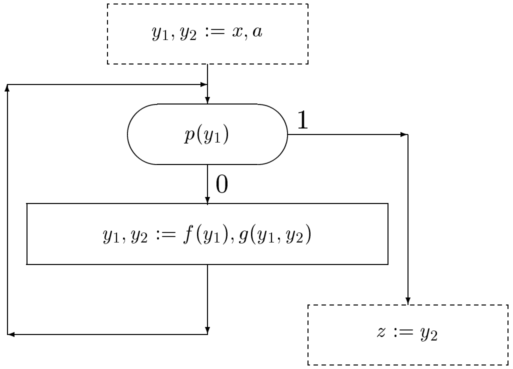
Definícia syntaxe podmienkového príkazu určuje, že podmienku reprezentuje predikátový symbol. Takže podmienkou nemôže byť formula vytvorená pomocou logických spojok resp. operácií (*not, and, or*), pretože tým by sme do schém zaviedli niektoré fixne interpretované symboly.

3. Štandardná programová schéma: Štandardnú programovú schému tvorí konečná postupnosť príkazov $S = \{st_0, st_1, \cdots, st_n, st_{n+1}\}$, kde

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

Príklad: grafická reprezentácia schémy:

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



Programová schéma nemusí obsahovať koncový príkaz, keď (v grafárskej terminológii) neexistuje žiadna “hrana”, ktorá do neho vedie.

1.2 Program – interpretovaná schéma

1. Interpretácia štandardnej programovej schémy: Interpretáciou programovej schémy S nazývame dvojicu $\mathcal{I} = (D, i)$, kde

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

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

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

2. Program: Programom nazývame dvojicu $(S, \mathcal{I})$, kde S je programová schéma a $\mathcal{I}$ interpretácia.

Príklad: Tromi rôznymi interpretáciami schémy S odvodíme principiálne odlišné programy, dokonca s rôznymi obormi interpretácie. Jediným spoločným prvkom týchto programov je tá istá štruktúra riadenia.

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

$\mathcal{I}_1$	N
a	1
p	$y_1 = 0$
f	$y_1 - 1$
g	$y_1 * y_2$

$\mathcal{I}_2$	N
a	0
p	$y_1 = 0$
f	$y_1 - 1$
g	$y_1 + y_2$

$\mathcal{I}_3$	Σ^*
a	nil
p	$isnil(y_1)$
f	$tail(y_1)$
g	$head(y_1).y_2$

- $(S, \mathcal{I}_1)$ počíta faktoriál x
- $(S, \mathcal{I}_2)$ počíta sumu $1 + 2 + \cdots + x$
- $(S, \mathcal{I}_3)$ obracia reťazec x

1.3 Výpočet interpretovanej schémy

V tejto časti zavedieme základné pojmy umožujúce *neformálne* popísať (operačný) význam ľubovolného programu (interpretovanej schémy) $P = (S, \mathcal{I})$.

1. Výpočet programu: Pri definícii výpočtu programu P so vstupom v – označenie $(S, \mathcal{I}, v)$ – predpokladáme, že

- v je *ohodnotenie vstupných premenných* – $v : X_x^k \mapsto D^k$,
- prirodzenú *operačnú sémantiku* so simultánnym priradením,
- *ohodnotenie výstupných premenných* ako dôsledok ukončenia výpočtu – $X_z^m \mapsto D^m$.

Ohodnotenie vstupných premených zodpovedá priradeniu hodnôt do vstupných premenných (napr. operáciou read). Ohodnotenie výstupných premenných zodpovedá priradeniu hodnôt do výstupných premenných (simulácia príkazu write).

Predpokladáme, že výpočet programu (interpretovanej schémy) sa definuje v súlade so zaužívaným štandardom pre túto triedu programov. Špecifickou črtou je len simultánne priradenie, ktoré treba interpetovať takto: v danom stave pracovných premenných sa (paralelne) vyhodnotia všetky výrazy na pravej strane priradenie a nový stav výpočtu vznikne priradením výsledkov vyhodnotenia do premenných na ľavej strane.

2. Výsledok výpočtu: Výsledkom výpočtu je hodnota $val(S, \mathcal{I}, v)$

- definovaná ohodnotením $\overline{d}$ výstupných premených $\overline{z}$, ak sa výpočet skončí,
- nedefinovaná, ak sa výpočet neskončí.

Význam daného programu definuje buď vstupno–výstupná charakterizácia programu (funkcia) alebo história jeho všetkých možných výpočtov.

3. História výpočtu: Históriu ľubovolného výpočtu je možné charakterizovať rôznymi spôsobmi, buď postupnosťou stavov alebo postupnosťou konfigurácií. Uvedené dva prístupy sa zreteľne odlišujú stupom abstrakcie.

- *Stav výpočtu* $[y_1, \cdots, y_n]$ – hodnoty pracovných premenných,
- *Konfigurácia* $[y_1, \cdots, y_n]_k$ – stav s udaním návestia k práve vykonaného príkazu,
- *Výpočtová postupnosť* – postupnosť stavov (konfigurácií) po každom kroku výpočtu.

1.4 Vlastnosti programových schém

1. Divergencia a zastavenie: Hovoríme, že program $P = (S, \mathcal{I})$ sa *zastaví*, ak pre každé ohodnotenie v vstupných premenných $\overline{x}$ je hodnota $val(S, \mathcal{I}, v)$ definovaná.

Definícia 1 *Schéma S sa zastaví, ak pre každú interpretáciu $\mathcal{I}$ sa zastaví program* $(S, \mathcal{I})$.

Hovoríme, že program $P = (S, \mathcal{I})$ *diverguje*, ak pre žiadne ohodnotenie v vstupných premenných $\overline{x}$ nie je hodnota $val(S, \mathcal{I}, v)$ definovaná.

Definícia 2 *Schéma S diverguje, ak pre každú interpretáciu $\mathcal{I}$ diverguje program* $(S, \mathcal{I})$.

Divergencia je duálnym, nie však komplementárnym, problémom k problému zastavenia! Nie je ťažké skonštruovať schému, ktorá sa pri jednej interpretácii zastaví a pri inej diverguje.

2. Ekvivalencia a izomorfizmus: Hovoríme, že schémy S_1, S_2 sú *kompatibilné*, ak majú rovnaké vektory vstupných a výstupných premenných. Programy $(S_1, \mathcal{I}_1), (S_2, \mathcal{I}_2)$ sú kompatibilné, ak sú kompatibilné schémy S_1, S_2 a obory interpretácií $\mathcal{I}_1, \mathcal{I}_2$ sú rovnaké.

Dva kompatibilné programy $(S_1, \mathcal{I}_1), (S_2, \mathcal{I}_2)$ sú *ekvivalentné* (označenie $\equiv$), ak pre rovnaké ohodnotenia vstupných premenných v sú hodnoty $val(S_1, \mathcal{I}_1, v), val(S_2, \mathcal{I}_2, v)$ buď obe nedefinované alebo obe rovnaké.

Definícia 3 *Dve kompatibilné schémy S_1, S_2 sú ekvivalentné, ak pre všetky intepretácie $\mathcal{I}$ sú programy $(S_1, \mathcal{I}), (S_2, \mathcal{I})$ ekvivalentné.*

Uvedený pojem ekvivalencie programov zodpovedá tzv. silnej ekvivalencii.

Definícia 4 *Dve kompatibilné schémy S_1, S_2 sú izomorfné (označenie $\cong$), ak pre každú interpretáciu $\mathcal{I}$ a pre každé ohodnotenie v sú postupnosti stavov výpočtov $(S_1, \mathcal{I}, v)$ a $(S_2, \mathcal{I}, v)$ rovnaké.*

Ekvivalencia schém je definovaná na základe vstupno–výstupnej charakterizácie, t.j. ekvivalentné schémy počítajú pri každej interpretácii rovnaké funkcie. Nezaujíma nás história výpočtu. Na druhej strane, izomorfizmus schém vyžaduje rovnaké histórie výpočtov pre všetky interpretácie. Keďže históriu výpočtu môžeme definovať rôznymi spôsobmi (napr. postupnosťami stavov, konfigurácií), definícia pojmu izomorfizmu schém zavisí od zvolenej histórie výpočtu. V danom prípade sme izomorfizmus definovali na základe histórie stavov výpočtov.

1.5 Herbrandove interpretácie

1. Herbrandovo univerzum: Herbrandovo univerzum $\mathcal{H}$ pozostáva z reťazcov symbolov, zostrojených zo vstupných premenných a funkčných symbolov:

- ak $x_i \in X_x$ potom "x_i" $\in \mathcal{H}$,
- ak $a \in F^0$ potom "a" $\in \mathcal{H}$,
- ak $f \in F^n$; "t_1", $\cdots$, "t_n" $\in \mathcal{H}$ potom "$f(t_1, \cdots, t_n)$" $\in \mathcal{H}$.

Príklad: Herbrandovo univerzum pre "abecedu" $x \in X_x, a \in F^0, h \in F^1, g \in F^2$
$\mathcal{H} = \{$"a", "x", "$h(a)$", "$h(x)$", "$g(x,x)$", "$g(a,a)$", "$g(a,x)$", "$g(x,a)$", "$h(h(a))$", "$h(h(x))$", "$h(g(a,a))$", "$h(g(a,x))$", "$h(g(x,a))$", "$h(g(x,x))$", "$g(h(a),a)$", "$g(a,h(a))$", $\ldots\}$

2. Herbrandova voľná interpretácia: Herbrandovou interpretáciou štandardnej programovej schémy S nazývame dvojicu $\mathcal{I}_H = (\mathcal{H}, i_H)$, kde:

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

1.6 Zladené interpretácie

1. Zladenie výpočtov a symbolických výpočtov:

Definícia 5 *Interpretácia $\mathcal{I}$ s ohodnotením v je zladená s voľnou interpretáciou $\mathcal{I}_H$ s ohodnotením predikátov práve vtedy, keď pre každé $p \in B^n$*

$$i_H(p)(\text{“}t_1\text{”}, \cdots, \text{“}t_n\text{”}) \iff i(p)(i^v(t_1), \cdots, i^v(t_n)).$$

Zatiaľ čo výpočtová cesta programom $(S, \mathcal{I})$ je definovaná výberom vstupných hodnôt (ohodnotením vstupných premenných), pri symbolickom výpočte $(S, \mathcal{I}_H)$ je výpočtová cesta charakterizovaná interpretáciou predikátových symbolov. Inými slovami, pri výpočtoch programu $(S, \mathcal{I})$ je pevná interpretácia žujme schému S a interpretáciu $\mathcal{I}_1$, definované v časti 1.2. S interpretáciou $\mathcal{I}_1$ a ohodnotením vstupu $v = [x \leftarrow 2]$ je zladená interpretácia, pri ktorej

$$p(\text{“}x\text{”}) = false \qquad p(\text{“}f(x)\text{”}) = false \qquad p(\text{“}f(f(x))\text{”}) = true.$$

V tomto prípade dostaneme nasledujúci výpočet (definovaný postupnosťou stavov) a "zladený" symbolický výpočet (charakterizovaný postupnosťou symbolických stavov).

$[2, 1]$	["x", "a"]
$[1, 2]$	["$f(x)$", "$g(x, a)$"]
$[0, 2]$	["$f(f(x))$", "$g(f(x), g(x, a))$"]
$[2]$	["$g(f(x), g(x, a))$"]

2. Vlastnosti zladených intepretácií: Prvá z vlastností zaručuje existenciu zladenej interpretácie.

Lema 6 *Ku každej interpretácii $\mathcal{I}$ symbolov danej schémy S a každému ohodnoteniu vstupných premenných v existuje voľná interpretácia $\mathcal{I}_H$, ktorá je s ou zladená (a naopak).*

Dôkaz: Nech $\forall \mathcal{I}$, $\forall p \in B^n$: $\quad T = T^0_{p,\mathcal{I}} + T^1_{p,\mathcal{I}}$, kde

$$T^k_{p,\mathcal{I}} = \{(t_1, \cdots, t_n) : i(p)(i^v(t_1), \cdots, i^v(t_n)) = k\}.$$

Interpretáciu $\mathcal{I}_H$, zladenú s $\mathcal{I}$, definuje predpis:

$$i_H(p)(\text{“}t_1\text{”}, \cdots, \text{“}t_n\text{”}) = k,$$

ak $(t_1, \cdots, t_n) \in T^k_{p,\mathcal{I}}$.

V prípade zladených intepretácií výpočet a "zladený' symbolický výpočet prechádzajú "tou istou cestou" v diagrame toku riadenia danej schémy. Formálne je táto vlastnosť formulovaná v leme 7 "o výpočtoch pri zladených interpretáciách".

Lema 7 *Nech S je schéma, $\mathcal{I}$ jej interpretácia, v ohodnotenie vstupov a $\mathcal{I}_H$ voľná interpretácia zladená s $\mathcal{I}$ a v. Potom pre j–te konfigurácie $[d_1, \cdots, d_n]^j_s$ a ["t_1", $\cdots$, "t_n"]$^j_{\overline{s}}$ výpočtov $(S, \mathcal{I}, v)$, $(S, \mathcal{I}_H,$ "$\overline{x}$"$)$ platí $s = \overline{s}$ a $i^v(t_i) = d_i$ pre všetky $i \in <1, n>$.*

Dôkaz: indukciou vzhľadom na výpočet. V každom indukčnom kroku treba brať do úvahy všetky možné príkazy.

Lema o výpočtoch pri zladených interpretáciách je kľúčovým výsledkom pre dôkaz nasledujúcich tvrdení.

Veta 8

1. *Schéma S sa zastaví práve vtedy, keď sa zastaví výpočet pre každú Herbrandovu interpretáciu schémy S.*

2. *Dve kompatibilné schémy S_1, S_2 sú ekvivalentné vtedy a len vtedy, keď pre každú Herbrandovu interpretáciu $\mathcal{I}_H$ sa programy $(S_1, \mathcal{I}_H)$, $(S_2, \mathcal{I}_H)$ buď oba zacyklia alebo sa oba zastavia, pričom platí*
$$val(S_1, \mathcal{I}_H, \text{“}\overline{x}\text{”}) = val(S_2, \mathcal{I}_H, \text{“}\overline{x}\text{”}).$$

3. *Dve kompatibilné schémy S_1, S_2 sú izomorfné práve vtedy, keď pre každú Herbrandovu interpretáciu je výpočtová postupnosť stavov S_1 identická s výpočtovou postupnosťou stavov S_2.*

Dôkaz: zrejme, ak platí niektorá vlastnosť pre všetky interpretácie, musí platiť aj pre Herbrandove. Opačná implikácia:

1. Zastavenie: Dôsledok uvedených liem.

2. Ekvivalencia: Nech $S_1 \equiv S_2$ na triede voľných interpretácií, t.j.
$$\forall \mathcal{I}_H : val(S_1, \mathcal{I}_H, \text{“}\overline{x}\text{”}) = val(S_2, \mathcal{I}_H, \text{“}\overline{x}\text{”}).$$
Potom pre všetky interpretácie $\mathcal{I}$ a ohodnotenia v existuje zladená interpretácia $\mathcal{I}_H$ taká, že $(S_1, \mathcal{I}), (S_1, \mathcal{I}_H)$ alebo oba cyklia alebo
$$val(S_1, \mathcal{I}, v) = i^v(val(S_1, \mathcal{I}_H, \overline{x})).$$
Analogická valstnosť platí pre S_2 (dôsledok lemy 7). Takže programy $(S_1, \mathcal{I}), (S_2, \mathcal{I})$ buď oba cyklia alebo sa oba zastavia s rovnakým výsledkom.

3. Izomorfizmus: Analógia s ekvivalenciou.

Kapitola 2

Nerozhodnuteľnosť vlastností schém

Prvým krokom pri štúdiu vlastností programových schém je odpoveď na otázku, či vieme rozhodnúť, či daná vlastnosť platí alebo nie.

Ako ukazujú nasledujúce príklady, v jednotlivých konkrétnych prípadoch sa dá (viac alebo menej) ľahko ukázať, že daná schéma (dané schémy) spĺňa analyzovanú vlastnosť.

Príklad: Najprv ukážeme, že schéma S_1 sa zastaví. Zrejme, vzhľadom na príkaz 1, nemôže prísť do hry pre žiadnu interpretáciu cyklus 2: … **goto** 2. V prípade cyklu 7: … **goto** 7 sú dve možnosti. Ak pri skoku z príkazu 1 platí $p(a)$ výpočet sa okamžite zastaví. Ak $p(a)$ neplatí a v príkaze 4 platí $p(fa)$ (ináč končíme), cyklus sa jedenkrát “otočí” ale ľahko overíme, že potom sa výpočet zastaví. Takže, pre ľubovolnú Herbrandovu interpretáciu $\mathcal{I}_H$ sa výpočet $(S, \mathcal{I}_H)$ zastaví.

Podobne jednoducho ukážeme, že schémy S_1, S_2 sú ekvivalentné. Schému S_2 dostaneme z S_1 vynechaním “zbytočného” príkazu 2.

S_1: **begin** $[y] := [a]$
- 1: **if** $p(y)$ **then goto** 7
- 2: **if** $p(y)$ **then goto** 2
- 3: $[y] := [f(y)]$
- 4: **if** $p(y)$ **then goto** 6
- 5: **goto end**
- 6: $[y] := [a]$
- 7: **if** $p(y)$ **then goto end**
- 8: $[y] := [f(y)]$
- 9: **goto** 7

end $[z] := [y]$

S_2: **begin** $[y] := [a]$
- 1: **if** $p(y)$ **then goto** 6
- 2: $[y] := [f(y)]$
- 3: **if** $p(y)$ **then goto** 5
- 4: **goto end**
- 5: $[y] := [a]$
- 6: **if** $p(y)$ **then goto end**
- 7: $[y] := [f(y)]$
- 8: **goto** 6

end $[z] := [y]$

Príklad: Podrobnou analýzou kompatibilných schém S_3, S_4, S_5 ľahko odhalíme, že schémy S_3, S_4 sú ekvivalentné a izomorfné (vzhľadom na históriu stavov). Schému S_4 dostaneme z S_3 rozvinutím cyklu. Nie sú však izomorfné vzhľadom na históriu konfigurácií. Na druhej strane sú schémy S_3, S_5 síce ekvivalentné ale nie sú izomorfné (ani vzhľadom na históriu stavov).

S_3: **begin** $[y] := [x]$

1: **if** $p(y)$ **then goto end**
2: $[y] := [f(y)]$
3: **goto** 1

end $[z] := [y]$

S_4: **begin** $[y] := [x]$

1: **if** $p(y)$ **then goto end**
2: $[y] := [f(y)]$
3: **if** $p(y)$ **then goto end**
4: $[y] := [f(y)]$
5: **goto** 3

end $[z] := [y]$

S_5: **begin** $[y] := [x]$

1: **if** $p(y)$ **then goto end**
2: **if** $p(f(y))$ **then goto** 5
3: $[y] := [f^2(y)]$
4: **goto** 1
5: $[y] := [f(y)]$

end $[z] := [y]$

2.1 Rozhodovacie problémy

Rozhodovací problém pre programové schémy odpovedá na zásadnú otázku, či pre danú triedu schém $\mathcal{P}$ a danú vlastnosť schém ϕ existuje všeobecný algoritmus, ktorý pre ľubovolnú schému $S \in \mathcal{P}$ (resp. ľubovolnú dvojicu schém $S_1, S_2 \in \mathcal{P}$) určí, či platí vlastnosť $\phi(S)$ (resp. $\phi(S_1, S_2)$) alebo nie.

Problém je *rozhodnuteľný* ak existuje všeobecný algoritmus, ktorý pre ľubovolnú schému (dvojicu schém) z danej triedy schém odpovie, či schéma (dvojica schém) spĺňa danú vlastnosť alebo nie. Problém je *nerozhodnuteľný*, ak takýto všeobecný algoritmus pre danú vlastnosť neexistuje.

Problém je *čiastočne rozhodnuteľný* ak existuje procedúra, ktorá pre ľubovolnú schému (dvojicu schém) z danej triedy schém odpovie kladne, keď schéma (dvojica schém) spĺňa danú vlastnosť a odpovie záporne alebo sa neskončí, keď schéma (dvojica schém) danú vlastnosť nespĺňa.

Pre triedu štandardných programových schém $\mathcal{S}$ sa študujú štyri rozhodovacie problémy: *problém zastavenia, problém divergencie, problém ekvivalencie, problém izomorfizmu.* Ukážeme, že v triede štandarných programových schém sú všetky štyri základné problémy nerozhodnuteľné. Problémy divergencie, ekvivalencie a izomorfizmu nie sú ani čiastočne rozhodnuteľné. Problém zastavenia je čiastočne rozhodnuteľný.

Dôkaz nerozhodnuteľnosti sa opiera o redukciu dvoch nerozhodnuteľných problémov špeciálnej triedy dvojpáskových automatov na rôzne varianty problému divergencie. Nerozhodnuteľnosť platí už pre relatívne úzku podtriedu štandardných schém. Nerozhodnuteľnosť ostatných problémov je dôsledkom nerozhodnuteľnosti divergencie.

2.2 Nerozhodnuteľnosť problému divergencie

Nerozhodnuteľnosť problému divergencie dokážeme pre triedu schém $\mathcal{S}_1 \subset \mathcal{S}$ redukciou nerozhodnuteľných problémov triedy dvojpáskových automatov $\mathcal{A}$ na "divergenčné" problémy schém z $\mathcal{S}_1$, t.j. ukážeme, že každý automat $A \in \mathcal{A}$ sa dá simulovať nejakou schémou $S \in \mathcal{S}_1$.

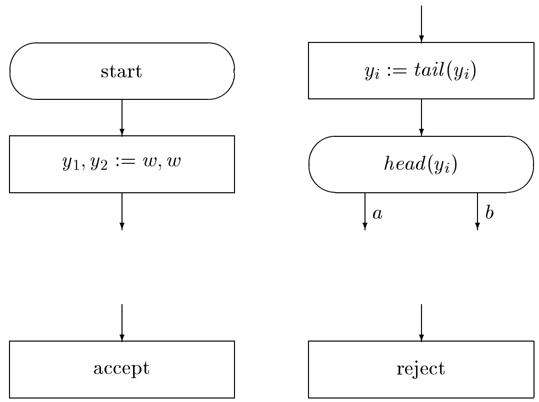
Veta 9 *Pre schémy $S \in \mathcal{S}_1$ nie sú nasledujúce problémy ani čiastočne rozhodnuteľné:*

- *S diverguje pre všetky interpretácie $\mathcal{I}$,*
- *existuje interpretácia $\mathcal{I}$ taká, že program $(S, \mathcal{I})$ diverguje.*

Pri dôkaz tejto vety postupne zadefinujeme triedu dvojpáskových automatov $\mathcal{A}$, triedu schém $\mathcal{S}_1$ a o popíšeme techniku redukcie automatov z $\mathcal{A}$ na schémy z $\mathcal{S}_1$.

1. Dvojpáskový konečný automat: Uvažujme triedu dvojpáskových konečných automatov $\mathcal{A}$, definovanú:

- vstupmi: nekonečné reťazce znakov nad dvojprvkovou abecedou $\Sigma^\infty = \{a, b\}^\infty$,
- operáciami: $tail(xw) = w$, $head(xw) = x$,
- stavmi: počiatočný, prechodový, akceptujúci, odmietací.



Automat A rozdeľuje množinu vstupných slov Σ^∞ do troch tried:

- $Accept(A)$ – množina akceptovaných slov,

- $Reject(A)$ – množina odmietnutých slov,
- $Loop(A)$ – množina slov, pre ktoré automat cyklí (nekonečné vstupné slovo).

Zrejme platí $\Sigma^{\infty} = Accept(A) \cup Reject(A) \cup Loop(A)$.

V nasledujúcej leme sú sformulované niektoré vlastnosti triedy dvojpáskových konečných automatov.

Lema 10

- *Problém "$Accept(A) = \emptyset$" nie je ani čiastočne rozhodnuteľný.*
- *Problém "$Loop(A) \neq \emptyset$" nie je ani čiastočne rozhodnuteľný.*

2. Podtrieda schém $\mathcal{S}_1$: Základnou myšlienkou dôkazu nerozhodnuteľnosti definovaných vlastností štandardných programových schém je redukcia triedy dvojpáskových automatov $\mathcal{A}$ na podtriedu štandardných schém $\mathcal{S}_1$.

Podtrieda $\mathcal{S}_1$ je definovaná nasledujúcimi syntaktickými ohraničeniami:

Symboly:

- $F^0 = \{a\}, F^1 = \{f\}, F^i = \emptyset$ pre $i \geq 2$
- $B^1 = \{p\}, B^i = \emptyset$ pre $i \geq 2$
- $X_x = \emptyset, X_y = \{y_1, y_2\}, X_z = \{z\}$

Príkazy:

- **begin** $[y_1, y_2] := [a, a]$
- i: $[y_j] := [f(y_j)]$ pre $j = 1, 2$
  i+1: **if** $p(y_j)$ **then** st
- i: **goto** i
- **end** $[z] := [a]$

Herbrandove interpretácie:

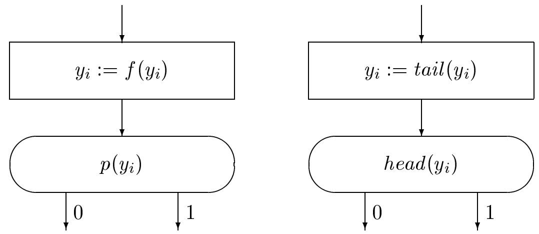
- Herbrandovo univerzum – {"a", "$f^i(a)$"}.
- Slovo $w \in \{0,1\}^*$ dĺžky n (resp. nekonečné slovo) zodpovedá voľnej interpretácii $\mathcal{I}_H$, ak pre všetky k $(1 \leq k \leq n)$ platí $i_H(p)(\text{"}f^k(a)\text{"}) = w_k$.

Uvedenou konštrukciou chceme dosiahnúť vlastnosť: ak na pásky automatu A dáme ľubovoľné slovo w, potom symbolický výpočet $(S, \mathcal{I}_H)$, kde $\mathcal{I}_H$ je interpretácia zladená s w sa zastaví práve vtedy, keď A akceptuje w.

3. Redukcia problému divergencie: Pri redukcii triedy automatov $\mathcal{A}$ na schémy z triedy $\mathcal{S}_1$ skonštruujeme ku každému dvojpáskovému automatu A dve schémy S', S'' tým, že preložíme:

- počiatočný stav $\rightsquigarrow$ počiatočný príkaz,
- prechodový stav $\rightsquigarrow$ dvojpríkaz,
- akceptujúci stav $\rightsquigarrow$ koncový príkaz,
- odmietací stav
  - schema S' $\rightsquigarrow$ večny cyklus,
  - schéma S'' $\rightsquigarrow$ koncový príkaz.

Pri redukcii zodpovedá každému automatu A so vstupným slovom w schéma $S' \in \mathcal{S}_1$ (resp. $S'' \in \mathcal{S}_1$) s interpretáciou $i_H(p)$ zladenou s w. Jednoduchou indukciou vzhľadom na počet modifikácií y_i dokážeme zladenie výpočtov automatu a schém:



inicializácia pracovnej premennej y_i:

$y_i := a$ $\qquad$ $y_i := w_0 w_1 w_2 \cdots$

stav po prvej modifikácii pracovnej premennej y_i:

$y_i := f(a)$ $\qquad$ $y_i := w_1 w_2 \cdots$
$i_H(p)(\text{"}f(a)\text{"}) = w_1$ $\qquad$ $head(w_1 w_2 \cdots) = w_1$

stav premennej y_i po $(k-1)$–modifikáciách:

$y_i := f^{k-1}(a)$ $\qquad$ $y_i := w_{k-1} w_k w_{k+1} \cdots$

stav po k–tej modifikácii pracovnej premennej y_i:

$y_i := f^k(a)$ $\qquad$ $y_i := w_k w_{k+1} \cdots$
$i_H(p)(\text{"}f^k(a)\text{"}) = w_k$ $\qquad$ $head(w_k w_{k+1} \cdots) = w_k$

Podstata dôkazu vety 9 spočíva vo využití nasledujúcich vlastností redukcie automatov na schémy.

Veta 11 *Každá interpretácia $\mathcal{I}_H$ sa dá popísať nekonečnou postupnosťou $w \in \{0,1\}^{\infty}$ tak, že platí $w_i = x$ práve vtedy, keď $i_H(p)(f^i(a)) = x$.*

- $Accept(A) = \emptyset \;\Leftrightarrow\; \forall \mathcal{I}_H : (S', \mathcal{I}_H)$ *diverguje,*
- $Loop(A) \neq \emptyset \;\Leftrightarrow\; \exists \mathcal{I}_H : (S'', \mathcal{I}_H)$ *diverguje.*

Dôkaz: V oboch prípadoch využijeme jednoduchú schému dôkazu sporom: aby sme dokázali $p \Leftrightarrow q$ predpokladáme, že platí p a neplatí q a naopak.
Nech $Accept(A) = \emptyset$ ale existuje $\mathcal{I}_H$ taká, že sa symbolický výpočet $(S', \mathcal{I}_H)$ zastaví. Keďže $i_H(p)$ je zladená s nejakým vstupným slovom w a akceptujúci stav je jediný, automat A musí slovo w akceptovať, čo je v spore s predpokladom.

Ostatné časti dôkazu vety 11 sa opierajú o analogické úvahy (cvičenie).

2.3 Nerozhodnuteľnosť problémov zastavenia, ekvivalencie a izomorfizmu

Veta 12 *Problém zastavenia schém v* $\mathcal{S}_1$ *je nerozhodnuteľný (je však čiastočne rozhodnuteľný).*

Myšlienka dôkazu: Problém zastavenia je komplementárnym problémom k problému divergenciu pre nejakú interpretáciu. Z tohoto titulu je problém zastavenia nerozhodnuteľný.
Myšlienka procedúry "čiastočne riešiacej" tento problém môže byť založená na simulácii nedeterministického symbolického výpočtu programovej schémy pomocou prehľadávania do šírky.

Veta 13 *Problémy ekvivalencie a izomorfizmu schém z* $\mathcal{S}_1$ *nie sú ani čiastočne rozhodnuteľné.*

Myšlienka dôkazu sporom:

- Nech S_1 je ľubovolná schéma a S_2 divergujúca schéma. Z rozhodnuteľnosti problému ekvivalencie schém by vyplývala aj rozhodnuteľnosť divergencie S_1.
- Nech S_1 je ľubovolná schéma a S_2 schéma, ktorá vznikne nahradením koncového príkazu príkazom **end goto end**. S_1 je izomorfná s S_2 práve vtedy, keď S_1 diverguje. Z rozhodnuteľnosti problému izomorfizmu schém by vyplývala aj rozhodnuteľnosť divergencie.

Príkazom **end goto end** sme si v dôkaze trochu "zjednodušili" úlohu, pretože formálne koncový príkaz neobsahuje príkaz skoku. Presnú konštrukciu dôkazu nechávame pre čitateľa ako jednoduché technické cvičenie.

2.4 Vlastnosti podtried štandardných schém

Keďže všetky základné vlastnosti programových schém sú pre triedu štandardných schém nerozhodnuteľné, študujú sa tieto vlastnosti na rôznych podtriedach štandardných schém. Cieľom je definovať "rozumné" triedy schém s rozhodnuteľnými vlastnosťami, resp. odhaliť "hranice" nerozhodnuteľnosti.

2.4.1 Podtriedy štandardných schém

Rôzne podtriedy štandardných schém sa dajú definovať buď syntaktickými ohraničeniami (stanovením dodatočných syntaktických pravidiel) alebo sémantickými ohraničeniami (určením vlastností prípustných interpretácií).

1. Syntaktické ohraničenia: dodatočné syntaktické ohraničenia

stromové schémy – graf schémy neobsahuje cyklus,

Janovove schémy – monadické symboly nad jedinou pracovnou premennou,

konzervatívne schémy – premenná z ľavej strany priradenia sa vyskytuje aj na pravej strane priraďovacieho príkazu,

progresívne schémy – premenná z ľavej strany priraďovacieho príkazu je použitá na pravej strane nasledujúceho príkazu priradenia.

2. Sémantické ohraničenia: vlastnosti prípustných interpretácií

liberálne schémy – pri ľubovolnej interpretácii $\mathcal{I}_H$ sa ten istý výraz počíta nanajvýš jeden krát,

voľné schémy – keď pre každú cestu vedúcu z počiatočného príkazu existujú interpretácia a ohodnotenie vstupných premenných také, že výpočet sleduje túto cestu,

dosiahnuteľné schémy – obsahuje len dosiahnuteľné príkazy, t.j. pre každý príkaz v schéme existuje interpretácia, pri ktorej sa príkaz vykoná,

priechodné schémy – obsahuje len dosiahnuteľné hrany, t.j. pre každú hranu v schéme existuje interpretácia, pri ktorej sa hranou "prejde".

V prípade schém, definovaných na základe sémantických ohraničení, je príslušnosť k triede (zvyčajne) nerozhodnuteľnou vlastnosťou. Explicitne túto vlastnosť dokážeme pre triedu voľných schém.

2.4.2 Voľné schémy

Definícia 14 *Schéma* S *je voľná práve vtedy, keď pre každú cestu vedúcu z počiatočného príkazu existujú interpretácia* $\mathcal{I}$ *a ohodnotenie vstupných premenných* v *také, že výpočet* $(S, \mathcal{I}, v)$ *sleduje túto cestu.*

Inými slovami, schéma je voľná, ak každá konečná cesta začínajúca počiatočným príkazom je počiatočným segmentom nejakého výpočtu.

Jednoduchým dôsledkom definície je, že schéma, ktorá pri symbolickom výpočte testuje tým istým predikátovým symbolom p ten istý vektor termov $\bar{t}$ dvakrát (resp. viackrát) nemôže byť voľná. Ak označíme hrany vedúce z **if**–príkazu $p^0(\bar{t})$ resp. $p^1(\bar{t})$, potom cesta $\cdots\ p^0(\bar{t}) \cdots\ p^1(\bar{t}) \cdots$ evidentne nereprezentuje žiadny výpočet (je neprípustná).

Príklad: Analýzou schémy S_6 zistíme, že počet opakovaní cyklov 1: ... **goto 1** a 4: ... **goto 4** musí byť rovnaký. S_6 teda nie je voľná, napr. 3–násobné otočenie prvého cyklu a 2–násobné opakovanie druhého cyklu nie je "prípustnou cestou".

S_6: **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 $[z] := [y_1]$

Cvičenie: Uvažujme triedu schém $\mathcal{S}_2$, ktorú dostaneme modifikáciou počiatočného príkazu v definícii triedy $\mathcal{S}_1$ na **begin** $[y_1, y_2] := [a, b]$. Dokážte, že $\mathcal{S}_2$ je triedou voľných schém.

1. Základné vlastnosti voľných schém:

Veta 15 *Problémy divergencie a zastavenia voľných schém sú rozhodnuteľné.*

Dôkaz: Voľná schéma diverguje práve vtedy, keď v grafe schémy neexistuje cesta z počiatočného do koncového príkazu. Ak totiž existuje, musí byť prípustná, t.j. existuje interpretácia a ohodnotenie, pri ktorých sa výpočet zastaví (čo je v spore s predpokladom).
Analogicky, voľná schéma sa zastaví, ak v grafe schémy neexistuje cyklus.

Veta 16 *Problém izomorfizmu voľných schém je rozhodnuteľný.*

Dôkaz: Ku každej voľnej schéme S priradíme regulárnu gramatiku nasledujúcou konštrukciou. Neterminálnymi symbolmi sú stavy s_i, zodpovedajúce jednotlivým návestiam v schéme S (s_b a s_e sú stavy pre návestia **begin** a **end**). Množinu terminálnych symbolov definujeme tým, že ku každému priradeniu (v príkaze s návestím i) priradíme terminálny symbol a_i a ku každej podmienke p priradíme dvojicu terminálnych symbolov p^+, p^-. Počiatočným stavom je s_b a pravidlá zostrojíme z príkazov schémy podľa nasledujúceho vzoru:

begin $[\overline{y}] := [t_1(\overline{x}), \cdots, t_n(\overline{x})]$	$s_b \to a_b s_1$
i: $[\overline{y}] := [t_1(\overline{x}, \overline{y}), \cdots, t_n(\overline{x}, \overline{y})]$	$s_i \to a_i s_{i+1}$
i: **goto** k	$s_i \to s_k$
i: **if** $p(\overline{x}, \overline{y})$ **then** $[\overline{y}] := [t_1(\overline{x}, \overline{y}), \cdots, t_n(\overline{x}, \overline{y})]$	$s_i \to p^+ a_i s_{i+1}$
	$s_i \to p^- s_{i+1}$
i: **if** $p(\overline{x}, \overline{y})$ **then goto** k	$s_i \to p^+ s_k$
	$s_i \to p^- s_{i+1}$
end $[\overline{z}] := [t_1(\overline{x}, \overline{y}), \cdots, t_n(\overline{x}, \overline{y})]$	$s_e \to a_e$

Označme $L(S)$ regulárny jazyk, generovaný regulárnou gramatikou odvodenou z voľnej schémy S. Z konštrukcie vyplýva, že v prípade voľných schém, ku každej Herbrandovskej interpretácii predikátových symbolov (t.j. ku každému symbolickému výpočtu) zodpovedá práve jedno slovo v

jazyku $L(S)$.
Ako cvičenie doporučujeme dokázať, že dve kompatibilné voľné schémy S_1, S_2 sú izomorfné práve vtedy, keď jazyky $L(S_1)$ a $L(S_2)$ generujú tie isté množiny slov (využite napr. vlastnosť $p^+L + p^-L = L$).
Z rozhodnuteľnosti posledne spomenutého problému vyplýva rozhodnuteľnosť izomorfizmu schém.

Poznámka: Skúste "našiť" myšlienku dôkazu na izomorfizmus, definovaný na základe histórii konfigurácií (resp. iných histórií).

Problém 17 *Problém ekvivalencie voľných schém je zatiaľ otvorený.*

2. O ďalších vlastnostiach voľných schém:

Tvrdenie 18 *Existuje štandardná schéma S, ku ktorej neexistuje ekvivalentná voľná schéma.*

Náčrt dôkazu: Uvažujme schému S_6 z časti 2.4.2. Z analýzy výstupných hodnôt vyplýva, že v závislosti na interpretácii sú výsledkami termy typu $g^k f^k x$. Podľa dôkazu vety 16 voľné schémy "generujú regulárne výsledné termy", takže uvedená schéma sa nedá preložiť do ekvivalentnej voľnej schémy.

V dôsledku tohoto tvrdenia neexistuje ani algoritmus, transformujúci danú štandardnú schému S do ekvivalentnej voľnej schémy.

Veta 19 *Problém príslušnosti ľubovoľnej štandardnej schémy do triedy voľných schém nie je ani čiastočne rozhodnuteľný.*

Dôkaz: tejto vety sa dosiahne redukciou Postovho problému priradenia na problém voľnosti.

3. Postov problém priradenia:

Postov systém nad abecedou $\Sigma = \{a, b\}$ tvorí konečná množina

$$C = \{(u_1, v_1), \cdots, (u_n, v_n)\}$$

dvojíc slov $u_i, v_i \in \Sigma^*$. *Riešením Postovho systému* je k–tica $(i_1, i_2, \cdots, i_k)$ (kde $1 \le i_j \le n$, $k \ge 1$) taká, že

$$u_{i_1} u_{i_2} \cdots u_{i_k} = v_{i_1} v_{i_2} \cdots v_{i_k}.$$

Príklad: Štvorica $(2, 1, 1, 3)$ je riešením Postovho systému $C = \{(b, bbb), (babbb, ba), (ba, a)\}$, pretože *babbb b b ba = ba bbb bbb a*.
Na druhej strane systém $C = \{(ab, bbb), (a, ba), (b, bb)\}$ nemá riešenie, pretože $|v_i| > |u_i|$.

Postov problém priradenia – Má daný Postov systém nad abecedou Σ riešenie?

Veta 20 *Postov problém priradenia je nerozhodnuteľný (je však čiastočne rozhodnuteľný).*

Dôsledok 21 *Duálny problém nie je ani čiastočne rozhodnuteľný.*

4. Redukcia problému voľnosti schémy na Postov problém: Ku každému Postovmu systému $C_n = \{(u_1, v_1), \cdots, (u_n, v_n)\}$ nad abecedou $\Sigma = \{a, b\}$ skonštruujeme schému S_{C_n} takú, že S_{C_n} nie je voľná práve vtedy keď C_n má riešenie, t.j.

S_{C_n} je voľná práve vtedy, keď C_n nemá riešenie.

Veta 22 *Problém voľnosti schémy nie je ani čiastočne rozhodnuteľný.*

Dôkaz: Princíp redukcie budeme demonštrovať na príklade konštrukcie schémy S_{C_4}, zodpovedajúcej Postovmu systému C_4.

S_{C_4}: **begin** $[y_1, y_2, y_3] := [a, a, b]$
1: **if** $p(y_3)$ **then goto** 8
2: $[y_3] := [f(y_3)]$
3: **if** $p(y_3)$ **then goto** 10
4: $[y_3] := [f(y_3)]$
5: **if** $p(y_3)$ **then goto** 12
6: $[y_1, y_2] := [u_4(y_1), v_4(y_2)]$
7: **goto** 13
8: $[y_1, y_2] := [u_1(y_1), v_1(y_2)]$
9: **goto** 13
10: $[y_1, y_2] := [u_2(y_1), v_2(y_2)]$
11: **goto** 13
12: $[y_1, y_2] := [u_3(y_1), v_3(y_2)]$
13: $[y_3] := [f(y_3)]$
14: **if** $p(y_3)$ **then goto** 17
15: $[y_3] := [f(y_3)]$
16: **goto** 1
17: **if** $p(y_1)$ **then goto end**
18: **if** $p(y_2)$ **then goto end**
19: $[y_3] := [f(y_3)]$
end $[z] := [y_3]$

Pri konštrukcii schémy S_{C_n} (nezávisle od n) sú použité nasledujúce symboly:

$$F^0 = \{a, b\}, \quad F^1 = \{f, f_a, f_b\}, \quad B^1 = \{p\}, \quad X_y = \{y_1, y_2, y_3\}.$$

Premennou y_3 sa kontroluje generovanie nejakej k–tice $(i_1, \cdots i_k)$. Ku každej Herbrandovskej interpretácii s ohodnotením predikátu p zodpovedá práve jedna k–tica. V danom príklade napr. interpretácia $p^0(b)$, $p^1(fb)$, $p^0(f^2b)$, $p^1(f^3b)$, $p^0(f^4b)$, $p^1(f^5b)$, $p^1(f^6b)$ generuje trojicu (2,1,1).

Priradenie do y_1 (y_2) simuluje postupnú kompozíciu u_i (v_i). Označenie $[y_i]$:=$[w(y_i)]$, kde $w = \sigma_1 \cdots \sigma_m \in \Sigma^*$ je ľubovolné slovo, je skratkou pre priradenie $[y_i]$:=$[f_{\sigma_1}(f_{\sigma_2}(\cdots f_{\sigma_m}(y_i)\cdots))]$. Pokračujúc v uvedenom príklade, vygenerovaním trojice (2,1,1) sa zárove generujú slová $y_1 = u_2u_1u_1$ a $y_2 = v_2v_1v_1$.

"Nevoľnosť" schémy S_{C_n} vzniká len vtedy, keď po skončení konštrukcie slov platí $y_1 = y_2$. Cesta 17–18–**end** je neprípustná ak $y_1 = y_2$, t.j. práve vtedy, keď má C_4 riešenie.

2.4.3 Janovove schémy

1. Syntax Janovových schém: Trieda Janovových schém je definovaná nasledujúcimi syntaktickými ohraničeniami:

- $F^0 = \emptyset$ a $F^i = \emptyset, B^i = \emptyset$ pre $i > 1$,
- $X_x = \{x\}, X_y = \{y\}, X_z = \{z\}$,
- symboly sa aplikujú len na y.

Janovove schémy sú podtriedou monadických schém (obsahujú len unárne symboly).

Skutočnosť, že Janovove schémy obsahujú len jednu premennú, možno interpretovať tak, že premenná reprezentuje "monolitný" stav. Tento stupe abstrakcie neuvažuje štruktúru stavu (pracovné premenné).

2. Vzťah Janovových a voľných schém:

Veta 23 *Problém voľnosti Janovovej schémy je rozhodnuteľný.*

Dôkaz: Evidentne, daná Janovova schéma nie je voľná práve vtedy, keď v nej existuje cesta obsahujúca dva rovnaké testovacie príkazy (t.j. s rovnakým predikátovým symbolom), medzi ktorými nie je priradenie.

Veta 24 *Každá Janovova schéma sa dá preložiť do ekvivalentnej voľnej Janovovej schémy.*

Myšlienka dôkazu: Existuje tzv. úplná množina transformácií, ktorá transformuje ľubovoľnú Janovovu schému do kánonického (normálneho) tvaru. Janovove schémy v kánonickom tvare sú voľné.
plná množina (ekvivalentných) transformácií sa opiera o základnú transformáciu, ktorou sa odstraujú dva za sebou nasledujúce testy pracovnej premennej predikátom p, medzi ktorými premenná nie je modifikovaná.

3. Vlastnosti Janovových schém: Všetky základné vlastnosti sú pre triedu Janovových schém rozhodnuteľné.

Veta 25 *Problémy divergencie, zastavenia a izomorfizmu Janovových schém sú rozhodnuteľné.*

Dôkaz: Tvrdenie vyplýva z preložiteľnosti Janovových schém do voľných schém a rozhodnuteľnosti uvedených problémov v triede voľných schém.

Veta 26 *Problém ekvivalencie Janovových schém je rozhodnuteľný.*

Náčrt dôkazu: Pri dokazovaní sa opierame o nasledujúcu vlastnosť. Dve Janovove schémy sú ekvivalentné práve vtedy, keď pre každú Herbrandovskú interpretáciu s ohodnotením predikátov sa buď oba výpočty nezastavia, alebo sa oba zastavia a "generujú" rovnaké reťazce $f_k \cdots f_1 f_0 x$.

2.4.4 Na hranici medzi rozhodnuteľnosťou a nerozhodnuteľnosťou

Syntaktické ohraničenia triedy štandardných schém môžu viesť k triedam, pre ktoré sú rozhodovacie problémy alebo nerozhodnuteľné (napr. trieda schém $\mathcal{S}_1$) alebo rozhodnuteľné (napr. trieda Janovových schém). V nasledujúcom prehľade tried schém ilustrujeme jemnosti, od ktorých môže záležať rozhodnuteľnosť resp. nerozhodnuteľnosť problému divergencie. Triedy schém $\mathcal{S}_1$ a $\mathcal{S}_2$ z nasledujúcej tabuľky sú (syntaktickým) zovšeobecnením tried $\mathcal{S}_1, \mathcal{S}_2$, ktoré boli definované v predchádzajúcich častiach tejto kapitoly.

$\mathcal{S}_1$ – divergencia nerozhodnuteľná: $\{y_1, y_2\}$, $\{a, f^1\}$, $\{p^1\}$

- **begin** $[y_1, y_2] := [a, a]$, **end** $[z_1, z_2] := [y_1, y_2]$
- $p(y_1)$, $p(y_2)$, $[y_1] := [f(y_1)]$, $[y_2] := [f(y_2)]$

$\mathcal{S}_2$ – divergencia rozhodnuteľná: $\{y_1, y_2\}$, $\{a, b, f^1\}$, $\{p^1\}$

- **begin** $[y_1, y_2] := [a, b]$, **end** $[z_1, z_2] := [y_1, y_2]$
- $p(y_1)$, $p(y_2)$, $[y_1] := [f(y_1)]$, $[y_2] := [f(y_2)]$

$\mathcal{S}_3$ – divergencia nerozhodnuteľná: $\{y_1, y_2\}$, $\{x\}$, $\{f^1\}$, $\{p^1\}$

- **begin** $[y_1, y_2] := [f(x), f(x)]$, **end** $[z_1, z_2] := [y_1, y_2]$
- $p(y_1)$, $p(y_2)$, $[y_1] := [f(y_1)]$, $[y_2] := [f(y_2)]$

$\mathcal{S}_4$ – divergencia nerozhodnuteľná: $\{y_1, y_2\}$, $\{x_1, x_2\}$, $\{f^1\}$, $\{p^1\}$

- **begin** $[y_1, y_2] := [x_1, x_2]$, **end** $[z_1, z_2] := [y_1, y_2]$
- $p(y_1)$, $p(y_2)$, $[y_2] := [y_1]$, $[y_1] := [f(y_1)]$, $[y_2] := [f(y_2)]$

$\mathcal{S}_5$ – divergencia rozhodnuteľná: $\{y_1, y_2\}$, $\{x_1, x_2\}$, $\{f^1\}$, $\{p^1\}$

- $p(y_1)$, $p(y_2)$, $[y_1] := [f(y_1)]$, $[y_2] := [f(y_2)]$

$\mathcal{S}_6$ – divergencia nerozhodnuteľná: $\{y_1, y_2\}$, $\{x_1, x_2\}$, $\{f^1\}$, $\{p^1\}$

- $p(y_1)$, $[y_1] := [y_2]$, $[y_1] := [f(y_1)]$, $[y_2] := [f(y_2)]$

$\mathcal{S}_7$ – divergencia rozhodnuteľná: $\{y_1, y_2\}$, $\{x_1, x_2\}$, $\{f^1, g^1\}$, $\{p^1\}$

- $p(y_1)$, $p(y_2)$, $[y_2] := [y_1]$, $[y_1] := [f(y_1)]$, $[y_2] := [g(y_2)]$

$\mathcal{S}_8$ – divergencia rozhodnuteľná: $\{y_1, y_2\}$, $\{x_1, x_2\}$, $\{f^1\}$, $\{p^2\}$

- $p(y_1, y_2)$, $[y_2] := [y_1]$, $[y_1] := [y_2]$, $[y_1] := [f(y_1)]$, $[y_2] := [f(y_2)]$

Kľúčovou vlastnosťou tried $\mathcal{S}_2, \mathcal{S}_5, \mathcal{S}_7$ je, že hodnoty pracovných premenných y_1, y_2 sú "oddeliteľné". Existuje interpretácia a ohodnotenie vstupov, pri ktorých sú hodnoty týchto pracovných premenných vždy rôzne, t.j. predikátom p sa nikdy netestuje tá istá hodnota viackrát. Na druhej strane pri triedach $\mathcal{S}_1, \mathcal{S}_3, \mathcal{S}_4, \mathcal{S}_6$ táto vlastnosť všeobecne neplatí (napr. v $\mathcal{S}_6$ je problémom prenos obsahu z y_2 do y_1).

V kontexte problému prekladu tried schém (pozri kapitolu 4) je pozoruhodný fakt, že v rozhodnuteľných prípadoch sa dajú schémy z daných tried preložiť (včšinou) do voľných schém.

Kapitola 3

Porovnávanie tried programových schém

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

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

Príklad: pre triedy voľných $\mathcal{V}$, Janovových $\mathcal{J}$ a štandardných schém $\mathcal{S}$ platí

$$\mathcal{J} \sqsubset \mathcal{V} \sqsubset \mathcal{S}.$$

Vlastnosti vyplývajú jednak z výsledkov predchádzajúcej kapitoly a dvak priamo z definícií uvažovaných tried schém (každá voľná schéma je štandardná; voľné schémy môžu, na rozdiel od Janovových, využiť viac pracovných premenných).

Príklad: Uvažujme schémy s rôznymi riadiacimi konštrukciami:

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

W: **begin** $[y_1, y_2] := [x, a]$
  while $p(y_1)$ **do** $[y_1, y_2] := [f(y_1), g(y_1, y_2)]$ **od**
end $[z] := [y_2]$

$$
\begin{aligned}
R\colon\quad & \textbf{begin } [y_1, y_2] := [x, a] \\
& \qquad \phi(y_1, y_2) \impliedby \textbf{if } p(y_1) \textbf{ then } y_2 \textbf{ else } \phi(f(y_1), g(y_1, y_2)) \\
& \textbf{end } [z] := [\phi(x, a)]
\end{aligned}
$$

3.1 Štruktúrované schémy

1. Syntax štruktúrovaných schém:

Symboly – ako pri štandardných schémach,

Termy a predikáty – ako pri štandardných schémach,

Príkazy – $C = \{st, st_i, \cdots\}$

1. počiatočný – **begin** $[\overline{y}] := [t_1(\overline{x}), \cdots, t_n(\overline{x})]$
2. koncový – **end** $[\overline{z}] := [t_1(\overline{x}, \overline{y}), \cdots, t_n(\overline{x}, \overline{y})]$
3. priraďovací – $[\overline{y}] := [t_1(\overline{x}, \overline{y}), \cdots, t_n(\overline{x}, \overline{y})]$
4. zložený – st_1 ; st_2,
5. príkaz cyklu – **while** $p(\overline{x}, \overline{y})$ **do** st **od**
6. podmienkový – **if** $p(\overline{x}, \overline{y})$ **then** st_1 **else** st_2 **fi**

Štruktúrované schémy – $\mathcal{W} = \{W, W_i, \cdots\}$

$$
\begin{aligned}
& \textbf{begin } [\overline{y}] := [t_1(\overline{x}), \cdots, t_n(\overline{x})] \\
& \qquad st \\
& \textbf{end } [\overline{z}] := [t_1(\overline{x}, \overline{y}), \cdots, t_n(\overline{x}, \overline{y})]
\end{aligned}
$$

2. Sémantika štruktúrovaných schém:

Interpretácia, program, stav a výsledok výpočtu – ako pri štandardných schémach,

Výpočet – prirodzená operačná sémantika so simultánnym priradením (analogicky, ako pri štandardných schémach).

3.2 Porovnanie štandardných a štruktúrovaných schém

Veta 27 *Ku každej štruktúrovanej schéme W existuje ekvivalentná štandardná schéma S_W, t.j.*

$$\mathcal{W} \sqsubseteq \mathcal{S}.$$

Dôkaz: indukciou vzhľadom na štruktúru tela st schémy W. Ako úvodný krok indukcie preložíme st na $1 : st$. äalší priebeh konštrukcie závisí od príkazu $i : st$:

- $i : [\overline{y}] := [\overline{t}(\overline{x}, \overline{y})]$ – pri preklade opíšeme,
- i : **if** q **then** st_1 **else** st_2 – preložíme na fragment schémy v stpci 1,
- $i : st_1; st_2$ – preložíme do fragmentu zo stpca 2,
- i : **while** q **do** st **od** – preložíme do fragmentu v stpci 3.

i : **if** q **then goto** $j+1$	i :	i : **if** q **then goto** $i+2$
$i+1$:	$\cdots$ st_1	$i+1$: **goto** $j+1$
$\cdots$ st_2	$j-1$:	$i+2$:
$j-1$:	j :	$\cdots$ st
j : **goto** k	$\cdots$ st_2	$j-1$:
$j+1$:	$k-1$:	j : **goto** i
$\cdots$ st_1	k : $\cdots$	$j+1$: $\cdots$
$k-1$:		
k : $\cdots$		

Po konečnom počte indukčných krokov výsledný program S_W neobsahuje štruktúrované príkazy. Dôkaz ekvivalencie W a S_W je priamočiary.

Veta 28 *Neexistuje štruktúrovaná schéma $W \in \mathcal{W}$, ekvivalentná s voľnou štandardnou schémou*

$$
\begin{aligned}
S\colon\quad & \textbf{begin } [y] := [x] \\
& \quad 1\colon \textbf{ if } p_1(y) \textbf{ then goto } 3 \\
& \quad 2\colon \textbf{ goto end} \\
& \quad 3\colon \textbf{ if } p_2(y) \textbf{ then goto } 5 \\
& \quad 4\colon \textbf{ goto end} \\
& \quad 5\colon [y] := [f(y)] \\
& \quad 6\colon \textbf{ goto } 1 \\
& \textbf{end } [z] := [y]
\end{aligned}
$$

Dôkaz: predpokladajme, že existuje $W \in \mathcal{W}$ taká, že $W \equiv S$. Evidentne W musí obsahovať aspo jeden cyklus s podmienkou p_1 (alebo p_2). Vzhľadom na predpoklad $W \equiv S$ existujú voľné interpretácie $\mathcal{I}_H$, pri ktorých sa cyklus **while** p_1 **do** $\cdots$ **od** vykoná. Uvažujme $\mathcal{I}_H$:

- $i(p_1)(t) = 1$ pre všetky $t \in \mathcal{H}$,

- $i(p_2)(t) = 1$ len pre tie $t \in \mathcal{H}$, ktoré sa vyskytujú pri výpočte $(W, \mathcal{I}_H, \text{“}x\text{”})$, kým sa nezačne výpočet cyklu s predikátom p_1 ($i(p_2)(t) = 1$ len pre konečný počet termov z $\mathcal{H}$).

Potom sa symbolický výpočet $(S, \mathcal{I}_H, \text{“}x\text{”})$ sa zastaví, zatiaľ čo $(W, \mathcal{I}_H, \text{“}x\text{”})$ cyklí – spor s predpokladom $S \equiv W$.

Dôsledok 29 $\mathcal{W} \sqsubset \mathcal{S}$.

3.3 Rekurzívne schémy

1. Syntax rekurzívnych schém:

Symboly –

- funkčné a predikátové symboly – ako pri štandardných schémach,
- vstupné a pracovné premenné – ako pri štandardných schémach,
- výstupná premenná – z,
- funkčné premenné – $\Phi \ = \bigcup_{i=1}^{\infty} \Phi^i$,
  - $\Phi^n : \{\phi, \phi_i, \cdots\}$ – n–árne funkčné premenné.

Termy – $T = \{t[\overline{\phi}](\overline{x}, \overline{y}), u, s, \cdots\}$

- základ ako pri štandardných schémach,
- ak $\phi \in \Phi^n, t_1, \cdots, t_n \in T$, potom $\phi(t_1, \cdots, t_n) \in T$,
- ak $q \in B, t_1, t_2 \in T$, potom **if** q **then** t_1 **else** $t_2 \ \in T$.

Rekurzívne schémy – $\mathcal{R} = \{R, R_i, \cdots\}$

$$\begin{array}{ll} R\colon & \textbf{begin } [\overline{y}] := [\overline{u}(\overline{x})] \\ & \quad \phi_1(\overline{y}) \Longleftarrow t_1[\overline{\phi}](\overline{x}, \overline{y}) \\ & \quad \quad \vdots \\ & \quad \phi_n(\overline{y}) \Longleftarrow t_n[\overline{\phi}](\overline{x}, \overline{y}) \\ & \textbf{end } [z] := [s[\overline{\phi}](\overline{x})] \end{array}$$

2. Neformálna sémantika rekurzívnych schém – rekurzívne programy:

Interpretácia – $\mathcal{I} = (D, i)$

Rekurzívny program – $P = (R, \mathcal{I})$

Ohodnotenie vstupných premenných – $v : X_x^k \mapsto D^k$

Výpočet programu P so vstupom v – $(R, \mathcal{I}, v)$

- výpočtová postupnosť – $\alpha_0 \Rightarrow \alpha_1 \Rightarrow \cdots \Rightarrow \alpha_i \Rightarrow \cdots$,
- stav výpočtu – term α_i,
- vstupný term $\alpha_0 = i(s[\overline{\phi}](v(x_1), v(x_2), \cdots, v(x_n)))$,
- krok výpočtu $\alpha_i \Rightarrow \alpha_{i+1}$ pozostáva z dvoch fáz:
  - stav α_{i+1} vytvoríme zo stavu α_i nahradením najľavejšieho a najvnútornejšieho výskytu symbolu ϕ_j, zodpovedajúcim výrazom $i(t_j[\overline{\phi}](\cdots))$,
  - v stave α_{i+1}, popísanom v predchádzajúcej fáze, uskutočníme všetky možné zjednodušenia, definované interpretáciou (preddefinovanými operáciami).

Výsledok výpočtu – $val(R, \mathcal{I}, v)$

- $val(R, \mathcal{I}, v) = d$ – ak sa d nedá ani prepísať rekurzívnymi definíciami ani zjednodušiť,
- $val(R, \mathcal{I}, v)$ nie je definovaná – v prípade nekonečného výpočtu.

Príklad: Uvažujme rekurzívnu schému R_1 a interpretáciu $\mathcal{I}$:

$$\begin{array}{ll} R_1\colon & \textbf{begin } [y_1, y_2] := [x, x] \\ & \quad \phi(y_1, y_2) \Longleftarrow \textbf{if } p(y_1) \textbf{ then } f_1(y_2) \\ & \quad \quad \textbf{else if } p(y_2) \textbf{ then } \phi(f_2(y_1), a) \\ & \quad \quad \quad \textbf{else } \phi(f_2(y_1), \phi(y_1, f_2(y_2))) \\ & \textbf{end } [z] := [\phi(x, x)] \end{array}$$

$\mathcal{I}$	N
a	1
p	$y = 0$
f_1	$y + 1$
f_2	$y - 1$

Program $P = (R_1, \mathcal{I})$ počíta známu Ackermannovu funkciu:

$$
\begin{array}{ll}
P: & \textbf{begin } [y_1, y_2] := [x, x] \\
& \quad \phi(y_1, y_2) \Longleftarrow \textbf{if } y_1 = 0 \textbf{ then } y_2 + 1 \\
& \qquad\qquad\qquad\qquad \textbf{else if } y_2 = 0 \\
& \qquad\qquad\qquad\qquad\quad \textbf{then } \phi(y_1 - 1, 1) \\
& \qquad\qquad\qquad\qquad\quad \textbf{else } \phi(y_1 - 1, \phi(y_1, y_2 - 1)) \\
& \textbf{end } [z] := [\phi(x, x)]
\end{array}
$$

Výpočet programu $(R_1, \mathcal{I}, \{x \mapsto 1\})$:

$$\underline{\phi}(1,1) \Rightarrow \phi(0, \underline{\phi}(1,0)) \Rightarrow \phi(0, \underline{\phi}(0,1)) \Rightarrow \underline{\phi}(0,2) \Rightarrow 3$$

Podčiarknuté funkčné premenné boli v nasledujúcom kroku (fáze) "nahradené" zodpovedajúcim výrazom na pravej strane rekurzívnej definície.

3. Model pre formálny popis operačnej sémantiky: Modelom pre popis operačnej sémantiky deklaratívnych programovacích jazykov sú *systémy na prepisovanie termov* (trs – term rewriting systems). V nasledujúcich riadkov sú zhrnuté základné pojmy z teórie prepisovania termov.

Prepisovacie pravidlo: orientovaná dvojica termov (orientovaná rovnosť)

$$l \mapsto r,$$

kde $l, r \in T$ a $Var(r) \subseteq Var(l)$.

Systém na prepisovanie termov: konečná množina prepisovacích pravidiel

$$R = \{l_1 \mapsto r_1, \cdots, l_n \mapsto r_n\}.$$

Prepisovací krok: prepísanie uzavretého termu s (termu bez premenných) prepisovacím systémom R do termu t

$$s \to_R t$$

- ak existuje podterm u termu s (označenie $s[u]$) a pravidlo $l \mapsto r \in R$ také, že $u = \sigma l$ (hovoríme, že u je R–redex),
- potom výsledkom prepísania termu s systémom R je term $t = s[\sigma r]$.

Výpočtová postupnosť: relácia $\to_R^*$ (R–odvodenie) – reflexívny a tranzitívny uzáver "prepisovacej" relácie $\to_R$

- term t je R–reducibilný, ak existuje term u taký, že $t \to_R u$ (obsahuje R–redex),
- term t je R–ireducibilný, ak sa nedá prepísať žiadnym pravidlom, t.j. neobsahuje žiadny R–redex,
- term u je R–normálnym tvarom termu t, ak $t \to_R^* u$ a u je R–ireducibilný term,

- R–odvodenie $s \to_R^* t$ je R–normálne, ak t je R–normálnym tvarom termu s.

Vlastnosti prepisovacích systémov: prepisovací systém R

- sa zastaví, ak neexistuje nekonečné odvodenie $s_0 \to_R s_1 \to_R \cdots \to_R s_n \to_R \cdots$,
- je jednoznačne normalizujúci, ak pre každé dve R–normálne odvodenia $s \to_R^* t$ a $s \to_R^* u$ platí $t = u$.
- je kanonický, keď sa zastaví a je jednoznačne normalizujúci.

4. Formálna operačná sémantika rekurzívnych programov: Uvažujme prepisovací systém

$$P = R + Z,$$

kde R je množina interpretovaných rekurzívnych definícií daného rekurzívneho programu a Z je kánonický prepisovací systém zodpovedajúci pravidlám, ktoré charakterizujú vlastnosti preddefinovaných funkcií (zjednodušujúce pravidlá).

Výpočtový krok: popisuje "metapravidlo" (riadiace poradie aplikácie prepisovacích pravidiel P):

$$s \Rightarrow_P t \quad \equiv \quad \to_R^{val} \to_Z^*$$

- pri kroku $\to_R^{val}$ sa použije niektoré z pravidiel R na redukcii najľavejšieho z najvnútornejších R–redexov termu t, t.j. najľavejšieho podtermu, ktorého vlastné podtermy neobsahujú symboly funkčných premenných,
- krokom $\to_Z^*$ sa term prepíše do Z–normálneho tvaru.

Výsledok výpočtu: term t je výsledkom rekurzívneho výpočtu programu P, ak je P–normálnym tvarom vzhľadom na prepisovací systém P a výpočtový krok $\Rightarrow_P$, t.j. t je Z–ireducibilným termom, ktorý neobsahuje žiadny symbol funkčnej premennej.

Poznámka: Krok $\to_R^{val}$ (formálne za definuje tzv. výpočtovým pravidlom) zodpovedá volaniu (vyhodnoteniu) parametrov "hodnotou", ktoré sa využíva napr. pri implementácii Lispu. äalšie "výpočtové pravidlá" budeme definovať a analyzovať pri podrobnejšom štúdiu operačnej a denotačnej sémantiky funkcionálnych jazykov (pozri časti 12.6).

3.4 Porovnanie štandardných a rekurzívnych schém

Uvažujme štandardné schémy s jediným výstupom.

Veta 30 *Každá štandardná schéma S sa dá preložiť do ekvivalentnej rekurzívnej schémy R_S, t.j.*

$$\mathcal{S} \sqsubseteq \mathcal{R}.$$

Dôkaz: Pred každý príkaz st_i schémy S predradíme funkčnú premennú $\phi_i(\overline{x}, \overline{y})$, resp. $\phi_b(\overline{x}, \overline{y})$, $\phi_e(\overline{x}, \overline{y})$ (deliace body). Po takejto úprave je každý príkaz "obložený" dvojicou funkčných premenných $\phi_i\ i : st\ \phi_{i+1}$. Pri zmene riadenia (príkaz **goto** j) predpokladáme, že príkaz je obložený $\phi_i\ i : \cdots$ **goto** $j\ \phi_j$. V ďalšom kroku prevedieme spätné substitúcie, na základe ktorých odvodíme vzťahy medzi ϕ_i, ϕ_j:

- $\underline{\phi_b}$ **begin** $[\overline{y}] := [\overline{t}(\overline{x})]$ $\underline{\phi_1}$

  $z \Leftarrow \phi_1(\overline{x}, t(\overline{x}))$

- $\underline{\phi_e}$ **end** $[z] := [t(\overline{x}, \overline{y})]$

  $\phi_e(\overline{x}, \overline{y}) \Leftarrow t(\overline{x}, \overline{y})$

- $\underline{\phi_i}$ $i:$ $[\overline{y}] := [\overline{t}(\overline{x}, \overline{y})]$ $\underline{\phi_{i+1}}$

  $\phi_i(\overline{x}, \overline{y}) \Leftarrow \phi_{i+1}(\overline{x}, \overline{t}(\overline{x}, \overline{y}))$

- $\underline{\phi_i}$ $i:$ **goto** j $\underline{\phi_j}$

  $\phi_i(\overline{x}, \overline{y}) \Leftarrow \phi_j(\overline{x}, \overline{y})$

- $\underline{\phi_i}$ $i:$ **if** $p(\overline{x}, \overline{y})$ **then** $[\overline{y}] := [\overline{t}(\overline{x}, \overline{y})]$ $\underline{\phi_{i+1}}$

  $\phi_i(\overline{x}, \overline{y}) \Leftarrow$ **if** $p(\overline{x}, \overline{y})$ **then** $\phi_{i+1}(\overline{x}, \overline{t}(\overline{x}, \overline{y}))$ **else** $\phi_{i+1}(\overline{x}, \overline{y})$

- $\underline{\phi_i}$ $i:$ **if** $p(\overline{x}, \overline{y})$ **then goto** j $\underline{\phi_j}$

  $\phi_i(\overline{x}, \overline{y}) \Leftarrow$ **if** $p(\overline{x}, \overline{y})$ **then** $\phi_j(\overline{x}, \overline{y})$ **else** $\phi_{i+1}(\overline{x}, \overline{y})$

Dôkaz ekvivalencie pôvodnej štandardnej a skonštruovanej rekurzívnej schémy dostaneme indukciou vzhľadom na výpočty schém:

- funkčné premenné ϕ_i zodpovedajú návestiam štandardnej schémy (simulujú tok riadenia),
- vektor argumentov $\overline{y}$ v rekurzívnych definíciách funkčných premenných ϕ_i zodpovedá vektoru pracovných premenných (simuluje sa ním zmena stavu výpočtu).

Po zjednodušenie systému rekurzívnych definícií jednoduchým dosadením dostaneme ekvivalentné schémy S a R_S.

Príklad:

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

$\phi_b(y_1, y_2) = z = \phi_1(x, a)$
$\phi_1(y_1, y_2) =$ **if** $p(y_1)$ **then** $\phi_e(y_1, y_2)$ **else** $\phi_2(y_1, y_2)$
$\phi_2(y_1, y_2) = \phi_3(f(y_1), g(y_1, y_2))$
$\phi_3(y_1, y_2) = \phi_1(y_1, y_2)$
$\phi_e(y_1, y_2) = y_2$

$z := \phi_1(x, a)$

$\phi_1(y_1, y_2) \Leftarrow$ **if** $p(y_1)$ **then** y_2 **else** $\phi_1(f(y_1), g(y_1, y_2))$

Veta 31 *K rekurzívnej schéme R_A neexistuje žiadna ekvivalentná štandardná schéma.*

R_A: **begin** $[y] := [a]$
  $\phi(y) \Longleftarrow$ **if** $p(y)$ **then** $f(y)$ **else** $h(\phi(g_1(y)), \phi(g_2(y)))$
 end $[z] := [\phi(a)]$

Dôkaz: Uvažujme podtriedu Herbrandových interpretácií $\{\mathcal{I}_n^*\}_{n\geq 0}$:

- Herbrandovo univerzum zo symbolmi a, f, g_1, g_2, h,
- $p(t) = 1$, keď t obsahuje n výskytov symbolov g_i.

Výstupné hodnoty $val(R_A, \mathcal{I}_n^*)$ pre $n = 0, 1, 2, \cdots$:

$n = 0$ $z = s_0 = f(a)$,
$n = 1$ $z = s_1 = h(fg_1a, fg_2a)$,
$n = 2$ $z = s_2 = h(h(fg_1g_1a, fg_2g_1a), h(fg_1g_2a, fg_2g_2a))$, atď.

Evidentne, rekurzívna schéma "počíta" výrazy s_n pre ľubovolné n. Každý výraz s_n sa dá reprezentovať úplným binárnym stromom T_n hbky n, ktorého vnútorné uzly sú označené symbolom h a listy (príslušnými) reťazcami symbolov (podtermami) nad abecedou f, g_i, a.

äalší postup dôkazu pozostáva z nasledujúcich krokov:

- Hra s kamemi na binárnom strome (pravidlá umiestnenia a odobratia kamea).
- Minimálny počet kameov pre pokrytie korea stromu.
- Na pokrytie stromu T_n treba práve $n+1$ kameov.
- Súvislosť medzi hrou a nutnými priradeniami (všetky podstromy sú navzájom rôzne).

V dôsledku predchádzajúcich úvah, na výpočet výrazu s_n treba aspo n premenných. Keďže štandardná schéma má len konečný počet premenných, nemôže "vypočítať" všetky výrazy s_n. K danej rekurzívnej schéme teda nemôže existovať ekvivalentná štandardná schéma.

Dôsledok 32 $\mathcal{S} \sqsubset \mathcal{R}$.

3.5 Čiastočne interpretované schémy

Obohatenie triedy štandardných schém o interpretované objekty (premenné, predikátové a funkčné symboly).

1. Schémy s interpretovanými predikátovými symbolmi: – $\mathcal{W}^p$

- konštanty – *true*, *false*,
- logické spojky: $\wedge, \vee, \neg$.

Vlastnosti: – porovnanie tried schém $\mathcal{W}^p, S$

- $\mathcal{W}^p = S$.

2. Schémy s počítadlami: – $\mathcal{S}^c$

- konečný počet počítadiel – premenných y_c (vstupná inicializácia 0),
- operácie: $y_c := y_c + 1, y_c := y_c - 1$; predikát: $y_c = 0$.

Vlastnosti: – porovnanie tried schém $\mathcal{S}^c, \mathcal{S}^{1c}, \mathcal{S}^{2c}, \mathcal{S}, \mathcal{R}$

- $\mathcal{S}^c \sqsupset \mathcal{S}$, $\mathcal{S}^{1c} = \mathcal{S}$, $\mathcal{S}^{2c} = \mathcal{S}^c$,
- $\mathcal{S}^c \not\sqsupseteq \mathcal{R}$, $\mathcal{S}^c \not\sqsubseteq \mathcal{R}$.

3. Schémy so zásobníkmi: – $\mathcal{S}^s$

- konečný počet zásobníkov – premenných y_s (vstupná inicializácia *empty*),
- operácie: $y_s := push(\overline{y}), \overline{y} := toppop(y_s)$; predikát: $y_s = empty$.

Vlastnosti: – porovnanie tried schém $\mathcal{S}^s, \mathcal{S}^{1s}, \mathcal{S}^{2s}, \mathcal{S}, \mathcal{R}$

- $\mathcal{S}^s \sqsupset \mathcal{S}$,
- $\mathcal{S}^s \not\sqsubseteq \mathcal{R}$, $\mathcal{S}^s \sqsupset \mathcal{R}$, $\mathcal{S}^{1s} = \mathcal{R}$, $\mathcal{S}^{1s} \sqsubset \mathcal{S}^{2s}$,
- $\mathcal{S}^c \sqsubset \mathcal{S}^s$.

Časť II

Správnosť programov

1. Špecifikácia programu: Predpokladáme, že špecifikácia programu je presným popisom požiadaviek, vyjadrujúcim <u>*čo*</u> má navrhovaný program urobiť. Najčastejšie sa vyjadruje pomocou vstupno–výstupnej charakterizácie, popísanej v danom špecifikačnom jazyku dvojicou podmienok:

vstupná podmienka – vymedzuje množinu zmysluplných vstupov,

výstupná podmienka – charakterizuje vlastnosti požadovaných výstupov.

Ako *špecifikačný jazyk* budeme používať predikátový počet 1. rádu (rozšírený o aritmetiku).

2. "Správne" programy: Správnosť programu je relatívna vlastnosť vyjadrujúca, že daný program vyhovuje (spa) danej špecifikácii, t.j. vstupnej podmienke p a výstupnej podmienke q:

totálna správnosť – vzhľadom na podmienky p, q:
pre každý vstup, spajúci vstupnú podmienku p, sa program P zastaví a výstupné hodnoty spajú výstupnú podmienku q,

zastavenie – vzhľadom na podmienku p:
pre každý vstup, vyhovujúci vstupnej podmienke p sa program P zastaví,

čiastočná správnosť – vzhľadom na podmienky p, q:
pre každý vstup, spajúci vstupnú podmienku p, pre ktorý sa program P zastaví, zodpovedajúce výstupné hodnoty spajú výstupnú podmienku q.

3. Metódy dokazovania: Spomedzi metód na dokazovanie správnosti programov spomenieme tri najznámejšie, Floydovu metódu, Hoareovu metódu a metódu intermitentov. Prvé dve metódy boli navrhnuté v prvom rade na dokazovanie čiastočnej správnosti programov. Neskôr boli rozšírené aj na dokazovanie zastavenia programov (uvedieme myšlienku dôkazu zastavenia programu modifikáciou Floydovej metódy). Metóda intermitentov je metódou pre dokazovanie totálnej správnosti.

Krátka charakterizácia spomenutých metód:

Floydova metóda – dôkaz čiastočnej správnosti programu sa redukuje na dôkaz platnosti špeciálne odvodených formúl špecifikačného jazyka,

Hoareova metóda – pri dokazovaní čiastočnej správnosti sa využíva špecializovaný logický (inferenčný) systém pre dokazovanie tzv. invariantných (induktívnych) formúl $\{p\}\ P\ \{q\}$,

Metóda intermitentov – pri dokazovaní totálnej správnosti programu sa využívajú špecializované formuly (intermitenty), zaručujúce, že program aspo raz dospeje do miesta v programe, ku ktorému sú pripojené.

Dôkazové techniky použité v uvedených metódach využívajú pri dokazovaní (čiastočnej) správnosti programov dve základné *indukčné metódy*:

výpočtová indukcia – indukcia vzhľadom na priebeh výpočtu (prechod ciest),

štrukturálna indukcia – indukcia vzhľadom na štruktúru programu resp. štruktúru toku dát.

Kapitola 4

Indukčné metódy pri dokazovaní správnosti

V nasledujúcich troch kapitolách budú prezentované tri základné metódy dokazovania (čiastočnej a/alebo totálnej) správnosti programov. Vychádzajú z rôznych princípov a sú určené pre rôzne triedy programovacích jazykov. Z hľadiska "teórie dokazovania" sa však odlišujú v prvom rade použitými indukčnými technikami.

V tejto kapitolke stručne zhrnieme princípy dvoch základných indukčných techník, výpočtovej a štrukturálnej (noetherovskej) indukcie.

4.1 Výpočtová indukcia

Výpočtová indukcia je indukciou vzhľadom na (vybrané) stavy výpočtu. Pri výpočtovej indukcii treba ukázať

- že daná vlastnosť platí pre vstupný stav,
- ak platí daná vlastnosť pre ľubovoľný stav výpočtu, potom platí aj vo všetkých nasledujúcich stavoch.

Uvažujme nasledujúce označenia:

- $next(P, s)$ – množina stavov, generovaných programom P zo stavu s,
- $next^*(P, s)$ – reflexívny a tranzitívny uzáver $next$,
- $S = next^*(P, s_0)$ – všetky stavy, dosiahnuteľné z daného vstupného stavu s_0,
- ϕ – totálny predikát nad S.

Princíp výpočtovej indukcie:

$$\{\phi(s_0)\ \wedge\ (\forall s \in S)[\phi(s) \Longrightarrow (\forall s' \in next(P, s))\ \phi(s')]\}\ \Longrightarrow\ (\forall s \in S)\phi(s).$$

Princíp výpočtovej indukcie sa využíva vo Floydovej metóde. Stačí uvažovať stavy výpočtu zodpovedajúce deliacim bodom programu a definovať množinu $next(P, s)$ tak, aby pozostávala zo stavov (deliacich bodov), do ktorých sa dostaneme z daného stavu (deliaceho bodu) po všetkých konečných cestách vedúcich z tohoto deliaceho bodu.

4.2 Štrukturálna (noetherovská) indukcia

Štrukturálna indukcia je zovšeobecním (klasickej) matematickej indukcie. Na rozdiel od matematickej indukcie "pracuje" štrukturálna indukcia na čiastočne usporiadanej množine (S,$\succ$).

Nižšie uvedené pravidlo štrukturálnej indukcie platí za predpokladu, že

- $\succ$ je dobre založené čiastočné usporiadanie na S,
- ϕ je totálny predikát nad S.

Princíp štrukturálnej indukcie:

$$(\forall a \in S)\ \{[(\forall b \in S)(a \succ b\ \Rightarrow\ \phi(b))]\ \Rightarrow\ \phi(a)\}\ \Longrightarrow\ (\forall c \in S)\phi(c)$$

Dôkaz sporom: Predpokladáme, že platí (indukčný) predpoklad ale neplatí dôsledok tvrdenia, t.j. existuje $c \in S$ také, že neplatí $\phi(c)$. Nech M je množina všetkých prvkov, pre ktoré neplatí ϕ. Pretože usporiadanie $\succ$ je dobre založené na S, musí v M existovať minimálny prvok m. Keďže všetky prvky menšie ako m nepatria do M, platí pre ne predikát ϕ. Teda podľa indukčného predpokladu musí platiť aj $\phi(m)$, čo je v spore s predpokladom.

Princíp štrukturálnej indukcie sa využíva v dvoch metódach, Hoareovej metóde a tzv. metóde intermitentov. Používa sa však odlišne.

Indukcia vzhľadom na štruktúru programu – Hoareova metóda

- usporiadanie $\succ$ je definované reláciou "byť podtermom".

Indukcia vzhľadom na tok dát – metóda intermitentov

- relácia $\succ$ je definovaná na hodnotách stavu výpočtu.

Kapitola 5

Floydova metóda

Základom Floydovej metódy je redukcia dôkazu čiastočnej správnosti programu na dokazovanie formúl špecifikačného jazyka (napr. predikátových formúl). Pri zdôvodnení metódy využívame výpočtovú indukciu.

5.1 Dôkaz správnosti Floydovou metódou

Dôkaz čiastočnej správnosti programu Floydovou metódou pozostáva zo štyroch krokov:

- Program rozdelíme *deliacimi bodmi* na konečné cesty (počiatočný, koncový, resp. vnútorné body).
- Pre každý vnútorný deliaci bod sformulujeme *invariant* $I_A(\overline{x},\overline{y})$ – podmienku, ktorá platí pri každom prechode deliacim bodom. Počiatočnému deliacemu bodu zodpovedá vstupná podmienka, koncovému výstupná podmienka.
- Ku každej konečnej ceste AB odvodíme a dokážeme *verifikačnú podmienku*: ak pri prechode deliacim bodom A je splnená podmienka I_A potom po prechode cestou AB bude v deliacom bode B splnená podmienka I_B

  $$\forall \overline{x},\overline{y}\ \ I_A(\overline{x},\overline{y})\ \&\ R_{AB}(\overline{x},\overline{y}) \Rightarrow I_B(\overline{x}, r_{AB}(\overline{x},\overline{y})).$$

  Sémantické vlastnosti cesty:

  - $R_{AB}(\overline{x},\overline{y})$ – podmienka pre prechod cesty AB,
  - $r_{AB}(\overline{x},\overline{y})$ – modifikácia pracovných premenných pri prechode AB.
- Overíme (dokážeme) platnosť všetkých odvodených verifikačných podmienok.

Veta 33 *Ak pre všetky cesty v programe P (definované deliacimi bodmi) platia zodpovedajúce verifikačné podmienky, potom je program P čiastočne správny.*

Dôkaz: tvrdenie vyplýva z tranzitivity implikácie (indukcia vzhľadom na výpočet).

Verifikačné podmienky sú postačujúcimi podmienkami pre čiastočnú správnosť programu. Uvedená metóda sa dá priamočiaro použiť aj pre formulovanie postačujúcich podmienok čiastočnej správnosti programovej schémy.

5.2 Invariant

Invarianty sú podmienky – formuly špecifikačného jazyka (napr. formuly predikátového počtu) viazané na ľubovoľné (deliace) body programu. Charakterizujú vzťah medzi priebehom výpočtu (dynamický pohľad) a platnosťou podmienok viazaných k deliacim bodom výpočtu (statický pohľad).

Invariant je podmienka splnená vždy, keď výpočet programu prechádza deliacim bodom (nie je však zaručené, že výpočet daný bod dosiahne).

Existujú rôzne možnosti vyjadrenia invariantu, vyjadrujúce napr.:

- vzťah vstupných a pracovných premenných $I_A(\overline{x},\overline{y})$,
- vzťah výstupných a pracovných premenných $I_A(\overline{y},\overline{z})$ (podcieľ).

Príklad: Uvažujme program, ktorý pre $x_1, x_2 > 0$ počíta $nsd(x_1,x_2) = max\{u: \ u|x_1 \ \& \ u|x_2\}$.

P: **begin** $[y_1,y_2] := [x_1,x_2]$
1: **if** $y_1 = 0$ **then goto end**
2: **if** $y_2 \geq y_1$ **then goto 5**
3: $[y_1,y_2] := [y_2,y_1]$
4: **goto** 1
5: $[y_2] := [y_2 - y_1]$
6: **goto** 1
end $[z] := [y_2]$

Invariant I – vzťah medzi vstupnými a pracovnými premennými

$$I_1(x_1,x_2,y_1,y_2) = y_1 \geq 0 \ \& \ y_2 \geq 0 \ \& \ (y_1 \neq 0 \ \vee \ y_2 \neq 0) \ \&$$
$$max\{u: \ u|y_1 \ \& \ u|y_2\} = max\{u: \ u|x_1 \ \& \ u|x_2\}$$

Invariant S – vzťah medzi výstupnými a pracovnými premennými

$$S_1(y_1,y_2,z) = y_1 \geq 0 \ \& \ y_2 \geq 0 \ \& \ (y_1 \neq 0 \ \vee \ y_2 \neq 0) \ \& z = max\{u: \ u|y_1 \ \& \ u|y_2\}$$

Konštrukcia invariantu k danému deliacemu bodu programu je nerozhodnuteľným problémom. Niektoré zo známych heuristík, o ktoré sa môžeme pri návrhu invariantu opierať, uvádzame v časti 8.1.

5.3 Konštrukcia verifikačných podmienok

Sptná substitúcia – $R^{new}_{AB}(\overline{x},\overline{y}) =??$ $\quad r^{new}_{AB}(\overline{x},\overline{y}) =??$
cesta programom (príkaz)
$R^{old}_{AB}(\overline{x},\overline{y}) = true$ $\quad r^{old}_{AB}(\overline{x},\overline{y}) = \overline{y}$

Princíp konštrukcie – indukcia vzhľadom na sptný prechod cestou: ak za príkazom $R_{AB}(\overline{x},\overline{y})$ a $r_{AB}(\overline{x},\overline{y})$, potom pred ním

- prázdny príkaz: (príkaz skoku)

  $$R_{AB}(\overline{x},\overline{y}) \qquad r_{AB}(\overline{x},\overline{y})$$

- priradenie $\overline{y} := \overline{t}(\overline{x},\overline{y})$:

  $$R_{AB}(\overline{x},\overline{t}(\overline{x},\overline{y})) \qquad r_{AB}(\overline{x},\overline{t}(\overline{x},\overline{y}))$$

- T–vetva podmienky $p(\overline{x},\overline{y})$:

  $$R_{AB}(\overline{x},\overline{y}) \ \& \ p(\overline{x},\overline{y}) \qquad r_{AB}(\overline{x},\overline{y})$$

- F–vetva podmienky $p(\overline{x},\overline{y})$:

  $$R_{AB}(\overline{x},\overline{y}) \ \& \ \neg p(\overline{x},\overline{y}) \qquad r_{AB}(\overline{x},\overline{y})$$

Príklad: konštrukcie R_{AB} a r_{AB}

R: $y_2 \leq x$ **r:** $[y_1+1, y_2+y_3, y_3+2]$

$p^F: y_2 > x$

R: $true$ **r:** $[y_1+1, y_2+y_3, y_3+2]$

$[y_1,y_2,y_3] := [y_1+1, y_2+y_3, y_3+2]$

R: $true$ **r:** $[y_1,y_2,y_3]$

1. Príklad dôkazu správnosti programu: Uvažujme program P, ktorý pre $x \geq 0$ počíta $\lfloor\sqrt{x}\rfloor$. Výstupná podmienka sa dá vyjadriť formulou $z^2 \leq x < (z+1)^2$.

P: **begin** $[y_1,y_2,y_3] := [0,0,1]$
1: $[y_2] := [y_2 + y_3]$
2: **if** $y_2 > x$ **then goto end**
3: $[y_1,y_3] := [y_1+1, y_3+2]$
4: **goto** 1
end $[z] := [y_1]$

Invarianty –

$I_B(x)$: $\quad x \geq 0$
$I_2(x,y_1,y_2,y_3)$: $\quad (y_1^2 \leq x \ \& \ y_2 = (y_1+1)^2 \ \& \ y_3 = 2y_1+1)$
$I_E(x,z)$: $\quad z^2 \leq x < (z+1)^2$

Verifikačné podmienky –

- cesta B2:
$$\forall x\ [I_B(x)\ \&\ true \Rightarrow I_2(x,0,1,1)]$$
- cesta 22:
$$\forall x, y_1, y_2, y_3\ [I_2(x,y_1,y_2,y_3)\ \&\ y_2 \le x \Rightarrow I_2(x, y_1+1, y_2+y_3+2, y_3+2)]$$
- cesta 2E:
$$\forall x, y_1, y_2, y_3\ [I_2(x,y_1,y_2,y_3)\ \&\ y_2 > x \Rightarrow I_E(x,y_1)]$$

2. Príklad dôkazu správnosti schémy: Princíp dôkazu čiastočnej správnosti programov Floydovou metódou je možné zovšeobecniť aj pre (štandardné) schémy. Pre danú schému je cieľom sformulovať verifikačné podmienky zaručujúce čiastočnú správnosť ľubovoľného programu, ktorý dostaneme interpretáciou danej schémy.

Pri dôkaze čiastočnej správnosti programu P, ktorý dostaneme interpretáciou schémy S, potom už stačí iba sformulovať jednotlivé invarianty (resp. vstupné a výstupné podmienky) a dokázať interpretované verifikačné podmienky.

Uvažujme nasledujúcu programovú schému S

S: **begin** $[y_1,y_2] := [x,a]$
1: **if** $p_1(y_1,b)$ **then goto** 4
2: $[y_1,y_2] := [f_1(y_1,c), f_2(y_2,a)]$
3: **goto** 1
4: **if** $p_2(y_2,a)$ **then goto end**
5: $[y_1,y_2] := [g_1(y_1,d), g_2(y_2,a)]$
6: **goto** 1
end $[z] := [g_1(y_1,d)]$

Položíme deliace body na počiatočný príkaz, príkazy s návestiami 1 a 4 a na koncový príkaz. Invarianty pre dané body označme $I_B(x)$, $I_1(x,y_1,y_2)$, $I_4(x,y_1,y_2)$ a $I_E(x,z)$. Pre jednotlivé konečné cesty odvodíme podmienky pre prechod cesty a zodpovedajúce modifikácie premenných.

$R_{B1} : true$	$r_{B1} : [y_1,y_2] := [x,a]$
$R_{14} : p_1(y_1,b)$	$r_{14} : [y_1,y_2] := [y_1,y_2]$
$R_{41} : \neg p_2(y_2,a)$	$r_{41} : [y_1,y_2] := [g_1(y_1,d), g_2(y_2,a)]$
$R_{11} : \neg p_1(y_1,b)$	$r_{11} : [y_1,y_2] := [f_1(y_1,c), f_2(y_2,a)]$
$R_{4E} : p_2(y_2,a)$	$r_{4E} : [z] := [g_1(y_1,d)]$

Posledným krokom je sformulovanie verifikačných podmienok pre jednotlivé cesty s použitím neinterpretovaných objektov.

- cesta B1: $\forall x\ [I_B(x)\ \&\ true \Rightarrow I_1(x,x,a)]$
- cesta 14: $\forall x, y_1, y_2\ [I_1(x,y_1,y_2)\ \&\ p_1(y_1,b) \Rightarrow I_4(x,y_1,y_2)]$

- cesta 41: $\forall x, y_1, y_2\ [I_4(x,y_1,y_2)\ \&\ \neg p_2(y_2,a) \Rightarrow I_1(x, g_1(y_1,d), g_2(y_2,a))]$
- cesta 11: $\forall x, y_1, y_2\ [I_1(x,y_1,y_2)\ \&\ \neg p_1(y_1,b) \Rightarrow I_1(x, f_1(y_1,c), f_2(y_2,a))]$
- cesta 4E: $\forall x, y_1, y_2\ [I_4(x,y_1,y_2)\ \&\ p_2(y_2,a) \Rightarrow I_E(x, g_1(y_1,d))]$

5.4 Dôkaz zastavenia programu

Princíp použitý vo Floydovej metóde dokazovania čiastočnej správnosti je možné využiť aj pri dokazovaní zastavenia (ukončenia výpočtu) programu. V postupe uvedenom v časti 5.1 priradíme k deliacemu bodu funkciu, ktorá zobrazuje stav výpočtu do dobre založenej množiny. Na základe takejto funkcie vyjadríme verifikačnú podmienku konvergencie programu.

Dobre "založená" množina – množina W s čiastočným usporiadaním $\succ$, v ktorom neexistuje nekonečná klesajúca postupnosť $x_1 \succ x_2 \succ \cdots \succ x_n \succ \cdots$ prvkov z W.

Zobrazenie stavu výpočtu do dobre založenej množiny W –

- ku každému deliacemu bodu A priradíme funkciu
$$u_A : D_{\overline{x}} \times D_{\overline{y}} \mapsto W,$$
- ku každej konečnej ceste AB postavíme verifikačnú podmienku konvergencie
$$\forall \overline{x}, \overline{y}\ \ p(\overline{x})\ \&\ R_{AB}(\overline{x},\overline{y}) \Rightarrow [u_A(\overline{x},\overline{y}) \succ u_B(\overline{x}, r_{AB}(\overline{x},\overline{y}))].$$

Veta 34 *Ak sú všetky podmienky ukončenia, skonštruované k programu P a vstupnej podmienke p, pravdivé potom sa program P zastaví pre všetky vstupy spajúce podmienku p.*

Dôkaz: tvrdenie vyplýva z tranzitivity implikácie (indukcia vzhľadom na cesty v programe)

Poznámka: verifikačné podmienky ukončenia (konvergencie programu) je možné zoslabiť pomocou invariantov $CI_A(\overline{x},\overline{y})$ a $CI_B(\overline{x},\overline{y})$:

$$\forall \overline{x}, \overline{y}\ \ [CI_A(\overline{x},\overline{y}) \Rightarrow u_A(\overline{x},\overline{y}) \in W]$$
$$\forall \overline{x}, \overline{y}\ \ [CI_A(\overline{x},\overline{y})\ \&\ R_{AB}(\overline{x},\overline{y})] \Rightarrow [CI_B(\overline{x}, r_{AB}(\overline{x},\overline{y}))\ \&\ u_A(\overline{x},\overline{y}) \succ u_B(\overline{x}, r_{AB}(\overline{x},\overline{y}))]$$

1. Príklad dôkazu zastavenia programu: Uvažujme známy program P, ktorý pre $x \ge 0$ počíta $\lfloor \sqrt{x} \rfloor$.

P: **begin** $[y_1,y_2,y_3] := [0,0,1]$
1: $[y_2] := [y_2 + y_3]$
2: **if** $y_2 > x$ **then goto end**
3: $[y_1,y_3] := [y_1+1, y_3+2]$
4: **goto** 1
end $[z] := [y_1]$

Invarianty –

$$
\begin{array}{ll}
CI_B(x): & x \geq 0 \\
CI_1(x, y_1, y_2, y_3): & (y_2 \leq x \ \& \ y_3 > 0)
\end{array}
$$

Zobrazenie –

$$
u_1(x, y_1, y_2, y_3): \qquad x - y_2
$$

Verifikačné podmienky ukončenia –

- $\forall x \ [CI_B(x) \ \& \ true \Rightarrow CI_1(x, 0, 0, 1)]$
- $\forall x, \overline{y} \ [CI_1(x, \overline{y}) \ \& \ y_2 + y_3 \leq x \Rightarrow CI_1(x, y_1 + 1, y_2 + y_3, y_3 + 2)]$
  $\forall x, \overline{y} \ [CI_1(x, \overline{y}) \ \& \ y_2 + y_3 \leq x] \ \Rightarrow \ [u_1(x, \overline{y}) \succ u_1(x, y_1 + 1, y_2 + y_3, y_3 + 2)]$
- $\forall x, \overline{y} \ [CI_1(x, \overline{y}) \ \Rightarrow \ u_1(x, \overline{y}) \in N]$

Kapitola 6

Hoareova metóda

Hoareova metóda pre dokazovanie čiastočnej správnosti štruktúrovaných programov sa opiera o vytvorenie špecializovaného logického (inferenčného) systému nad induktívnymi formulami. Ak odvodíme v takomto systéme induktívnu formulu $\{p\} \ P \ \{q\}$, štruktúrovaný program P je čiastočne správny vzhľadom na vstupnú podmienku p a výstupnú podmienku q.

6.1 Logický systém

Na úvod stručne uvádzame základné pojmy z teórie dôkazov.

Formálny jazyk – dobre vytvorené formuly α, β.

Sémantika – platnosť (pravdivosť) formúl: $\models \alpha$;
formula α platí v danej triede modelov resp. v danom konkrétnom modeli.

Logický (inferenčný, dokazovací) systém – dokazateľnosť formúl: $\vdash \alpha$;
formula α sa dá dokázať (odvodiť) z *axióm* a *inferenčných pravidiel* systému

$$
\frac{\alpha_1 \cdots \alpha_n}{\beta}.
$$

Zdravé inferenčné pravidlo – z platných formúl α_i (t.j. $\models \alpha_i$ pre všetky i) inferuje platný dôsledok, $\models \beta$.

Dôkaz – formula α je *dokazateľná* (t.j. $\vdash \alpha$) ak existuje postupnosť formúl $\alpha_1, \alpha_2, \ldots, \alpha_n$ (*formálny dôkaz* formuly α) taká, že

- $\alpha \equiv \alpha_n$
- pre každé i, $(1 \leq i \leq n)$ buď α_i je axióma alebo existuje pravidlo

$$
\frac{\alpha_{j_1}, \ldots, \alpha_{j_k}}{\alpha_i}
$$

  také, že všetky α_{j_m} $(1 \leq m \leq k)$ sú členmi postupnosti $\alpha_1, \ldots, \alpha_{i-1}$.

Vlastnosti systému –

zdravý systém – dokazateľné sú len pravdivé formuly

$$\vdash \alpha \quad \Rightarrow \quad \models \alpha,$$

úplný systém – každá pravdivá formula je dokazateľná

$$\models \alpha \quad \Rightarrow \quad \vdash \alpha.$$

6.2 Induktívna (invariantná) formula

Induktívne formuly charakterizujú vstupno–výstupné vzťahy programových segmentov (príkazov, programov).

Induktívnou formulou $\{p\}\, S\, \{q\}$ sa popisuje vstupno–výstupná charakterizácia programového segmentu S s tým, že podmienka p je vstupným invariantom a podmienka q je výstupným invariantom S. Význam induktívnej formuly závisí od významu podmienok p, q a významu programu S.

Definícia 35 *Induktívna formula $\{p\}\, S\, \{q\}$ platí (označenie $\models \{p\}\, S\, \{q\}$) práve vtedy, keď vstupné hodnoty programu (príkazu) S spajú podmienku p a po zastavení programu S výstupné hodnoty S vyhovujú podmienke q.*

Induktívna formula $\{p\}\, S\, \{q\}$ teda platí, keď je programový segment S čiastočne správny vzhľadom na vstupnú podmienku p a výstupnú podmienku q. Analogicky môžeme definovať aj induktívnu formulu pre totálnu správnosť $\langle p\rangle\ S\ \langle q\rangle$.

Definícia 36 *Induktívna formula $\langle p\rangle\ S\ \langle q\rangle$ platí (označenie $\models\ \langle p\rangle\ S\ \langle q\rangle$) práve vtedy, keď pre všetky vstupné hodnoty programu (príkazu) S, ktoré spajú podmienku p sa program S zastaví a výstupné hodnoty S vyhovujú podmienke q.*

V tejto prednáške sa zameriame hlavne na induktívne formuly "pre čiastočnú" správnosť.

Príklad: Uvažujme jednoduché induktívne formuly:

- $\{x = 0\}\, x := x + 1\, \{x = 1\}$ – je platná formula,
- $\{x = 0\}\, x := x + 1\, \{x = 2\}$ – nie je platná formula,
- $\{x = 0\}\, x := x + 1\, \{x > 0\}$ – je platná formula, ale ...

Tretia formula evidentne platí, ale oproti prvej je výstupná podmienka (vzhľadom na vstupnú a programový segment – príkaz priradenia) príliš voľná (slabá), alebo naopak vstupná je (vzhľadom na výstupnú) príliš reštriktívna (silná).

1. Najslabšia vstupná podmienka: Najslabšia vstupná podmienka k programu P a výstupnej podmienke q – označenie $wp(P, q)$:

$$\forall p \ \text{ ak platí } \ \{wp(P,q)\}\, P\, \{q\},\ \{p\}\, P\, \{q\} \ \text{ potom } \ p \Longrightarrow wp(P,q).$$

Nasledujúce formuly vyjadrujú najslabšiu vstupnú podmienku pre zakladné štruktúrované konštrukcie:

- *priradenie* – $wp(x := s, q) = q[x/s]$,
- *kompozícia* – $wp(S_1; S_2, q) = wp(S_1, wp(S_2, q))$,
- *vetvenie* – $wp(\textbf{if}\ b\ \textbf{then}\ S_1\ \textbf{else}\ S_2\ \textbf{fi}, q) = \text{if } b \text{ then } wp(S_1, q) \text{ else } wp(S_2, q) \text{ fi}$,
- *iterácia* – $wp(\textbf{while}\ b\ \textbf{do}\ S\ \textbf{od}, q)$ nevieme vyjadriť v jazyku 1. rádu.

Zatiaľ nemáme dostatočný formálny aparát, ktorý by umožnil exaktne dokázať, že uvedené formulky skutočne vyjadrujú najslabšie vstupné podmienky pre daný príkaz a danú výstupnú podmienku. Takýto dôkaz vyžaduje formálnu definíciu významu (sémantiky) programovacieho jazyka, špecifikačného jazyka a jazyka formúl správnosti (induktívnych formúl). Vybudovanie potrebného aparátu a zdôvodnenie uvedených tvrdení je súčasťou nadstavbovej prednášky "Formálna sémantika a teória správnosti".

Pri vyjadrení najslabšej vstupnej podmienky sa predpokladá, že premenná na ľavej strane priradenia neobsahuje "vnorenú" referenciu poľa (t.j. x je alebo individuálna premenná alebo má tvar $a[t]$ a term t neobsahuje referenciu poľa a). Problémy s referenciou položiek poľa (a riešenie týchto problémov) sa podrobnejšie rozoberajú v už spomenutej nadstavbovej prednáške "Formálna sémantika a teória správnosti".

2. Najsilnejšia výstupná podmienka: Najsilnejšia výstupná podmienka k programu P a vstupnej podmienke p – označenie $sp(P, p)$:

$$\forall q \ \text{ ak platí } \ \{p\}\, P\, \{sp(P,p)\},\ \{p\}\, P\, \{q\} \ \text{ potom } \ sp(P,p) \Longrightarrow q.$$

Nasledujúce formuly vyjadrujú najsilnejšiu výstupnú podmienku pre zakladné štruktúrované konštrukcie:

- *priradenie* – $sp(x := s, p) = \exists y\ \{p[x/y]\ \&\ x = s[x/y]\}$
  ("nová" premenná y sa nevyskytuje v p a s),
- *kompozícia* – $sp(S_1; S_2, p) = sp(S_2, sp(S_1, p))$,
- *vetvenie* – $sp(\textbf{if}\ b\ \textbf{then}\ S_1\ \textbf{else}\ S_2\ \textbf{fi}, p) = sp(S_1, p\&b) \wedge sp(S_2, p\&\neg b)$,
- *iterácia* – $sp(\textbf{while}\ b\ \textbf{do}\ S\ \textbf{od}, p)$ nevieme vyjadriť.

Poznámky týkajúce sa vyjadrenia najslabšej vstupnej podmienky, uvedené v závere predchádzajúceho odseku, plne platia aj pre vyjadrenie najsilnejšej výstupnej podmienky.

6.3 Inferenčný systém $\mathcal{H}$

Uvažujme programovací jazyk štruktúrovaného typu, ktorý využíva jazykové konštrukcie definované triedou štruktúrovaných programových schém.

Hoareova metóda dokazovania čiastočnej správnosti programov (pre štruktúrované jazyky) sa opiera o logický systém pre dokazovanie čiastočnej správnosti programov založený na jazyku induktívnych formúl $\{p\}\ P\ \{q\}$.

Uvažujme formálny inferenčný systém

$$\mathcal{H} = (Ax, Pr)$$

pre dokazovanie induktívnych formúl $\{p\}\ S\ \{q\}$, pozostávajúci z množiny axióm Ax a množiny inferenčných pravidiel Pr.

Množina axióm – Ax

- všetky platné formuly špecifikačného jazyka
- axióma priradenia (schéma axióm)

$$\{p(\overline{x}, g(\overline{x}, \overline{y}))\}\ \overline{y} := g(\overline{x}, \overline{y})\ \{p(\overline{x}, \overline{y})\}$$

Inferenčné pravidlá – Pr

- *pravidlo kompozície:*

$$\frac{\{p\}\ S_1\ \{q\} \qquad \{q\}\ S_2\ \{r\}}{\{p\}\ S_1; S_2\ \{r\}}$$

- *pravidlo alternatívy:*

$$\frac{\{p\&b\}\ S_1\ \{q\} \qquad \{p\&\neg b\}\ S_2\ \{q\}}{\{p\}\ \ \mathbf{if}\ b\ \mathbf{then}\ S_1\ \mathbf{else}\ S_2\ \mathbf{fi}\ \ \{q\}}$$

- *pravidlo iterácie:*

$$\frac{\{p\&b\}\ S\ \{p\}}{\{p\}\ \ \mathbf{while}\ b\ \mathbf{do}\ S\ \mathbf{od}\ \ \{p\&\neg b\}}$$

- *pravidlo následku:*

$$\frac{p \Rightarrow p_1 \qquad q_1 \Rightarrow q \qquad \{p_1\}\ S\ \{q_1\}}{\{p\}\ S\ \{q\}}$$

Veta 37 *Nech P je štruktúrovaný program, $p(\overline{x})$ vstupný a $q(\overline{x}, \overline{z})$ výstupný predikát. Ak $\mathcal{H} \vdash \{p\}\ P\ \{q\}$ potom je program P ωifq }.*

V oboch prípadoch vychádza axióma priradenia z vyjadrenia najslabšej vstupnej podmienky priraďovacieho príkazu.

Predpoklad, podľa ktorého sú všetky platné formuly špecifikačného jazyka axiómami systému $\mathcal{H}$ umožuje "prekonať" možný (a veľmi pravdepodobný) problém neúplnosti špecifikačného jazyka $\mathcal{L}$ (napr. predikátový kalkul prvého rádu s Peanovou aritmetikou). plnosť $\mathcal{H}$ sa tým uvažuje nezávisle od úplnosti $\mathcal{L}$ (tzv. relatívna úplnosť).

1. Príklad dôkazu čiastočnej správnosti: Uvažujme program Q, počítajúci $\lfloor\sqrt{x}\rfloor$.

$$
\begin{array}{ll}
\text{Q:} & \mathbf{begin}\ \{x \geq 0\} \\
& \quad [y_1, y_2, y_3] := [0, 1, 1]; \\
& \quad \mathbf{while}\ y_2 \leq x\ \mathbf{do}\ [y_1, y_2, y_3] := [y_1 + 1, y_2 + y_3 + 2, y_3 + 2]\ \mathbf{od}; \\
& \quad [z] := [y_1] \\
& \mathbf{end}\ \{z^2 \leq x < (z+1)^2\}
\end{array}
$$

lohou je dokázať induktívnu formulu

$$\{x \geq 0\} \quad \lfloor\sqrt{x}\rfloor \quad \{z^2 \leq x < (z+1)^2\},$$

t.j., že Q je čiastočne správny vzhľadom na vstupnú podmienku $x \geq 0$ a výstupnú podmienku $z^2 \leq x < (z+1)^2$. Ukážeme, že formula

$$R(x, y_1, y_2, y_3):\ (y_1^2 \leq x)\ \&\ (y_2 = (y_1+1)^2)\ \&\ (y_3 = 2y_1 + 1)$$

je invariantom, postačujúcim na dôkaz čiastočnej správnosti programu Q.

1. $x \geq 0 \ \Rightarrow\ R(x, 0, 1, 1)$ (vyplýva z definície invariantu)
2. $\{R(x, 0, 1, 1)\}\ [y_1, y_2, y_3] := [0, 1, 1]\ \{R(x, y_1, y_2, y_3)\}$ (axióma priradenia)
3. $\{x \geq 0\}\ [y_1, y_2, y_3] := [0, 1, 1]\ \{R(x, y_1, y_2, y_3)\}$ (pravidlo následku pre 1. a 2.)
4. $R(x, y_1, y_2, y_3)\ \&\ y_2 \leq x \ \Rightarrow\ R(x, y_1 + 1, y_2 + y_3 + 2, y_3 + 2)$ (vyplýva z definície invariantu)
5. $\{R(x, y_1 + 1, y_2 + y_3 + 2, y_3 + 2)\}$
$[y_1, y_2, y_3] := [y_1 + 1, y_2 + y_3 + 2, y_3 + 2]$
$\{R(x, y_1, y_2, y_3)\}$ (axióma priradenia)
6. $\{R(x, y_1, y_2, y_3)\ \&\ y_2 \leq x\}$
$[y_1, y_2, y_3] := [y_1 + 1, y_2 + y_3 + 2, y_3 + 2]$
$\{R(x, y_1, y_2, y_3)\}$ (pravidlo následku pre 4. a 5.)
7. $\{R(x, y_1, y_2, y_3)\}$
$\mathbf{while}\ y_2 \leq x\ \mathbf{do}\ [y_1, y_2, y_3] := [y_1 + 1, y_2 + y_3 + 2, y_3 + 2]\ \mathbf{od}$
$\{R(x, y_1, y_2, y_3)\ \&\ y_2 > x\}$ (pravidlo iterácie pre 6.)

8. $\{x \geq 0\}$
$[y_1, y_2, y_3] := [0, 1, 1];$
while $y_2 \leq x$ **do** $[y_1, y_2, y_3] := [y_1 + 1, y_2 + y_3 + 2, y_3 + 2]$ **od**
$\{R(x, y_1, y_2, y_3)$ & $y_2 > x\}$ (pravidlo kompozície pre 3. a 7.)

9. $R(x, y_1, y_2, y_3)$ & $y_2 > x \quad \Rightarrow \quad y_1^2 \leq x < (y_1 + 1)^2$ (vyplýva z definície invariantu)

10. $\{y_1^2 \leq x < (y_1 + 1)^2\}\ [z] := [y_1]\ \{z^2 \leq x < (z + 1)^2\}$ (axióma priradenia)

11. $\{R(x, y_1, y_2, y_3)$ & $y_2 > x\}\ [z] := [y_1]\ \{z^2 \leq x < (z + 1)^2\}$ (pravidlo následku pre 9. a 10.)

12. $\{x \geq 0\}$
$[y_1, y_2, y_3] := [0, 1, 1];$
while $y_2 \leq x$ **do** $[y_1, y_2, y_3] := [y_1 + 1, y_2 + y_3 + 2, y_3 + 2]$ **od** ;
$[z] := [y_1]$
$\{z^2 \leq x < (z + 1)^2\}$ (pravidlo kompozície pre 8. a 12.)

Pri Hoareovej metóde sú postačujúcimi podmienkami čiastočnej správnosti programu špecifikačné formuly, ktoré využíva pravidlo následku, t.j. v danom príklade formuly uvedené v bodoch 1,4,9.

2. Príklad dôkazu čiastočnej správnosti schém: Princíp dôkazu čiastočnej správnosti programov je možné, podobne ako pri Floydovej metóde, zovšeobecniť pre štruktúrované schémy. Pri formálnom odvodení dôkazu sa pre danú schému odvodia (verifikačné) predpoklady, ktoré zaručujú čiastočnú správnosť ľubovoľného programu, ktorý dostaneme interpretáciou danej schémy.

Pre konkrétny program P, ktorý vznikne interpretáciou schémy, potom už stačí iba sformulovať invariant a dokázať jeho vlastnosti (verifikačné predpoklady – formuly použité v dôkaze).

Uvažujme nasledujúcu štruktúrovanú programovú schému S

S: **begin** $\{p(x_1, x_2)\}$
$[y_1, y_2] := [x_1, x_2]$
while $r(y_1, y_2)$ **do**
if $s(y_1, y_2)$ **then** $[y_1] := [f(y_1, y_2)]$ **else** $[y_2] := [g(y_1, y_2)]$ **fi**
od
$[z] := [y_1]$
end $\{q(x_1, x_2, z)\}$

Invariant označme $R(x_1, x_2, y_1, y_2)$ a dôkaz čiastočnej správnosti bude vyzerať nasledovne.

1. $p(x_1, x_2) \quad \Rightarrow \quad R(x_1, x_2, x_1, x_2)$ (verifikačný predpoklad)

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

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

4. $R(x_1, x_2, y_1, y_2)$ & $s(y_1, y_2) \quad \Rightarrow \quad R(x_1, x_2, f(y_1, y_2), y_2)$ (verifikačný predpoklad)

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

6. $\{R(x_1, x_2, y_1, y_2)$ & $s(y_1, y_2)\}\ [y_1] := [f(y_1, y_2)]\ \{R(x_1, x_2, y_1, y_2)\}$
(pravidlo následku pre 4. a 5.)

7. $R(x_1, x_2, y_1, y_2)$ & $\neg s(y_1, y_2) \quad \Rightarrow \quad R(x_1, x_2, y_1, g(y_1, y_2))$ (verifikačný predpoklad)

8. $\{R(x_1, x_2, y_1, g(y_1, y_2))\}\ [y_2] := [g(y_1, y_2)]\ \{R(x_1, x_2, y_1, y_2)\}$ (axióma priradenia)

9. $\{R(x_1, x_2, y_1, y_2)$ & $\neg s(y_1, y_2))\}\ [y_2] := [g(y_1, y_2)]\ \{R(x_1, x_2, y_1, y_2)\}$
(pravidlo následku pre 7. a 8.)

10. $\{R(x_1, x_2, y_1, y_2))\}$
if $s(y_1, y_2)$ **then** $[y_1] := [f(y_1, y_2)]$ **else** $[y_2] := [g(y_1, y_2)]$
$\{R(x_1, x_2, y_1, y_2)\}$ (pravidlo alternatívy pre 6. a 9.)

11. $R(x_1, x_2, y_1, y_2)$ & $r(y_1, y_2) \quad \Rightarrow \quad R(x_1, x_2, y_1, y_2)$ (tautológia)

12. $\{R(x_1, x_2, y_1, y_2)$ & $r(y_1, y_2)\}$
if $s(y_1, y_2)$ **then** $[y_1] := [f(y_1, y_2)]$ **else** $[y_2] := [g(y_1, y_2)]$
$\{R(x_1, x_2, y_1, y_2)\}$ (pravidlo následku pre 10. a 11.)

13. $\{R(x_1, x_2, y_1, y_2)\}$
while $r(y_1, y_2)$ **do**
if $s(y_1, y_2)$ **then** $[y_1] := [f(y_1, y_2)]$ **else** $[y_2] := [g(y_1, y_2)]$ **fi**
od
$\{R(x_1, x_2, y_1, y_2)$ & $\neg r(y_1, y_2)\}$ (pravidlo iterácie pre 12.)

14. $\{p(x_1, x_2)\}$
$[y_1, y_2] := [x_1, x_2]$
while $r(y_1, y_2)$ **do**
if $s(y_1, y_2)$ **then** $[y_1] := [f(y_1, y_2)]$ **else** $[y_2] := [g(y_1, y_2)]$ **fi**
od
$\{R(x_1, x_2, y_1, y_2)$ & $\neg r(y_1, y_2)\}$ (pravidlo kompozície pre 3. a 13.)

15. $R(x_1, x_2, y_1, y_2)$ & $\neg r(y_1, y_2) \quad \Rightarrow \quad q(x_1, x_2, y_1)$ (verifikačný predpoklad)

16. $\{q(x_1, x_2, y_1)\}\ [z] := [y_1]\ \{q(x_1, x_2, z)\}$ (axióma priradenia)

17. $\{R(x_1, x_2, y_1, y_2)$ & $\neg r(y_1, y_2)\}\ [z] := [y_1]\ \{q(x_1, x_2, z)\}$ (pravidlo následku pre 15. a 16.)

18. $\{p(x_1, x_2)\}$
$[y_1, y_2] := [x_1, x_2]$
while $r(y_1, y_2)$ **do**
if $s(y_1, y_2)$ **then** $[y_1] := [f(y_1, y_2)]$ **else** $[y_2] := [g(y_1, y_2)]$ **fi**
od
$[z] := [y_1]$
$\{q(x_1, x_2, z)\}$ (pravidlo kompozície pre 14. a 17.)

Pri dôkaze správnosti programu P, ktorý vznikne interpretáciou schémy S, postačuje sformulovať invariant $R(x_1, x_2, y_1, y_2)$ a dokázať verifikačné predpoklady 1, 4, 7, 15.

6.4 Poznámky o úplnosti Hoareovských kalkulov

V tejto časti upozorujeme na problémy s úplnosťou Hoareovských kalkulov, ktoré vznikajú pri danom špecifikačnom jazyku v súvislosti rozširovaním programovacieho jazyka. Podrobnejšia diskusia o tejto problematike a dôkazy úplnosti Hoareovských systémov pre vybrané triedy programovacích jazykov je súčasťou nadstavbovej prednášky "Formálna sémantika a teória správnosti".

Uvažujme špecifikačný jazyk $\mathcal{L}$, triedu modelov programov, definovanú sémantikou $\mathcal{M}$ a Hoareovský dokazovací systém $\mathcal{H}$. Príčinami neúplnosti Hoareovských inferenčných systémov sú:

- neúplnosť logiky špecifikačného jazyka $\mathcal{L}$ (napr. predikátový kalkul prvého rádu s Peanovou aritmetikou)
  riešenie: relatívna úplnosť - všetky pravdivé formuly $\mathcal{L}$ sú axiómami $\mathcal{H}$.
- neúplnosť vyvolaná nedostatočnou výrazovou silou špecifikačného jazyka $\mathcal{L}$ (napr. už keď programovací jazyk obsahuje cykly)
  riešenie: rozšírenie špecifikačného jazyka $\mathcal{L}$ – napr. definovateľnosť $wp(P, q)$.
- neúplnosť vyvolaná kombináciou rôznych programovacích konštrukcií (napr. kombinácia rekurzie, procedúry ako parametre, statického rozsahu premenných, globálnych premenných, interných procedúr)
  riešenie: vhodne definovať programovací jazyk.
- neúplnosť vyplývajúca zo zvolených modelov sémantiky (známym príkladom je inicializácia premenných s dynamickým rozsahom)
  riešenie: sémantika musí definovať adekvátnu triedu modelov.

Kapitola 7

Metóda intermitentov

Na rozdiel od Floydovej a Hoareovej metódy, ktoré boli navrhnuté pre dokazovanie čiastočnej správnosti programov (hoci sa obe dajú rozšíriť aj na dokazovanie totálnej správnosti), je cieľom metódy intermitentov dôkaz totálnej správnosti programov. Zameranie na dôkaz totálnej správnosti je inherentne zabudované v tzv. intermitentoch – podmienkách, ktoré sa pri tejto metóde viažu s deliacimi bodmi.

7.1 Intermitent

Podobne ako invarianty, aj intermitenty sú podmienky (formuly špecifikačného jazyka, napr. predikátového počtu) viazané na ľubovoľné (deliace) body programu. Rozdiel medzi obomi pojmami spočíva vo vzťahu medzi priebehom výpočtu a platnosťou podmienok viazaných k deliacim bodom výpočtu.

Intermitent je podmienka, splnená aspo pri jednom prechode deliacim bodom (zaručuje sa ou, že výpočet prejde daným deliacim bodom aspo raz).

Totálnu správnosť $\langle p \rangle$ P $\langle q \rangle$ programu P vyjadruje nasledujúca formula využívajúca vlastnosti intermitentov (vyjadrených klauzulou ak **niekedy**):

ak **niekedy** p **v** počiatočnom bode programu P
potom **niekedy** q **v** koncovom bode programu P.

7.2 Dôkaz správnosti metódou intermintentov

Metóda dokazovania (totálnej) správnosti programov pomocou intermitentov sa opiera o princíp štrukturálnej indukcie.

Myšlienku metódy predvedieme na príklade jednoduchého programu pre výpočet nsd dvoch prirodzených čísiel.

D: **begin** $[y_1, y_2] := [x_1, x_2]$
  1: **if** $y_1 = y_2$ **then goto end**
  2: **if** $y_1 > y_2$ **then** $[y_1] := [y_1 - y_2]$ **goto** 2
  3: **if** $y_2 > y_1$ **then** $[y_2] := [y_2 - y_1]$ **goto** 3
  4: **goto** 1
end $[z] := [y_2]$

Označenie:

- $\langle p \rangle\ P\ \langle q \rangle$ – ak niekedy p na začiatku P potom niekedy q na konci P.
- $\langle p \rangle_i$ – niekedy p v i.

Veta 38 $\langle x_1 > 0\ \&\ x_2 > 0 \rangle\ D\ \langle z = nsd(x_1, x_2) \rangle$.

Dôkaz:

$\langle x_1 > 0\ \&\ x_2 > 0 \rangle_{begin}$
$x_1 = x_2 \qquad \langle y_1 = x_1\ \&\ y_2 = x_2\ \&\ x_1 = x_2 > 0 \rangle_1$
$x_1 > x_2 \qquad \langle y_1 = x_1\ \&\ y_2 = x_2\ \&\ x_1 > x_2 > 0 \rangle_2$
$x_1 < x_2 \qquad \langle y_1 = x_1\ \&\ y_2 = x_2\ \&\ x_2 > x_1 > 0 \rangle_3$
$\langle z = nsd(x_1, x_2) \rangle_{end}$

Lema 39 $\langle y_1 = a\ \&\ y_2 = b\ \&\ a = b > 0 \rangle_1$
$\vee\ \langle y_1 = a\ \&\ y_2 = b\ \&\ a > b > 0 \rangle_2$
$\vee\ \langle y_1 = a\ \&\ y_2 = b\ \&\ b > a > 0 \rangle_3$
$\Rightarrow\ \langle z = nsd(a, b) \rangle_{end}$.

Dôkaz: $(N \times N, \succ) - (a, b) \succ (a', b') \quad \Leftrightarrow \quad a + b > a' + b'$

1. $\langle y_1 = a\ \&\ y_2 = b\ \&\ a = b > 0 \rangle_1$
   $\langle z = nsd(a, b) = b \rangle_{end}$
2. $\langle y_1 = a\ \&\ y_2 = b\ \&\ a > b > 0 \rangle_2$
   $\langle y_1 = a' = a - b\ \&\ y_2 = b' = b \rangle_2$
   $\{a' > 0, b' > 0, a + b > a' + b'; nsd(a', b') = nsd(a - b, b) = nsd(a, b)\}$
   $a' = b' \qquad \langle y_1 = a'\ \&\ y_2 = b'\ \&\ a' = b' > 0 \rangle_1$
   $\langle z = nsd(a', b') = nsd(a, b) \rangle_{end}$ $-$ *indukcia*
   $a' > b' \qquad \langle y_1 = a'\ \&\ y_2 = b'\ \&\ a' > b' > 0 \rangle_2$
   $\langle z = nsd(a', b') = nsd(a, b) \rangle_{end}$ $-$ *indukcia*
   $b' > a' \qquad \langle y_1 = a'\ \&\ y_2 = b'\ \&\ b' > a' > 0 \rangle_3$
   $\langle z = nsd(a', b') = nsd(a, b) \rangle_{end}$ $-$ *indukcia*
3. $\langle y_1 = a\ \&\ y_2 = b\ \&\ b > a > 0 \rangle_3$
   $\langle y_1 = a' = a\ \&\ y_2 = b' = b - a \rangle_3$
   $\{a' > 0, b' > 0, a + b > a' + b'; nsd(a', b') = nsd(a, b - a) = nsd(a, b)\}$
   $a' = b' \qquad \langle y_1 = a'\ \&\ y_2 = b'\ \&\ a' = b' > 0 \rangle_1$
   $\langle z = nsd(a', b') = nsd(a, b) \rangle_{end}$ $-$ *indukcia*
   $a' > b' \qquad \langle y_1 = a'\ \&\ y_2 = b'\ \&\ a' > b' > 0 \rangle_2$
   $\langle z = nsd(a', b') = nsd(a, b) \rangle_{end}$ $-$ *indukcia*
   $b' > a' \qquad \langle y_1 = a'\ \&\ y_2 = b'\ \&\ b' > a' > 0 \rangle_3$
   $\langle z = nsd(a', b') = nsd(a, b) \rangle_{end}$ $-$ *indukcia*

Na rozdiel od Floydovej alebo Hoareovej metódy nedá sa pri formulácii metódy intermitentov oprieť o žiadne syntaktické objekty (cesty, zložky príkazov). Použité usporiadanie odráža konkrétny výpočet, preto nie je možné určiť postupnosť krokov (kuchárku), ktoré by priamočiaro popisovali metódu dôkazu.

Kapitola 8

Konštrukcia invariantov a správnych programov

Pri dokazovaní správnosti programu (či už čiastočnej alebo totálnej) je problémom dokázať (overiť), či daný program zodpovedá danej (formálnej) špecifikácii. Pri praktickom programovaní sme však skôr postavení pred problém skonštruovať (vytvoriť, navrhnúť) program, ktorý zodpovedá daným požiadavkám, t.j. ktorý bude správny (korektný) vzhľadom na dané vstupno–výstupné špecifikácie.

V tejto kapitole ukážeme, že predvedené myšlienky dokazovania správnosti programov je možné použiť aj pri systematickej konštrukcii správnych programov. Pod systematickou konštrukciou správnych programov budeme rozumieť systematickú konštrukciu programu spolu s dôkazom jeho správnosti.

Budeme predpokladať, že formálna špecifikácia problému je daná dvojicou vstupno–výstupných podmienok. Ukážeme, že pri systematickom vývoji správnych programov je možné použiť myšlienku už známej Hoareovej metódy.

Pre úplnosť treba poznamenať, že známa a rozšírená je aj modifikácia tohoto prístupu navrhnutá E. Dijkstrom. Pri Dijkstrovej metóde sa predpokladá, že špecifikácia problému je daná (len) výstupnou podmienkou. Pri systematickom vývoji správneho programu sa konštruuje popri programe aj najslabšia vstupná podmienka k danej výstupnej podmienke a skonštruovanému programu.

8.1 Návrh invariantov

Konštrukcia správnych programov je úzko sptá s návrhom (konštrukciou) invariantov programu. Niektoré zo známych heuristík pre konštrukciu invariantov predvedieme na príklade systematickej konštrukcie programu, špecifikovaného danou vstupnou a výstupnou podmienkou:

$$p(x): \; x \geq 0 \qquad\qquad q(x,z): \; z^2 \leq x < (z+1)^2.$$

Heuristika 1: v prípade konjunkcie vo výstupnej podmienke je vhodné vziať jeden z členov konjunkcie za invariant a druhý člen použiť ako podmienku, kontrolujúcu cyklus.

Návrh invariantu: $I_1(x, y_1): \; (y_1^2 \leq x)$

Pri výbere invariantu vychádzame z predpokladaného "spôsobu spracovania". Výslednú hodnotu môžeme získať postupným zvyšovaním hodnoty pracovnej premennej tak, aby invariant zostal zachovaný.

Jednoduchou aplikáciou naznačenej heuristiky odvodíme program:

Q1: **begin** $\{x \geq 0\}$
  $[y_1] := [0]$;
  while $(y_1+1)^2 \leq x$ **do** $[y_1] := [y_1+1]$ **od** ;
  $[z] := [y_1]$
end $\{z^2 \leq x < (z+1)^2\}$

äalšia modifikácia invariantu môže vychádzať z "pokusu" o optimalizáciu programu.

Heuristika 2: výpočtovo zložitý výraz v podmienke cyklu (obsahujúci premenné modifikované telom cyklu) je vhodné nahradiť novou pracovnou premennou, pričom je potrebné zabezpečiť jej aktualizáciu v tele cyklu.

Návrh invariantu: $I_2(x, y_1, y_2): \; (y_1^2 \leq x) \;\&\; (y_2 = (y_1+1)^2)$

Na základe uvedeného optimalizačného princípu odvodíme program:

Q2: **begin** $\{x \geq 0\}$
  $[y_1, y_2] := [0, 1]$;
  while $y_2 \leq x$ **do** $[y_1, y_2] := [y_1+1, y_2+2y_1+3]$ **od** ;
  $[z] := [y_1]$
end $\{z^2 \leq x < (z+1)^2\}$

Heuristiku 2 možno využiť aj pri optimalizácii výrazov v tele cyklu.

Návrh invariantu: $I_3(x, y_1, y_2, y_3): \; (y_1^2 \leq x) \;\&\; (y_2 = (y_1+1)^2) \;\&\; (y_3 = 2y_1+1)$

Po príslušnej úprave tela cyklu dostaneme program $Q3$, o ktorom sme už skôr dokázali, že spľa danú vstupno–výstupnú charakterizáciu.

Q3: **begin** $\{x \geq 0\}$
  $[y_1, y_2, y_3] := [0, 1, 1]$;
  while $y_2 \leq x$ **do** $[y_1, y_2, y_3] := [y_1+1, y_2+y_3+2, y_3+2]$ **od**;
  $[z] := [y_1]$
end $\{z^2 \leq x < (z+1)^2\}$

8.2 Systematický vývoj korektných programov

Cieľom je skonštruovať program P, ktorý je čiastočne správny vzhľadom na danú vstupnú podmienku p a danú výstupnú podmienku q. Označme problém konštrukcie programu P, čiastočne správneho vzľadom na podmienky p, q

$$\{p\}\ P^?\ \{q\}.$$

Pri riešení tohoto problému treba vychádzať z možností daného programovacieho jazyka. V našom prípade uvažujeme jazykové konštrukcie – príkazy priradenia, kompozície, vetvenia a iterácie. Postup popísaný v nasledujúcich riadkoch sa však dá priamočiaro rozšíriť aj na ďalšie jazykové konštrukcie.

1. Vývojové kroky: Pri konštrukcii programu volí programátor konštrukciu, ktorou plánuje vyriešiť daný problém, a tým dekomponuje problém na "jednoduchšie" podproblémy. Pri formulácii podproblémov sa využívajú princípy Hoareovej metódy na dokazovanie čiastočnej správnosti programov.

Priradenie – $\{p\}\ P^?\ \{q\}$

návrh riešenia: $P \equiv x := s$

predpoklady: $p \Rightarrow q[x/s]$

redukcia na podproblémy: vyriešené

Ak platí predpoklad, programový segment (priradenie) rieši daný problém.

Kompozícia – $\{p\}\ P^?\ \{q\}$

návrh riešenia: $P \equiv P_1; P_2$

predpoklady: stanoviť "medzipodmienku" r

redukcia na podproblémy: $\{p\}\ P_1\ \{r\} \quad \{r\}\ P_2\ \{q\}$

Ak sa v ďalšom priebehu konštrukcie programu podarí stanoviť "medzipodmienku" r a vyriešiť podproblémy $\{p\}\ P_1\ \{r\}$ a $\{r\}\ P_2\ \{q\}$, výsledný program P je podľa pravidla kompozície čiastočne správny vzhľadom na p a q.

Vetvenie – $\{p\}\ P^?\ \{q\}$

návrh riešenia: $P \equiv$ **if** b **then** P_1 **else** P_2 **fi**

predpoklady: stanoviť podmienku vetvenia b

redukcia na podproblémy: $\{p\&b\}\ P_1\ \{q\} \quad \{p\&\neg b\}\ P_2\ \{q\}$

Ak sa v ďalšom priebehu konštrukcie programu podarí nájsť "vetviacu" podmienku b a vyriešiť podproblémy $\{p\&b\}\ P_1\ \{q\}$ a $\{p\&\neg b\}\ P_2\ \{q\}$, výsledný program P je podľa pravidla vetvenia čiastočne správny vzhľadom na p a q.

Iterácia – $\{p\}\ P^?\ \{q\}$

návrh riešenia: $P \equiv$ **while** b **do** P_1 **od**

predpoklady: stanoviť podmienku cyklu b a invariant r také, že $p \Rightarrow r$ a $p\&\neg b \Rightarrow q$

redukcia na podproblémy: $\{r\&b\}\ P_1\ \{r\}$

Ak sa v ďalšom priebehu konštrukcie programu podarí vyriešiť podproblémy $\{r\&b\}\ P_1\ \{r\}$ a platia uvedené vlastnosti, výsledný program P je podľa pravidla cyklu čiastočne správny vzhľadom na p a q.

Poznámka: Pri riešení problému $\{p\}\ P\ \{q\}$ môžeme plne využívať aj pravidlo následku. Ak dokážeme $\{r\}\ P\ \{s\}$, $p \Rightarrow r$ a $s \Rightarrow q$, podľa pravidla následku je program P správny aj vzhľadom na p a q.

2. Príklad systematickej konštrukcie programu: lohou je navrhnúť program, ktorý "pivotizuje" dané pole A vzhľadom na hodnotu–pivota x, ktorý sa v danom poli (zaručene) vyskytuje, a nájsť také dva indexy $j < i$, že všetky prvky naľavo od i budú menšie alebo rovné prvkom na pravej strane od j. Formálne je úloha zadaná dvojicou podmienok

- vstupná podmienka – $p \equiv \exists k\ \{1 \le k \le n\ \&\ A[k] = x\}$,
- výstupná podmienka – $q \equiv j < i\ \ \&\ \ \forall p, q\ \{1 \le p < i\ \&\ j < q \le n\ \ \Rightarrow\ \ A[p] \le A[q]\}$.

Systematická konštrukcia programu P spajúceho danú špecifikáciu pozostáva z nasledujúcich krokov:

1. *problém:* $\{p\}\ P\ \{q\}$
   návrh riešenia: $P \equiv P_1; P_2$
   $P_1 \equiv \langle$inicializuj premenné $i, j\rangle$
   $P_2 \equiv \langle$prezri pole zdola cez i, zhora cez j a rob príslušné úpravy$\rangle$
   dekompozícia na podproblémy: $\{p\}\ P_1\ \{r\} \quad \{r\}\ P_2\ \{q\}$
   heuristika: $r \equiv I_i \& I_j$
   $I_i \equiv\ \forall p\ \{1 \le p < i\ \ \Rightarrow\ \ A[p] \le x\}$
   $I_j \equiv\ \forall q\ \{j < q \le n\ \ \Rightarrow\ \ A[q] \ge x\}$
2. *problém:* $\{p\}\ P_1\ \{r\}$
   návrh riešenia: $P_1 \equiv [i, j] := [1, n]$
   dekompozícia na podproblémy: vyriešené
   treba dokázať: $p \Rightarrow I_1 \& I_n \quad (I_1 \& I_n$ triviálne platí$)$
3. *problém:* $\{r\}\ P_2\ \{q\}$
   návrh riešenia: $P_2 \equiv$ **while** b **do** P_{21} **od**
   $P_{21} \equiv \langle$zvyšuj i, znižuj j a manipuluj pole$\rangle$
   dekompozícia na podproblémy: $\{p_1 \& b\}\ P_{21}\ \{p_1\}$

heuristika: $p_1 \equiv r \quad b \equiv i \leq j$

treba dokázať: $r \Rightarrow p_1 \quad (r \Rightarrow r)$

$p_1 \& \neg b \Rightarrow q \quad (I_i \,\&\, I_j \,\&\, i > j \Rightarrow q)$

4. *problém:* $\{p_1 \& b\}\, P_{21}\, \{p_1\}$

návrh riešenia: $P_{21} \equiv P_{211}; P_{212}; P_{213}$

$P_{211} \equiv \langle$nájdi i na potenciálnu výmenu$\rangle$

$P_{212} \equiv \langle$nájdi j na potenciálnu výmenu$\rangle$

$P_{213} \equiv \langle$vyme prvky na nájdených i, j indexoch$\rangle$

dekompozícia na podproblémy: $\{p_1 \& b\}\, P_{211}\, \{p_2\} \quad \{p_2\}\, P_{212}\, \{p_3\} \quad \{p_3\}\, P_{213}\, \{p_1\}$

heuristika: $p_2 \equiv I_i \,\&\, I_j \,\&\, A[i] \geq x \quad p_3 \equiv I_i \,\&\, I_j \,\&\, A[i] \geq x \geq A[j]$

5. *problém:* $\{p_1 \& b\}\, P_{211}\, \{p_2\}$

návrh riešenia: $P_{211} \equiv$ **while** b_1 **do** P_{2111} **od**

$P_{2111} \equiv \langle$inkrementuj $i\rangle$

dekompozícia na podproblémy: $\{p_1 \& b_1\}\, P_{2111}\, \{p_1\}$

heuristika: $b_1 \equiv A[i] < x$

treba dokázať: $p_1 \& \neg b_1 \Rightarrow p_2 \quad (I_i \,\&\, I_j \,\&\, A[i] \geq x \Rightarrow I_i \,\&\, I_j \,\&\, A[i] \geq x)$

$p_1 \& b \Rightarrow p_1 \quad (I_i \,\&\, I_j \,\&\, i \leq j \Rightarrow I_i \,\&\, I_j)$

6. *problém:* $\{p_1 \& b_1\}\, P_{2111}\, \{p_1\}$

návrh riešenia: $P_{2111} \equiv [i] := [i+1]$

dekompozícia na podproblémy: vyriešené

treba dokázať: $p_1 \& b_1 \Rightarrow p_1[i/i+1] \quad (I_i \,\&\, I_j \,\&\, A[i] < x \Rightarrow I_{i+1} \,\&\, I_j)$

7. *problém:* $\{p_2 \& b\}\, P_{212}\, \{p_3\}$

návrh riešenia: $P_{212} \equiv$ **while** b_2 **do** P_{2121} **od**

$P_{2121} \equiv \langle$dekrementuj $j\rangle$

dekompozícia na podproblémy: $\{p_2 \& b_2\}\, P_{2121}\, \{p_2\}$

heuristika: $b_2 \equiv A[j] > x$

treba dokázať: $p_2 \& \neg b_2 \Rightarrow p_3 \quad (I_i \,\&\, I_j \,\&\, A[i] \geq x \,\&\, A[j] \leq x \Rightarrow I_i \,\&\, I_j \,\&\, A[i] \geq x \geq A[j])$

8. *problém:* $\{p_2 \& b_2\}\, P_{2121}\, \{p_2\}$

návrh riešenia: $P_{2121} \equiv [j] := [j-1]$

dekompozícia na podproblémy: vyriešené

treba dokázať: $p_2 \& b_2 \Rightarrow p_2[j/j-1] \quad (I_i \,\&\, I_j \,\&\, A[i] \geq x \,\&\, A[j] > x \Rightarrow I_i \,\&\, I_{j-1} \,\&\, A[i] \geq x)$

9. *problém:* $\{p_3\}\, P_{213}\, \{p_1\}$

návrh riešenia: $P_{213} \equiv$ **if** b_3 **then** P_{2131} **fi**

$P_{2131} \equiv \langle$vyme prvky a zabezpeč posuny i a $j\rangle$

dekompozícia na podproblémy: $\{p_3 \& b_3\}\, P_{2131}\, \{p_1\}$

heuristika: $b_3 \equiv b$

10. *problém:* $\{p_3 \& b\}\, P_{2131}\, \{p_1\}$

návrh riešenia: $P_{2131} \equiv P_{21311}; P_{21312}$

$P_{21311} \equiv \langle$zabezpeč výmenu i a $j\rangle$

$P_{21312} \equiv \langle$zabezpeč posuny i a $j\rangle$

dekompozícia na podproblémy: $\{p_3 \& b\}\, P_{21311}\, \{p_4\} \quad \{p_4\}\, P_{21312}\, \{p_1\}$

heuristika: $p_4 \equiv I_i \,\&\, I_j \,\&\, A[i] \leq x \leq A[j]$

11. *problém:* $\{p_3 \& b\}\, P_{21311}\, \{p_4\}$

návrh riešenia: $P_{21311} \equiv [A[i], A[j]] := [A[j], A[i]]$

dekompozícia na podproblémy: vyriešené

treba dokázať: $p_3 \& b \Rightarrow p_4[A[i]/A[j], A[j]/A[i]]$

$(I_i \,\&\, I_j \,\&\, A[i] \geq x \geq A[j] \,\&\, i \leq j \Rightarrow I_i \,\&\, I_j \,\&\, A[i] \leq x \leq A[j])$

12. *problém:* $\{p_4\}\, P_{21312}\, \{p_1\}$

návrh riešenia: $P_{21312} \equiv [i, j] := [i+1, j-1]$

dekompozícia na podproblémy: vyriešené

treba dokázať: $p_4 \Rightarrow p_1[i/i+1, j/j-1] \quad (I_i \,\&\, I_j \,\&\, A[i] \leq x \leq A[j] \Rightarrow I_{i+1} \,\&\, I_{j-1})$

Uvedeným postupom, ilustrujúcim metódu systematickej konštrukcie správnych programov, sme odvodili nasledovný program:

P: **begin** $\{\exists k\ \{1 \leq k \leq n \ \&\ A[k] = x\}\}$

$\quad [i, j] := [1, n]$

$\quad$ **while** $i \leq j$ **do**

$\qquad$ **while** $A[i] < x$ **do** $[i] := [i+1]$ **od**

$\qquad$ **while** $A[j] > x$ **do** $[j] := [j-1]$ **od**

$\qquad$ **if** $i \leq j$ **then**

$\qquad\quad [A[i], A[j]] := [A[j], A[i]]$

$\qquad\quad [i, j] := [i+1, j-1]$ **fi**

$\quad$ **od**

end $\{j < i \ \&\ \forall p, q\ \{1 \leq p < i \ \&\ j < q \leq n \ \Rightarrow\ A[p] \leq A[q]\}\}$

V priebehu konštrukcie sme zárove dokázali jeho správnosť vzhľadom na dané (vstupnú a výstupnú) podmienky.

Časť III

Sémantika programov

1. Sémantika programov a programovacích jazykov: neformálny vs. formálny popis sémantiky programov a programovacích jazykov

- neformálny prístup – význam programovacích konštrukcií a programov sa vyjadruje neformálne v prirodzenom jazyku (nejednoznačnosť prirodzeného jazyka),
- formálny prístup – význam programovacích konštrukcií a programov sa vyjadruje vo formálnom, matematickom jazyku.

2. Formálny popis významu programov: umožuje

- definovať význam programu počítačovo nezávislými pojmami (na rôznych úrovniach abstrakcie),
- jednoznačne popísať význam programu (v danom modeli resp. v triede modelov),
- odôvodniť (resp. vytvoriť) teórie a inferenčné systémy na dokazovanie vlastností programov (axiómy, zdravosť a úplnosť inferenčných pravidiel atď),
- konzistentne definovať význam typových aj beztypových programovacích jazykov atď.

3. Rozdiely medzi jazykami logiky a programovania:

- rôzny pohľad na premenné (statický vs. dynamický),
- možnosť pracovať len s konečnými aproximáciami nekonečných objektov,
- zložitejšie obory interpretácie resp. sémantické obory – domény (vnútorná štruktúra aproximácií),
- program môže realizovať čiastočnú funkciu (nekonečný výpočet) atď.

Kapitola 9

Formálna sémantika programov

Syntaktický obor $\mathcal{P}$ – množina správne vytvorených programov v danom jazyku,

Sémantický obor $\mathcal{M}$ – množina významov,

Sémantika $\mathcal{S}$ – sémantická funkcia, zobrazujúca množinu programov do množiny významov

$$\mathcal{S}: \quad \mathcal{P} \quad \mapsto \quad \mathcal{M},$$

Sémantika programu $\mathcal{S}[\![P]\!]$ – denotuje význam programu P vzhľadom na sémantiku $\mathcal{S}$.

9.1 Porovnanie sémantických definícií

Sémantika danej triedy programov $\mathcal{P}$ sa dá vyjadriť rôznym spôsobom. Rôzne sémantiky sa odlišujú výberom sémantického oboru $\mathcal{M}$.

Definícia 40 *Sémantika* $\mathcal{S}_2 : \mathcal{P} \mapsto \mathcal{M}_2$ *je* **definovateľná** *zo sémantiky* $\mathcal{S}_1 : \mathcal{P} \mapsto \mathcal{M}_1$*, ak existuje funkcia* $\mathcal{F} : \mathcal{M}_1 \mapsto \mathcal{M}_2$ *taká, že pre každý program* $P \in \mathcal{P}$ *platí*

$$\mathcal{S}_2[\![P]\!] = \mathcal{F}(\mathcal{S}_1[\![P]\!]).$$

Sémantiky $\mathcal{S}_1, \mathcal{S}_2$ sú ekvivalentné ak sú navzájom definovateľné.

Definícia 41 *Sémantika* $\mathcal{S}_1 : \mathcal{P} \mapsto \mathcal{M}_1$ *je* **podrobnejšia** *ako sémantika* $\mathcal{S}_2 : \mathcal{P} \mapsto \mathcal{M}_2$ *práve vtedy, keď pre ľubovolné dva programy* $P_1, P_2 \in \mathcal{P}$ *platí*

$$\mathcal{S}_1[\![P_1]\!] = \mathcal{S}_1[\![P_2]\!] \quad \Longrightarrow \quad \mathcal{S}_2[\![P_1]\!] = \mathcal{S}_2[\![P_2]\!].$$

Duálne, sémantika $\mathcal{S}_2$ je abstraktnejšia.

Lema 42 *Nech* $\mathcal{S}_1, \mathcal{S}_2$ *sú sémantiky nad syntaktickým oborom* $\mathcal{P}$*. Potom* $\mathcal{S}_2$ *je definovateľná z* $\mathcal{S}_1$ *práve vtedy, keď* $\mathcal{S}_1$ *je podrobnejšia ako* $\mathcal{S}_2$*.*

Dôkaz: Ak $\mathcal{S}_2$ je definovateľná z $\mathcal{S}_1$, potom tvrdenie vyplýva priamo z definície.
Predpokladajme teda, že $\mathcal{S}_1$ je podrobnejšia ako $\mathcal{S}_2$:

- $\mathcal{P}/\mathcal{S}_1$ – rozklad $\mathcal{P}$ vzhľadom na $\mathcal{S}_1$,
- $F_2 : \mathcal{P}/\mathcal{S}_1 \mapsto \mathcal{M}_2$ – predpis $X \mapsto \mathcal{S}_2[\![P]\!]$ pre $P \in X$,
- $F_1 : \mathcal{M}_1 \mapsto \mathcal{P}/\mathcal{S}_1$ – predpis $m \mapsto \{P_1 \in \mathcal{P} : \mathcal{S}_1[\![P_1]\!] = m\}$,
- $\mathcal{F}(\mathcal{S}_1[\![P]\!]) = F_2(F_1(\mathcal{S}_1[\![P]\!])) = \mathcal{S}_2[\![P]\!]$.

9.2 Kompozičná sémantika

Ak program P pozostáva zo zložiek $P_1, \cdots, P_n$ potom budeme požadovať aby jeho význam $\mathcal{S}[\![P]\!]$ bol funkciou významov $\mathcal{S}[\![P_1]\!], \cdots, \mathcal{S}[\![P_n]\!]$ jednotlivých zložiek.

1. Induktívna definícia syntaxe jazyka: abstraktná syntax

- elementárne syntaktické objekty (napr. priraďovací príkaz),
- množina *syntaktických konštruktorov* $K_i : \mathcal{P}^n \mapsto \mathcal{P}$,

  while_do_od: (BOOL STAT) STAT
  if_then_else_fi: (BOOL STAT STAT) STAT
  _;_: (STAT STAT) STAT
- ľubovoľný "zložený" program P v danom jazyku má tvar $P = K(P_1, \cdots, P_n)$.

2. Induktívna definícia sémantiky jazyka:

- charakterizuje sa význam elementárnych syntaktických objektov (priraďovací príkaz),
- ku každému syntaktickému konštruktoru K_i sa definuje zodpovedajúci *sémantický konštruktor* $\mathcal{K}_i : \mathcal{M}^n \mapsto \mathcal{M}$,
- sémantika každého zloženého príkazu $P = K(P_1, \cdots, P_n)$ sa definuje *sémantickým pravidlom*

$$\mathcal{S}[\![K(P_1, \cdots, P_n)]\!] = \mathcal{K}(\mathcal{S}[\![P_1]\!], \cdots, \mathcal{S}[\![P_n]\!]).$$

9.3 Základné prístupy pri popise významu programov

Základné prístupy pri popise významu programov

- operačná (výpočtová) sémantika,
- denotačná (matematická) sémantika,
- deduktívna (axiomatická) sémantika

sa odlišujú výberom sémantického oboru $\mathcal{M}$ v definícii $\mathcal{S} : \mathcal{P} \mapsto \mathcal{M}$.

Pri neformálnej charakterizácii základných prístupov budeme využívať nasledujúcu vlastnosť priestorov funkcií.

Lema 43 $D_1 \times D_2 \mapsto D_3 \cong D_1 \mapsto (D_2 \mapsto D_3)$

Intuitívna predstava: zafixovaním jedného parametra binárnej funkcie dostaneme unárnu funkciu.

1. Operačná sémantika:

Abstraktný počítač – je definovaný napr.

- množinou stavov (konfigurácií) $S = \{s, s_i, \cdots\}$,
- množinou elementárnych inštrukcií $I = \{i, i_k, \cdots\}$,
- prechodovou funkciou $q : I \times S \mapsto S$.

Sémantický obor – trieda zobrazení množiny stavov S do množiny výpočtových postupností S^∞, t.j. postupností stavov (konfigurácií) $s_0, s_1, \cdots, s_n, \cdots$ generovaných abstraktným počítačom v priebehu výpočtu začínajúcom vstupným stavom (konfiguráciou) $s_0 \in S$.

Význam programu – zobrazenie vstupného stavu do zodpovedajúcej výpočtovej postupnosti.

Sémantická funkcia –

$$\mathcal{O}(P, s_0) = s_0, s_1, \cdots, s_n \qquad (\cdots)$$
$$\mathcal{O} : \mathcal{P} \times S \mapsto S^\infty$$
$$\mathcal{O} : \mathcal{P} \mapsto (S \mapsto S^\infty)$$

Sémantický popis jazyka –

- transformácie stavu, realizované elementárnymi príkazmi jazyka,
- pravidlá, podľa ktorých sa z postupnosti stavov (konfigurácií), zodpovedajúcich zložkám príkazov jazyka, vytvoria postupnosti stavov (konfigurácií), zodpovedajúce príkazom jazyka.

2. Denotačná sémantika:

Sémantický obor – priestor funkcií resp. relácií nad množinou stavov.

Význam programu – vstupno–výstupný vzťah vyjadrený funkciou resp. reláciou.

Sémantická funkcia –

$$\mathcal{M}(P, s_0) = s_n \qquad \text{(alebo } \perp)$$
$$\mathcal{M} : \mathcal{P} \times S \mapsto S$$
$$\mathcal{M} : \mathcal{P} \mapsto (S \mapsto S)$$

Sémantický popis jazyka –

- elementárne príkazy vyjadríme ako funkciu (reláciu) definovanú nad oborom stavov,
- pre každý syntaktický konštruktor ukážeme, ako sa z funkcií (relácií), zodpovedajúcich zložkám príkazu skonštruuje funkcia (relácia) vyjadrujúca význam príkazu.

Intuitívna vlastnosť: $\mathcal{M}$ je definovateľná z $\mathcal{O}$.

3. Deduktívna sémantika:

Špecifikačný jazyk –

- jazyk $L = \{p, q, \cdots\}$ na popis vstupno–výstupných podmienok,
- najslabšia vstupná podmienka $wp(P, q)$ k výstupnej podmienke q a programu P s vlastnosťou: ak je splnená $wp(P, q)$ a program P sa skončí, musí byť splnená výstupná podmienka q.

Sémantický obor – priestor transformátorov predikátov $L \mapsto L$.

Význam programu – transformátor predikátov, definujúci napr. $wp(P, q)$ k danej výstupnej podmienke q.

Sémantická funkcia –

$$\mathcal{D}(P, p) = q \qquad \text{(napr. } q = wp(P, p))$$
$$\mathcal{D} : \mathcal{P} \times L \mapsto L$$
$$\mathcal{D} : \mathcal{P} \mapsto (L \mapsto L)$$

Sémantický popis jazyka –

- elementárne príkazy sa charakterizujú transformátormi predikátov,
- pre každý syntaktický konštruktor ukážeme, ako sa z transformátorov predikátov, zodpovedajúcich zložkám príkazu skonštruuje transformátor predikátov vyjadrujúci význam príkazu.

Intuitívna vlastnosť: $\mathcal{D}$ je definovateľná z $\mathcal{M}$.

9.4 Ekvivalencia programov

Ekvivalencia programov je sémantická vlastnosť. Pri jej definícii sa vychádza z významu programov.

Definícia 44 *Programy P_1, P_2 sú ekvivalentné vzhľadom na sémantiku $\mathcal{S}$ práve vtedy, keď platí*

$$\mathcal{S}[\![P_1]\!] = \mathcal{S}[\![P_2]\!].$$

Preto rôzne sémantické charakterizácie významu programu vedú k rôznym ekvivalenciám.

1. Operačná ekvivalencia: $\mathcal{O}[\![P_1]\!] = \mathcal{O}[\![P_2]\!]$.
Pre každý vstup realizujú programy P_1, P_2 na danom abstraktnom počítači rovnaké výpočtové postupnosti.

2. Denotačná ekvivalencia: $\mathcal{M}[\![P_1]\!] = \mathcal{M}[\![P_2]\!]$.
Programy P_1, P_2 realizujú ten istý vstupno–výstupný vzťah (funkciu, reláciu).

3. Deduktívna ekvivalencia: $\mathcal{D}[\![P_1]\!] = \mathcal{D}[\![P_2]\!]$.
Programy P_1, P_2 definujú ekvivalentné transformátory predikátov, ktoré zobrazujú ľubovoľnú výstupnú podmienku do ekvivalentných (najslabších) vstupných podmienok.

Kapitola 10

Algebraická štruktúra sémantických oborov

Základy teórie sémantických oborov – domén postavil začiatkom 70–tich rokov americký matematik D. Scott. Podstatou tejto teórie je štúdium algebraických štruktúr, potrebných na modelovanie nekonečných výpočtov, konečných aproximácií nekonečných objektov, integráciu formálnej sémantiky s teóriou vypočítateľnosti atď.

1. Sémantické domény: Pri konštrukcii sémantických domén sa vychádza z nasledujúcich požiadaviek:

- doména je množina objektov daného typu, čiastočne usporiadaná reláciou *aproximácie* $\sqsubseteq$,
- nekonečný objekt je reprezentovaný limitou jeho konečných aproximácií (každá postupnosť konečných aproximácií má limitu),
- existuje najmenšia aproximácia ľubovoľného objektu – $\perp$ (určuje len "príslušnosť" k doméne),
- každý prvok domény sa dá vyjadriť pomocou spočitateľnej bázy danej domény atď.

2. Charakterizácia výpočtov funkciami: Základnou požiadavkou je vyjadriť sémantiku (význam) ľubovoľného programu ako (totálnu) funkciu nad príslušnou sémantickou doménou:

- matematický aparát musí umožniť popis ľubovoľnej (každej) funkcie definovanej výpočtom (ľubovoľného) programu,
- nekonečný výpočet programu – výsledok funkcie vyjadrujúcej význam programu je nedefinovaný prvok $\perp$ (žiadna informácia),
- ak dostane funkcia na vstupe viac informácii o argumente (lepšia aproximácia) nemôže sa zmenšiť "úrove" aproximácie výsledku,
- ak je argumentom funkcie nedefinovaný prvok $\perp$ (t.j. výsledok výpočtu argumentu sa neskončí) výsledok môže byť definovaný (v niektorých prípadoch žiadame, aby bol aj výsledok nedefinovaný) atď.

3. Štruktúra sémantických oborov: Pri štúdiu sémantiky programov sa zvyčajne uvažuje jedna z troch uvedených algebraických štruktúr:

- úplné čiastočné usporiadanie – cpo (používa sa najčastejšie),
- úplné svzy (základný prístup D. Scotta pre beztypové jazyky),
- metrické priestory (moderný prístup).

10.1 plné čiastočné usporiadanie

Čiastočne usporiadaná množina – množina C, čiastočne usporiadaná usporiadaním $\sqsubseteq$, t.j. reláciou

- reflexívnou – $x \sqsubseteq x$
- antisymetrickou – ak $x \sqsubseteq y$ a $y \sqsubseteq x$ potom $x = y$
- tranzitívnou – ak $x \sqsubseteq y$ a $y \sqsubseteq z$ potom $x \sqsubseteq z$

Najmenšia horná najväčšia dolná hranica – množiny $X \subseteq C$

- $z = \sqcup X \in C$ je najmenšou hornou hranicou X ak
  - $x \sqsubseteq z$ pre všetky $x \in X$
  - $\forall y \in C$ také, že $x \sqsubseteq y$ pre všetky $x \in X$ platí $z \sqsubseteq y$
- $y = \sqcap X \in C$ je najvčšou dolnou hranicou X ak
  - $y \sqsubseteq x$ pre všetky $x \in X$
  - $\forall z \in C$ také, že $z \sqsubseteq x$ pre všetky $x \in X$ platí $z \sqsubseteq y$

Reťazec – postupnosť $\{x_i\}_0^\infty = x_0, x_1, x_2, \ldots, x_n, \ldots$ taká, že pre všetky i platí $x_i \sqsubseteq x_{i+1}$ (resp. $x_i \sqsupseteq x_{i+1}$)

plné čiastočné usporiadanie – cpo $(C, \sqsubseteq)$
čiastočné usporiadanie $(C, \sqsubseteq)$ s vlastnosťou

- existuje najmenší element vzhľadom na $\sqsubseteq$, t.j. prvok $\perp$ (dolník) taký, že $\forall x \in C \ \ \perp \sqsubseteq x$
- každý reťazec $\{x_i\}_0^\infty$ má najmenšiu hornú hranicu $\sqcup_0^\infty x_i$ v C.

Poznámka: Pre úplný zvz platia aj duálne vlastnosti (teda existencia najväčšieho elementu a najväčšej dolnej hranice pre reťazec)

10.2 Konštrukcia cpo

1. Diskrétne cpo: $(C \cup \{\perp_C\}, \sqsubseteq)$.

Nech $\perp_C \notin C$ je novým prvkom množiny C, čiastočne usporiadanej reláciou:

$$x_1 \sqsubseteq x_2 \text{ práve vtedy, keď } x_1 = \perp_C \ \vee x_1 = x_2.$$

Potom platí:

Lema 45 *$(C \cup \{\perp_C\}, \sqsubseteq)$ je cpo.*

Dôkaz: Vzhľadom na definíciu usporiadania je prvok $\perp_C$ dolníkom C. Usporiadanie pripúšťa len triviálne reťazce (obsahujú najviac dva rôzne prvky), ktoré majú vždy najmenšiu hornú hranicu.

2. Priamy súčin cpo $(C_1, \sqsubseteq_1)$ a $(C_2, \sqsubseteq_2)$: $(C_1 \times C_2, \sqsubseteq)$.

Uvažujme usporiadanie na dvojiciach $\langle x, y\rangle \in C_1 \times C_2$, definované reláciou

$$\langle x_1, y_1\rangle \sqsubseteq \langle x_2, y_2\rangle \Leftrightarrow x_1 \sqsubseteq_1 x_2 \wedge y_1 \sqsubseteq_2 y_2.$$

Potom platí:

Lema 46 *Ak $(C_1, \sqsubseteq_1)$ a $(C_2, \sqsubseteq_2)$ sú cpo potom aj $(C_1 \times C_2, \sqsubseteq)$ je cpo.*

Dôkaz: Prvok $\perp_{C_1 \times C_2} = \langle \perp_{C_1}, \perp_{C_2}\rangle$ je zrejme dolníkom $C_1 \times C_2$ vzhľadom na dané usporiadanie. K ukončeniu dôkazu stačí položiť $\sqcup_0^\infty \langle x_i, y_i\rangle = \langle \sqcup_0^\infty x_i, \sqcup_0^\infty y_i\rangle$.

Cvičenie: Ak $(C_1, \sqsubseteq_1)$ a $(C_2, \sqsubseteq_2)$ sú diskrétne cpo, potom $(C_1 \times C_2, \sqsubseteq)$ nemusí byť diskrétnym cpo.

3. Zjednotenie disjunktných cpo $(C_1, \sqsubseteq_1)$ a $(C_2, \sqsubseteq_2)$: $(C_1 \uplus C_2 \uplus \perp, \sqsubseteq)$.

Uvažujme množinu $C = C_1 \uplus C_2 \uplus \{\perp\}$ takú, že $\perp \notin C_1 \uplus C_2$ a usporiadanie $x \sqsubseteq y$ práve vtedy, keď $x, y \in C_1$ a $x \sqsubseteq_1 y$ alebo $x, y \in C_2$ a $x \sqsubseteq_2 y$ alebo $x = \perp$ a $y \in C_1 \uplus C_2$. Potom platí:

Lema 47 *Ak $(C_1, \sqsubseteq_1)$ a $(C_2, \sqsubseteq_2)$ sú cpo potom aj $(C, \sqsubseteq)$ je cpo.*

Dôkaz: Priamočiare overenie vlastností cpo.

4. Funkcie z cpo $(C_1, \sqsubseteq_1)$ do $(C_2, \sqsubseteq_2)$: $(C_1 \mapsto C_2, \sqsubseteq)$.

Uvažujme usporiadanie na funkciách z $C_1 \mapsto C_2$, definované reláciou

$$f \sqsubseteq g \text{ práve vtedy, keď } \forall x \in C_1 \text{ platí } f(x) \sqsubseteq_2 g(x).$$

Lema 48 *Ak $(C_1, \sqsubseteq_1)$ a $(C_2, \sqsubseteq_2)$ sú cpo potom aj $(C_1 \mapsto C_2, \sqsubseteq)$ je cpo.*

Dôkaz: Ľahko sa dá overiť, že všade nedefinovaná funkcia $\perp_f \in C_1 \mapsto C_2$, definovaná v λ-notácii vzťahom $\perp_f = \lambda x.\ \perp_{C_2}$, t.j. $\forall x \in C_1 \ \perp_f(x) = \perp_{C_2}$ je dolníkom priestoru funkcií.
K ľubovoľnému reťazcu $\{f_i\}_0^\infty$ definujme $f = \sqcup_0^\infty f_i$ predpisom $f(x) = \sqcup_0^\infty f_i(x)$. Potom

- pretože pre každé i a x platí $f_i(x) \sqsubseteq_2 \sqcup_0^\infty f_i(x)$, platí aj $f_i \sqsubseteq f$,
- nech $f_i \sqsubseteq g$ pre všetky i; pre každé $x \in C$ a i platí $f_i(x) \sqsubseteq_{C_2} \sqcup_0^\infty f_i(x) \sqsubseteq_{C_2} g(x)$ a teda $f \sqsubseteq g$.

5. Monotónne funkcie z cpo $(C_1, \sqsubseteq_1)$ do $(C_2, \sqsubseteq_2)$: $(C_1 \mapsto_m C_2, \sqsubseteq)$.

Funkcia $f \in C_1 \mapsto C_2$ je **monotónna** ak pre všetky $x, y \in C_1$ platí $x \sqsubseteq_1 y \Rightarrow f(x) \sqsubseteq_2 f(y)$. Množinu všetkých monotónnych funkcií budeme označovať $C_1 \mapsto_m C_2$. Uvažujme to isté čiastočné usporiadanie na funkciách ako v prípade cpo $(C_1 \mapsto C_2, \sqsubseteq)$.

Lema 49 *Ak $(C_1, \sqsubseteq_1)$, $(C_2, \sqsubseteq_2)$ sú cpo potom aj $(C_1 \mapsto_m C_2, \sqsubseteq)$ je cpo.*

Dôkaz: Keďže $\perp_f$ je evidentne monotónna funkcia je dolníkom $C_1 \mapsto_m C_2$.
Ukážeme, že $\sqcup$ reťazca monotónnych funkcií je monotónna funkcia. Ak pre $x_1, x_2 \in C_1$ platí $x_1 \sqsubseteq_1 x_2$, potom $f_i(x_1) \sqsubseteq_2 f_i(x_2)$ pre všetky i. Odtiaľ $f(x_1) = \sqcup_0^\infty f_i(x_1) \sqsubseteq_2 \sqcup_0^\infty f_i(x_2) = f(x_2)$ a teda $f(x_1) \sqsubseteq_2 f(x_2)$.

Cvičenie: Ak pre $\forall i$ platí, že $f_i(x_1) \sqsubseteq_2 f_i(x_2)$, potom $\sqcup_0^\infty f_i(x_1) \sqsubseteq_2 \sqcup_0^\infty f_i(x_2)$.

6. Striktné funkcie z cpo $(C_1, \sqsubseteq_1)$ do $(C_2, \sqsubseteq_2)$: $(C_1 \mapsto_s C_2, \sqsubseteq)$.

Funkcia $f \in C_1 \mapsto C_2$ je **striktná** ak $f(\perp_{C_1}) = \perp_{C_2}$. Množinu striktných funkcií budeme označovať $C_1 \mapsto_s C_2$. Uvažujme opť to isté čiastočné usporiadanie na funkciách ako v prípade cpo $(C_1 \mapsto C_2, \sqsubseteq)$.

Lema 50 *Ak $(C_1, \sqsubseteq_1)$, $(C_2, \sqsubseteq_2)$ sú cpo potom aj $(C_1 \mapsto_s C_2, \sqsubseteq)$ je cpo.*

Dôkaz: Všade nedefinovaná funkcia $\perp_f$ je zrejme striktná. Je teda aj dolníkom cpo striktných funkcií.
Pre každý reťazec $\{f_i\}_0^\infty$ striktných funkcií je aj $f = \sqcup_0^\infty f_i$ striktná funkcia ($\forall i:\ f_i(\perp_{C_1}) = \perp_{C_2}$ a teda aj $f(\perp_{C_1}) = \perp_{C_2}$).

Lema 51 *Ak C_1 je diskrétne cpo, potom $C_1 \mapsto_s C_2 \ \subseteq\ C_1 \mapsto_m C_2$.*

Dôkaz: Nech $x_1 \sqsubseteq_1 x_2$. Potom buď $x_1 = \perp_{C_1}$ a teda $f(x_1) = f(\perp_{C_1}) = \perp_{C_2} \sqsubseteq_2 f(x_2)$, alebo $x_1 = x_2$ a teda aj $f(x_1) = f(x_2)$.

Kapitola 11

Sémantika imperatívnych programov

1. Syntax jednoduchého imperatívneho jazyka:

- syntaktické obory – výrazy, príkazy,
- syntax programovacieho jazyka.

2. Algebraické základy sémantiky:

- algebraická štruktúra sémantických oborov – cpo,
- sémantické obory – stav pamti,
- sémantické funkcie,
- "metajazyk" pre popis sémantiky.

3. Operačná sémantika:

- príkaz cyklu umožuje nekonečné výpočty (vtedy nie je definovaná výstupná hodnota),
- vstupno–výstupná charakterizácia operačnej sémantiky.

4. Denotačná sémantika:

- sémantika príkazu cyklu – čiastočne definovaná funkcia (keď sa výpočet neskončí, hodnota nie je definovaná),
- rozšírenie čiastočných funkcií $\Sigma \mapsto \Sigma$ na totálne $(\Sigma \cup \perp) \mapsto (\Sigma \cup \perp)$,
- úlohu nedefinovaného stavu (reprezentujúceho nekonečný výpočet) bude hrať nový stav – "dolník" $\perp \notin \Sigma$,
- výsledkom výpočtu, začínajúceho nedefinovaným stavom nemôže byť "dobre definovaný" stav,
- sémantiku cyklu $\mathcal{M}[\![$**while** b **do** S **od**$]\!]\sigma$ popíšeme postupnosťou funkcií – aproximujúcich k "otočení" cyklu.

5. Porovnanie operačnej a denotačnej sémantiky:

- operačná vstupno–výstupná sémantika a denotačná sémantika sú navzájom ekvivalentné.

11.1 Syntax jazyka

Na príklade jednoduchého imperatívneho programovacieho jazyka ukážeme základné princípy formálneho prístupu k popisu sémantiky programov (a programovacích jazykov). S modelovaním ďalších prvkov a čt imperatívnych jazykov sa zoznámime v nadstavbovej prednáške "Formálna sémantika a teória správnosti".

Predpokladáme, že analyzovaný jazyk umožuje

- (implicitne) deklarovať individuálne premenné jediného typu integer (t.j. nemajú žiadnu štruktúru), jednoznačne identifikujúce pamťové miesta,
- používať riadiace konštrukcie: priradenie, kompozícia príkazov, príkaz vetvenia a iteratívny príkaz **while**.

1. Symboly: $Ivar, Icon, Bcon$

celočíselné premenné – $Ivar =_{nt} \{x, y, z, \ldots\}$,

celočíselné konštanty – $Icon =_{nt} \{m, n, \ldots\}$,

booleovské konštanty – $Bcon = \{$ **true** , **false** $\}$.

2. Syntaktické obory: Jednotlivé syntaktické obory $Iexp, Bexp, Stat$ sú definované jednak pomocou BNF– notácie a dvak formou tzv. abstraktnej syntaxe (bez syntaktického cukru).

celočíselné výrazy – $Iexp =_{nt} \{s, s_i, \ldots\}$,

$$s ::= \; x \mid m \mid s_1 + s_2 \mid \cdots \mid \textbf{if } b \textbf{ then } s_1 \textbf{ else } s_2 \textbf{ fi},$$

$$Iexp \; = \; Ivar \uplus Icon \uplus Iexp \times Iexp \uplus \cdots \uplus Bexp \times Iexp \times Iexp.$$

booleovské výrazy – $Bexp =_{nt} \{b, b_i, \ldots\}$,

$$b ::= \; \textbf{true} \mid \textbf{false} \mid s_1 \textbf{ eq } s_2 \mid \cdots \mid \textbf{not } b \mid b_1 \textbf{ and } b_2 \mid \cdots,$$

$$Bexp \; = \; Bcon \uplus Iexp \times Iexp \uplus \cdots \uplus Bexp \uplus Bexp \times Bexp \uplus \cdots.$$

príkazy – $Stat =_{nt} \{S, S_i, \ldots\}$,

$$S ::= \; x := s \mid S_1; S_2 \mid \textbf{if } b \textbf{ then } S_1 \textbf{ else } S_2 \textbf{ fi} \mid \textbf{while } b \textbf{ do } S \textbf{ od} \mid \textbf{skip} ,$$

$$Stat \; = \; Ivar \times Iexp \uplus Stat \times Stat \uplus Bexp \times Stat \times Stat \uplus Bexp \times Stat \uplus \{ \textbf{ skip } \}.$$

program – príkaz.

Syntaktická identita – $\equiv$ $(\equiv_{Iexp}, \equiv_{Bexp}, \equiv_{Stat})$
dve konštrukcie sú syntakticky identické ak pozostávajú z rovnakých postupností symbolov

11.2 Sémantické obory

Prvoradým predpokladom štúdia sémantiky formálnych jazykov je uvedomiť si rozdiel medzi syntaktickými a sémantickými obormi, t.j. medzi syntaktickými a sémantickými objektami.

1. Množina celých čísiel: $V =_{nt} \{\alpha, \alpha_i, \ldots\}$,

- aritmetické funkcie na celých číslach, napr. $\_ +_V \_ : V \times V \mapsto V$,

2. Množina pravdivostných hodnôt: $W = \{tt, ff\} = T$,

- relácie (podmnožiny $V \times V$), napr. $\_ =_W \_ : V \times V \mapsto W$,
- booleovské funkcie, napr. $\neg_W \_ : W \mapsto W$; $\_ \wedge_W \_ : W \times W \mapsto W$,
- $if\_then\_else\_fi : W \times C \times C \mapsto C$ pre ľubovoľnú množinu C

$$if\ \beta\ then\ c_1\ else\ c_2\ fi = \begin{cases} c_1 & \text{ak } \beta = tt \\ c_2 & \text{ak } \beta = ff \end{cases}$$

3. Množina stavov: $\Sigma =_{df} Ivar \mapsto V =_{nt} \{\sigma, \ldots\}$,

- stav σ – charakterizácia stavu pamti,
- $\sigma(x)$ – okamžitá hodnota premennej x v stave σ,
- variant stavu $\sigma\{x/\alpha\} : \Sigma \times Ivar \times V \mapsto \Sigma$
  - $\sigma\{x/\alpha\}(y) = \alpha$ ak $x \equiv y$,
  - $\sigma\{x/\alpha\}(y) = \sigma(y)$ ak $x \not\equiv y$.

Lema 52 *Pre všetky $\sigma \in \Sigma$, $\alpha_1, \alpha_2 \in V$ a $x, y \in Ivar$ také, že $x \not\equiv y$ platí:*

- $\sigma\{x/\sigma(x)\} = \sigma$
- $\sigma\{x/\alpha_1\}\{x/\alpha_2\} = \sigma\{x/\alpha_2\}$
- $\sigma\{x/\alpha_1\}\{y/\alpha_2\} = \sigma\{y/\alpha_2\}\{x/\alpha_1\}$

Dôkaz: priamo z vlastností variantu pre všetky $z \in Ivar$. V prvých dvoch bodoch stačí dokazovať pre dve možnosti ($x \equiv z$, $x \not\equiv z$) a v poslednom pre tri ($z \not\equiv x \not\equiv y$, $z \not\equiv x \equiv y$, $z \equiv x \not\equiv y$).

Cvičenie: Dokážte tvrdenia lemy.

4. Sémantické obory: Objekty množín V, W, Σ sú z hľadiska vzájomnej aproximácie navzájom neporovnateľné (reprezentujú úplnú informáciu o danom objekte). Prirodzeným predpokladom je uvažovať ich rozšírenie na diskrétne cpo:

- pravdivostné hodnoty $W_\perp = W \cup \{\perp_W\}$,
- celé čísla $V_\perp = V \cup \{\perp_V\}$,
- stavy $\Sigma_\perp = \Sigma \cup \{\perp_\Sigma\}$
  - $\perp_\Sigma\{x/\alpha\} = \perp_\Sigma$.

Rozšírenie sémantických domén o $\perp$ treba premietnúť aj do definície "sémantickej" funkcie if–then–else–fi, t.j. pre ľubovoľné cpo C, položme

$$if\ \beta\ then\ c_1\ else\ c_2\ fi = \begin{cases} c_1 & \text{ak } \beta = tt \\ c_2 & \text{ak } \beta = ff \\ \perp_C & \text{ak } \beta = \perp_W \end{cases}.$$

Evidentne, Lema 52 platí aj pre rozšírenú množinu stavov $\Sigma_\perp$.

11.3 Sémantika výrazov

Pri definovaní sémantiky výrazov sú jednotlivým syntaktickým objektom priradené sémantické objekty, teda je definovaný vzájomný vzťah $Bexp$ a W, resp. $Iexp$ a V. Sémantická funkcia $\mathcal{V}$ (resp. $\mathcal{W}$) priraďuje význam celočíselným (booleovským) výrazom v závislosti od konkrétneho stavu výpočtu $\Sigma_\perp$. Obe funkcie sú definované súbežnou indukciou vzhľadom na konštrukciu termov.

Symboly funkcií v syntaxi jazyka (+, **and**, **if–then–else–fi** atď.) striktne odlišujeme od ich sémantických protajškov ($+_V$, $\wedge_W$, if–then–else–fi atď.).

1. Význam celočíselných výrazov: $\mathcal{V} : Iexp \mapsto (\Sigma_\perp \mapsto_s V_\perp), \quad \sigma \in \Sigma$

- $\mathcal{V}[\![x]\!]\sigma =_{df} \sigma(x)$
- $\mathcal{V}[\![m]\!]\sigma =_{df} \alpha_m$
- $\mathcal{V}[\![s_1 + s_2]\!]\sigma =_{df} \mathcal{V}[\![s_1]\!]\sigma \ +_V \ \mathcal{V}[\![s_2]\!]\sigma$
- ...
- $\mathcal{V}[\![\textbf{if}\ b\ \textbf{then}\ s_1\ \textbf{else}\ s_2\ \textbf{fi}]\!]\sigma =_{df}$ if $\mathcal{W}[\![b]\!]\sigma$ then $\mathcal{V}[\![s_1]\!]\sigma$ else $\mathcal{V}[\![s_2]\!]\sigma$ fi
- $\mathcal{V}[\![s]\!] \perp_\Sigma =_{df} \perp_V$

2. Význam booleovských výrazov: $\mathcal{W} : Bexp \mapsto (\Sigma_\perp \mapsto_s W_\perp), \quad \sigma \in \Sigma$

- $\mathcal{W}[\![\ \textbf{true}\ ]\!]\sigma =_{df} tt$
- $\mathcal{W}[\![\ \textbf{false}\ ]\!]\sigma =_{df} ff$
- $\mathcal{W}[\![s_1\ \textbf{eq}\ s_2]\!]\sigma =_{df} \mathcal{V}[\![s_1]\!]\sigma =_W \mathcal{V}[\![s_2]\!]\sigma$
- ...
- $\mathcal{W}[\![\ \textbf{not}\ b]\!]\sigma =_{df} \neg_W \mathcal{W}[\![b]\!]\sigma$
- $\mathcal{W}[\![b_1\ \textbf{and}\ b_2]\!]\sigma =_{df} \mathcal{W}[\![b_1]\!]\sigma \ \wedge_W \ \mathcal{W}[\![b_2]\!]\sigma$
- ...
- $\mathcal{W}[\![b]\!] \perp_\Sigma =_{df} \perp_W$

Príklad: Nech $\sigma(x) = 3, \sigma(y) = 4$, potom

$$\mathcal{V}[\![\textbf{if}\ x\ \textbf{eq}\ y\ \textbf{then}\ x + z\ \textbf{else}\ y + 1\ \textbf{fi}]\!]\sigma = 5.$$

Dôkaz:
$\mathcal{V}[\![\textbf{if}\ x\ \textbf{eq}\ y\ \textbf{then}\ x + z\ \textbf{else}\ y + 1\ \textbf{fi}]\!]\sigma =$
if $\mathcal{W}[\![x\ \textbf{eq}\ y]\!]\sigma$ then $\mathcal{V}[\![x + z]\!]\sigma$ else $\mathcal{V}[\![y + 1]\!]\sigma$ fi =
$\mathcal{V}[\![y + 1]\!]\sigma = \mathcal{V}[\![y]\!]\sigma + \mathcal{V}[\![1]\!]\sigma = \sigma(y) + 1 = 4 + 1 = 5$

keď $\mathcal{W}[\![x\ \textbf{eq}\ y]\!]\sigma = \mathcal{V}[\![x]\!]\sigma =_W \mathcal{V}[\![y]\!]\sigma = \sigma(x) =_W \sigma(y) = 3 =_W 4 = ff$

Cvičenie: Dokážte, že v stave σ, definovanom v predchádzajúcom príklade platí

$$\mathcal{W}[\![\ \textbf{true}\ \ \textbf{and}\ \ \textbf{not}\ (x\ \textbf{eq}\ y)]\!]\sigma = tt.$$

11.4 Operačná vstupno–výstupná sémantika

Vo všeobecnosti sa operačnou sémantikou popisuje program ako funkcia, ktorá počiatočnému stavu výpočtu priradí postupnosť stavov popisujucich výpočet.

$$\mathcal{O} : Stat \mapsto (\Sigma_\perp \mapsto_s \Sigma_\perp^\infty).$$

Cieľom tejto kapitoly je popísať operačnú i výpočtovú sémantiku jazyka a porovnať obe definície. Z tohoto dôvodu použijeme v tejto časti tzv. vstupno–výstupnú charakterizáciu operačnej sémantiky

$$\mathcal{O}_v : Stat \mapsto (\Sigma_\perp \mapsto_s \Sigma_\perp).$$

Sémantická funkcia $\mathcal{O}_v$ síce popisuje vstupno–výstupný vzťah, ale je definovaná na striktne operačných základoch (výpočtových postupnostiach). Zásadný rozdiel medzi vstupno–výstupnou operačnou sémantikou a denotačnou sémantikou (ktorá je vždy charakterizovaná vstupno–výstupným vzťahom) je v tom, že denotačná sémantika je vyjadrená matematickým aparátom bez operačného vplyvu.

Pre zjednodušenie zápisu budeme sémantickú funkciu, definujúcu vstupno–výstupnú operačnú sémantiku označovať $\mathcal{O}$ (namiesto $\mathcal{O}_v$).

- $\mathcal{O}[\![x := s]\!]\sigma =_{df} \sigma\{x/\mathcal{V}[\![s]\!]\sigma\}$
- $\mathcal{O}[\![S_1; S_2]\!]\sigma =_{df} \mathcal{O}[\![S_2]\!]\sigma'$, kde $\sigma' = \mathcal{O}[\![S_1]\!]\sigma$
- $\mathcal{O}[\![$**if** b **then** S_1 **else** S_2 **fi**$]\!]\sigma =_{df}$ $if\ \mathcal{W}[\![b]\!]\sigma\ then\ \mathcal{O}[\![S_1]\!]\sigma\ else\ \mathcal{O}[\![S_2]\!]\sigma\ fi$
- $\mathcal{O}[\![$**while** b **do** S **od**$]\!]\sigma =_{df}$
  - $\sigma' \in \Sigma$, ak existuje $n \geq 0$ a $\sigma_0, \ldots, \sigma_n$ také, že $\sigma_i \in \Sigma$, $\sigma = \sigma_0$, $\sigma' = \sigma_n$ a $\sigma_i = \mathcal{O}[\![S]\!]\sigma_{i-1}$ (pre $i = 1, 2, \ldots, n$), $\mathcal{W}[\![b]\!]\sigma_i = tt$ (pre $i = 0, 1, \ldots, n-1$) a $\mathcal{W}[\![b]\!]\sigma_n = ff$
  - $\perp_\Sigma$, inak

Poznámka: Sémantika iterácie má zreteľne operačný charakter (využíva výpočtovú postupnosť). Stav $\mathcal{O}[\![$**while** b **do** S **od**$]\!]\sigma = \perp_\Sigma$ môže nastať v dvoch prípadoch. Ak je podmienka, kontrolujúca cyklus vždy splnená, teda $\forall i\ \mathcal{W}[\![b]\!](\mathcal{O}[\![S^i]\!]\sigma) = tt$, resp. v prípade nekonečného výpočtu pri vyhodnocovaní podmienky b, teda $\exists i$, že $\sigma_i = \perp_\Sigma$ a $\mathcal{W}[\![b]\!]\sigma_i = \perp_W$.

Cvičenie: Definujte "štandardnú" (nie vstupno–výstupnú) operačnú sémantiku daného jazyka.

Lema 53 $\forall S \in Stat\quad \mathcal{O}[\![S]\!] \perp_\Sigma = \perp_\Sigma$.

Dôkaz: indukciou vzhľadom na štruktúru S:

- $S \equiv x := s$: $\mathcal{O}[\![S]\!] \perp_\Sigma = \perp_\Sigma \{x/\mathcal{V}[\![s]\!] \perp_\Sigma\} = \perp_\Sigma$,
- $S \equiv S_1; S_2$: $\mathcal{O}[\![S_1; S_2]\!] \perp_\Sigma = \mathcal{O}[\![S_2]\!](\mathcal{O}[\![S_1]\!] \perp_\Sigma) =_{ind.hyp.} \mathcal{O}[\![S_2]\!] \perp_\Sigma =_{ind.hyp.} \perp_\Sigma$,
- $S \equiv$ **if** b **then** S_1 **else** S_2 **fi** : $\mathcal{O}[\![S]\!] \perp_\Sigma = if\ \mathcal{W}[\![b]\!] \perp_\Sigma\ then \cdots fi = if\ \perp_W\ then \cdots fi = \perp_\Sigma$.

11.5 Denotačná sémantika

$$\mathcal{M} : Stat \mapsto (\Sigma_\perp \mapsto_s \Sigma_\perp)$$

- $\mathcal{M}[\![x := s]\!] =_{df} \lambda\sigma.\ \sigma\{x/\mathcal{V}[\![s]\!]\sigma\}$
- $\mathcal{M}[\![S_1; S_2]\!] =_{df} \lambda\sigma.\ \mathcal{M}[\![S_2]\!](\mathcal{M}[\![S_1]\!]\sigma)$
- $\mathcal{M}[\![$**if** b **then** S_1 **else** S_2 **fi**$]\!] =_{df} \lambda\sigma.\ if\ \mathcal{W}[\![b]\!]\sigma\ then\ \mathcal{M}[\![S_1]\!]\sigma\ else\ \mathcal{M}[\![S_2]\!]\sigma\ fi$
- $\mathcal{M}[\![$**while** b **do** S **od**$]\!] =_{df} \sqcup_0^\infty \phi_i$, kde
  - $\phi_0 = \lambda\sigma.\ \perp_\Sigma$
  - $\phi_{i+1} = \lambda\sigma.\ if\ \mathcal{W}[\![b]\!]\sigma\ then\ \phi_i(\mathcal{M}[\![S]\!]\sigma)\ else\ \sigma\ fi$, pre $i \geq 0$

Poznámka: Postupnosť aproximácií ϕ_i vyjadruje, čo sa dá vypočítať s i otočeniami cyklu (ak je na výpočet zo stavu σ potrebných viac opakovaní, $\phi_i(\sigma) = \perp_\Sigma$).

Príklad: Pre každý stav $\sigma \in \Sigma$ platí
$\mathcal{M}[\![x := 0;\ y := x + 1]\!]\sigma = \mathcal{M}[\![y := x + 1]\!](\mathcal{M}[\![x := 0]\!]\sigma) = \mathcal{M}[\![y := x + 1]\!]\sigma\{x/\mathcal{V}[\![0]\!]\sigma\} =$
$\mathcal{M}[\![y := x + 1]\!]\sigma\{x/0\} = \sigma\{x/0\}\{y/\mathcal{V}[\![x + 1]\!]\sigma\{x/0\}\} =$
$\sigma\{x/0\}\{y/(\mathcal{V}[\![x]\!]\sigma\{x/0\} + \mathcal{V}[\![1]\!]\sigma\{x/0\})\} = \sigma\{x/0\}\{y/(\sigma\{x/0\}(x) + 1)\} =$
$\sigma\{x/0\}\{y/(0 + 1)\} = \sigma\{x/0\}\{y/1\}$.

Cvičenie: Ukážte, že pre $z \not\equiv x$, $z \not\equiv y$, $x \not\equiv y$ a ľubovolný stav $\sigma \in \Sigma$ platí

$$\mathcal{M}[\![z := x;\ x := y;\ y := z]\!]\sigma\{x/2\}\{y/3\} = \sigma\{x/3\}\{y/2\}\{z/2\}.$$

Lema 54 $\forall S \in Stat\quad \mathcal{M}[\![S]\!] \in \Sigma_\perp \mapsto_s \Sigma_\perp$.

Dôkaz: indukciou vzhľadom na i dokážeme, že $\{\phi_i\}_0^\infty$ je reťazcom striktných funkcií. Prípad $\phi_0 \sqsubseteq \phi_1$ vyplýva priamo z definície ϕ_0. Pri dôkaze $\phi_{i+1} \sqsubseteq \phi_{i+2}$ treba uvažovať pre ľubovoľné σ' tri prípady $\beta = \mathcal{W}[\![b]\!]\sigma'$:

- $\beta = \perp_W$: zrejme $\phi_{i+1}(\sigma') = \perp_\Sigma = \phi_{i+2}(\sigma')$,
- $\beta = tt$: z indukčnej hypotézy $\phi_i(\mathcal{M}[\![S]\!]\sigma') \sqsubseteq \phi_{i+1}(\mathcal{M}[\![S]\!]\sigma')$,
- $\beta = ff$: zrejme $\phi_{i+1}(\sigma') = \sigma' = \phi_{i+2}(\sigma')$.

Keďže ϕ_i sú striktné funkcie, $\sqcup_0^\infty \phi_i$ existuje a $\mathcal{M}[\![S]\!]$ je striktná funkcia, pretože $\Sigma_\perp \mapsto_s \Sigma_\perp$ je cpo.

Príklady:

1. $\mathcal{M}[\![$**while true do** S **od**$]\!] = \sqcup_0^\infty \phi_i$
   - $\phi_0 = \lambda\sigma.\ \perp_\Sigma$
   - $\phi_{i+1} = \lambda\sigma.\ if\ \mathcal{W}[\![\mathbf{true}]\!]\sigma\ then\ \phi_i(\mathcal{M}[\![S]\!]\sigma)\ else\ \sigma\ fi = \lambda\sigma.\ \perp_\Sigma$
2. $\mathcal{M}[\![$**while false do** S **od**$]\!] = \sqcup_0^\infty \phi_i$
   - $\phi_0 = \lambda\sigma.\ \perp_\Sigma$
   - $\phi_{i+1} = \lambda\sigma.\ if\ \mathcal{W}[\![\mathbf{false}]\!]\sigma\ then\ \phi_i(\mathcal{M}[\![S]\!]\sigma)\ else\ \sigma\ fi = \lambda\sigma.\ \sigma$
3. $\mathcal{M}[\![$**while** $x > 0$ **do** $x := x - 1$ **od**$]\!]\sigma\{x/2\} = \sigma\{x/0\}$ pre $\sigma \in \Sigma$:

   $$\phi_i = \lambda\sigma.\ if\ \mathcal{W}[\![x > 0]\!]\sigma\ then\ \phi_{i-1}(\mathcal{M}[\![x := x - 1]\!]\sigma)\ else\ \sigma\ fi$$

   - $\phi_0(\sigma\{x/2\}) = \perp_\Sigma$
   - $\phi_1(\sigma\{x/2\}) = \phi_0(\sigma\{x/1\}) = \perp_\Sigma$
   - $\phi_2(\sigma\{x/2\}) = \phi_1(\sigma\{x/1\}) = \phi_0(\sigma\{x/0\}) = \perp_\Sigma$
   - $\phi_3(\sigma\{x/2\}) = \phi_2(\sigma\{x/1\}) = \phi_1(\sigma\{x/0\}) = \sigma\{x/0\}$
   - $\phi_i(\sigma\{x/2\}) = \sigma\{x/0\}$ pre $i > 3$.

   Takže $(\sqcup_0^\infty \phi_i)(\sigma\{x/2\}) = \sqcup_0^\infty \phi_i(\sigma\{x/2\}) = \sqcup_0^\infty \sigma\{x/0\} = \sigma\{x/0\}$

11.6 Porovnanie operačnej a denotačnej sémantiky

Lema 55 *Nech* $S^0 =$ **skip** *a* $S^i = S^{i-1}; S$. *Potom*
$\mathcal{O}[\![$**while** b **do** S **od**$]\!]\sigma =$

- $\sigma' \in \Sigma$, *ak* $\exists n: \; n \geq 0$ *také, že* $\sigma' = \mathcal{O}[\![S^n]\!]\sigma$, $\mathcal{W}[\![b]\!](\mathcal{O}[\![S^m]\!]\sigma) = tt$ *pre* $m = 0, 1, \cdots, n-1$ *a* $\mathcal{W}[\![b]\!](\mathcal{O}[\![S^n]\!]\sigma) = ff$,
- $\perp_\Sigma$ *v opačnom prípade.*

Dôkaz: priamo z definície $\mathcal{O}$ – na základe operačného popisu významu kompozície sa eliminujú medzistavy.

Lema 56 *Pre všetky* $\sigma, \sigma' \in \Sigma$ *a* $i \geq 0$ *platí* $\sigma' = \phi_i(\sigma)$ *práve vtedy, keď* $\exists j : 0 \leq j < i$ *také, že (pre* $k = 0, 1, \cdots, j-1$*)*

- $\sigma' = \mathcal{M}[\![S^j]\!]\sigma$,
- $\mathcal{W}[\![b]\!](\mathcal{M}[\![S^k]\!]\sigma) = tt$ *a* $\mathcal{W}[\![b]\!](\mathcal{M}[\![S^j]\!]\sigma) = ff$.

Dôkaz: indukciou vzhľadom na i (intuícia!). Lema oznamuje existenciu najmenšej aproximácie ϕ_j, ktorá pre vstup σ vypočíta výsledok $\phi_j(\sigma) \neq \perp_\Sigma$. Ak $\phi_i(\sigma) \neq \perp_\Sigma$, k ľubovolnému vstupnému stavu $\sigma \in \Sigma$ dostaneme $\sigma' \neq \perp_\Sigma$, ak sa dá cyklus vypočítať na menej iteračných krokov ako i. Pre ϕ_j sa j-krát testuje podmienka, ale telo cyklu sa vykoná $j-1$-krát.

Veta 57 $\forall S \in Stat \;\; \mathcal{O}[\![S]\!] = \mathcal{M}[\![S]\!]$.

Dôkaz: indukciou vzhľadom na štruktúru S. Ak S nie je príkaz cyklu, výsledok vyplýva priamo z definície. Pre $S \equiv$ **while** b **do** S_1 **od** uvažujme dva prípady:

Prípad $\mathcal{O}[\![S]\!]\sigma = \sigma' \;\Rightarrow\; \mathcal{M}[\![S]\!]\sigma = \sigma'$:

- $\sigma' \in \Sigma$:
  Podľa lemy 55 $\exists n \geq 0 : \; \sigma' = \mathcal{O}[\![S_1^n]\!]\sigma$, (pre $m = 0, 1, \ldots, n-1$) $\mathcal{W}[\![b]\!](\mathcal{O}[\![S_1^m]\!]\sigma) = tt$ a $\mathcal{W}[\![b]\!](\mathcal{O}[\![S_1^n]\!]\sigma) = ff$. Keďže na základe indukčnej hypotézy platí $\mathcal{O}[\![S_1]\!] = \mathcal{M}[\![S_1]\!]$, $\exists n \geq 0 : \; \sigma' = \mathcal{M}[\![S_1^n]\!]\sigma$, $\mathcal{W}[\![b]\!](\mathcal{M}[\![S_1^m]\!]\sigma) = tt$ (pre $m = 0, 1, \ldots, n-1$) a $\mathcal{W}[\![b]\!](\mathcal{M}[\![S_1^n]\!]\sigma) = ff$. Uvažujme reťazec funkcií

  - $\phi_0 = \lambda\sigma. \perp_\Sigma$
  - $\phi_{i+1} = \lambda\sigma.\, if \; \mathcal{W}[\![b]\!]\sigma \; then \; \phi_i(\mathcal{M}[\![S_1]\!]\sigma) \; else \; \sigma \; fi$ pre $i \geq 0$.

  Keďže z lemy 56 vyplýva $\sigma' = \phi_i(\sigma)$ pre $i \geq n+1$, dostaneme: $\sigma' = \sqcup_0^\infty \phi_i(\sigma) = (\sqcup_0^\infty \phi_i)\sigma = \mathcal{M}[\![S]\!]\sigma$.

- $\sigma' = \perp_\Sigma$:
  Potom musí platiť $\forall k : \; \phi_k(\sigma) = \perp_\Sigma$ a teda aj $\mathcal{M}[\![S]\!]\sigma = \perp_\Sigma$. Predpokladajme opak, t.j. že existuje j také, že $\phi_j(\sigma) = \sigma' \in \Sigma$. Potom podľa lemy 56, indukčnej hypotézy a lemy 55 $\exists n \geq 0 : \; \sigma'' = \mathcal{O}[\![S_1^n]\!]\sigma$, (pre $m = 0, 1, \ldots, n-1$) $\mathcal{W}[\![b]\!](\mathcal{O}[\![S_1^m]\!]\sigma) = tt$ a $\mathcal{W}[\![b]\!](\mathcal{O}[\![S_1^n]\!]\sigma) = ff$. Takže $\mathcal{O}[\![S]\!]\sigma = \sigma' \in \Sigma$, čo je v spore s predpokladom.

Prípad $\mathcal{M}[\![S]\!]\sigma = \sigma' \;\Rightarrow\; \mathcal{O}[\![S]\!]\sigma = \sigma'$:

Predpokladajme, že $\mathcal{O}[\![S]\!]\sigma = \sigma'' \neq \sigma'$. Na základe predchádzajúcej úvahy aj $\mathcal{M}[\![S]\!]\sigma = \sigma''$, čo je v spore s predpokladom.

Kapitola 12

Sémantika rekurzívnych funkcionálnych programov

1. Rekurzívny program: konštrukcia, známa z funkcionálneho programovania – pozostáva zo systému rekurzívnych definícií, napr.

$$P: \quad \phi(x,y) \Longleftarrow \textbf{if } x=0 \textbf{ then } 1 \textbf{ else } \phi(x-1, \phi(x,y)) \textbf{ fi}.$$

Táto konštrukcia môže byť chápaná v dvoch významoch

- ako rovnica (deklaratívno–denotačný význam),
- ako nahrádzanie ľavej strany pravou (operačno–výpočtový význam).

2. Denotačná sémantika: najmenší pevný bod rekurzívnej definície

- funkcia f je riešením rovnice $\phi(\overline{x}) = t[\phi](\overline{x})$ ak $f(\overline{x}) \equiv t[f](\overline{x})$ v silnej ekvivalencii,
- jednoznačné riešenie f_τ – najmenší pevný bod $t[\phi]$,
- pre zjednodušenie budeme uvažovať funkcie z $D^n \mapsto D$ – nie je to na úkor všeobecnosti, výsledky platia aj pre "všeobecnejšie" funkcie,
- riešenie príkladu – $f_\tau(x,y):$ if $x \geq 0$ then 1 else $\perp$ fi.

3. Operačná sémantika: výpočet prepisovaním termov

- deterministický charakter výpočtu – výpočtové pravidlá,
- "volanie hodnotou" – $\underline{\phi(1,0)} \to_v \phi(0, \underline{\phi(1,0)}) \to_v \cdots \to_v^*$,
- "volanie menom" – $\underline{\phi(1,0)} \to_n \underline{\phi(0,\phi(1,0))} \to_n 1$.

4. Vzťah operačnej a denotačnej sémantiky:

- funkcia, vypočítaná programom P s pravidlom r – c_P^r,
- $c_P^v(x,y)$: if $x = 0$ then 1 else $\perp$ fi,
- $c_P^n(x,y)$: if $x \geq 0$ then 1 else $\perp$ fi ,
- korektné výpočtové pravidlo r – $f_\tau \equiv c_P^r$,
- "volanie hodnotou" nie je korektné pravidlo.

12.1 Syntaktické obory

Symboly –

- premenné – $X =_{nt} \{x, y, x_i, \ldots\}$
- symboly funkcií $F = \bigcup_{i=0}^{\infty} F^i$: $\quad f_F \in F^n$ – $arity(f) = n$
- predikátové symboly $B = \bigcup_{i=0}^{\infty} B^i$: $\quad p \in B^n$ – $arity(p) = n$
- funkčné premenné $\Phi = \bigcup_{i=0}^{\infty} \Phi^i$: $\quad \phi \in \Phi^n$ – $arity(\phi) = n$

Termy – $Exp = \{t, t_i, \cdots\}$ resp. $Bexp = \{b, b_i, \cdots\}$

- $X \subseteq Exp$, $F^0 \subseteq Exp$, $B^0 \subseteq Bexp$
- ak $f_F \in F^n$, $t_1, \cdots, t_n \in Exp$, potom $f_F(t_1, \cdots, t_n) \in Exp$
- ak $p \in B^n$, $t_1, \cdots, t_n \in Exp$, potom $p(t_1, \cdots, t_n) \in Bexp$
- ak $b \in Bexp$, $t_1, t_2 \in Exp$, potom **if** b **then** t_1 **else** t_2 **fi** $\in Exp$
- ak $\phi \in \Phi^n$, $t_1, \cdots, t_n \in Exp$, potom $\phi(t_1, \cdots, t_n) \in Exp$

Poznámka: Označením f_F zvýrazujeme, že symbol je syntaktický objekt.

Rekurzívna definícia – $\phi(x_1, \cdots, x_n) \Longleftarrow t[\phi](x_1, \cdots, x_n)$

Program – "telo" programu $t_0[\phi_1, \cdots, \phi_n](\overline{x})$ so systémom rekurzívnych definícií

$$\begin{aligned}
\phi_1(\overline{x}) &\Longleftarrow t_1[\phi_1, \cdots, \phi_n](\overline{x}) \\
\phi_2(\overline{x}) &\Longleftarrow t_2[\phi_1, \cdots, \phi_n](\overline{x}) \\
&\vdots \\
\phi_n(\overline{x}) &\Longleftarrow t_n[\phi_1, \cdots, \phi_n](\overline{x})
\end{aligned}$$

12.2 Obory interpretácie a sémantické obory

1. Interpretácia symbolov:

- obor interpretácie D_0,
- symboly funkcií z F^n sú interpretované totálnymi funkciami z $D_0^n \mapsto D_0$,
- obor interpretácie $W_0 = \{tt, ff\}$,
- predikátové symboly z B^n sú interpretované totálnymi funkciami (predikátmi) z $D_0^n \mapsto W_0$.

2. Sémantické obory:

Diskrétne cpo –

- $D = D_0 \uplus \{\perp\}$ $\quad$ ($D \equiv D_\perp$ z predchádzajúcej kapitoly),
- $W = W_0 \uplus \{\perp\}$ $\quad$ ($W \equiv W_\perp$ z predchádzajúcej kapitoly),
- silná ekvivalencia – $x \equiv y$ práve vtedy, keď $x \sqsubseteq y$ a zárove $y \sqsubseteq x$ $\quad$ (dôsledok: $\perp\equiv\perp$).

Priestor (monotónnych) funkcií – $\{f, f_i, \cdots\}$

$$D^n \mapsto D.$$

Označenie – $\Omega =_{df} \perp_{(D^n \mapsto D)}$.

Priestor (spojitých) funkcionálov – $\{\tau, \tau_i, \cdots\}$

$$(D^n \mapsto D) \mapsto (D^n \mapsto D).$$

V ďalšom budeme budovať teóriu len pre monotónne funkcie a spojité funkcionály.

3. Monotónne funkcie z $D^n \mapsto D$:

Monotónna funkcia – $\forall \overline{d_1}, \overline{d_2} \in D^n :$ *ak* $\overline{d_1} \sqsubseteq \overline{d_2}$, potom $f(\overline{d_1}) \sqsubseteq f(\overline{d_2})$.

- monotónnosť funkcie závisí aj od oboru – porovnaj if–then–else funkcie na $D_0 = \{ff \sqsubseteq tt\}$ a diskrétnom cpo $D = \{\perp, tt, ff\}$.

Príklady: Funkcie $f(x)$ a $g(x)$ sú monotónne, ale funkcia $h(x)$ nie je monotónna.

1. $f(x) :_m$ if $x = 0$ then 1 else x fi
2. $g(x) :_m$ if $x =\perp$ then 0 else 1 fi
3. $h(x) :_n$ if $x \equiv\perp$ then 0 else 1 fi

Silná ekvivalencia monotónnych funkcií – $f \equiv g$, t.j. $\forall \overline{x} : f(\overline{x}) \equiv g(\overline{x})$

Rozšírenie (interpretovaných) totálnych funkcií –

$$D_0^n \mapsto D_0 \text{ resp. } D_0^n \mapsto W_0 \quad \text{na} \quad D^n \mapsto D \text{ resp. } D^n \mapsto W.$$

Prirodzené rozšírenie – $g \in D^n \mapsto D$ je prirodzeným rozšírením $f \in D_0^n \mapsto D_0$ keď: $g(\overline{d}) = \perp$ práve vtedy, keď aspo jeden z argumentov g je $\perp$ a pre všetky $\overline{d} \in D_0^n$ platí $g(\overline{d}) = f(\overline{d})$.

Lema 58 *Prirodzené rozšírenie funkcie $f \in D_0^n \mapsto D_0$ je monotónnou funkciou.*

Dôkaz: sporom, t.j. predpokladáme, že $f(x_1, \cdots, x_n)$ je prirodzene rozšírená ale nie monotónna funkcia. Takže existujú $\overline{a} = \langle a_1, \cdots, a_n\rangle, \overline{b} = \langle b_1, \cdots, b_n\rangle$ tak, že $\overline{a} \sqsubseteq \overline{b}$, $\forall i :\ a_i \sqsubseteq b_i$ a $f(\overline{a}) \not\sqsubseteq f(\overline{b})$. Potom zrejme $\overline{a} \sqsubset \overline{b}$ a teda $\exists i_0 : a_{i_0} \sqsubset b_{i_0}$, čo v diskrétnom obore platí len keď $a_{i_0} = \perp$. Odtiaľ $f(\overline{a}) = \perp \sqsubseteq f(\overline{b})$, čo je v spore s predpokladom.

Opačné tvrdenie neplatí – ternárna if–then–else je monotónna, ale if tt then x else $\perp$ fi $= x$.

if tt then x else y fi $= x$
if ff then x else y fi $= y$
if $\perp$ then x else y fi $= \perp$

Slabá ekvivalencia – $x = y$, prirodzené rozšírenie funkcie $\_ = \_ : D_0^2 \mapsto W_0$ na $\_ = \_ : D^2 \mapsto W$

Cvičenie: Jednoargumentová funkcia je monotónna práve vtedy, keď je striktná (t.j. $f(\perp) = \perp$) alebo konštantná.

Cvičenie: Slabá ekvivalencia je monotónnou funkciou, ale silná ekvivalencia nie je monotónnou funkciou.

4. Spojité funkcionály:

Cpo monotónnych funkcií – $[D^n \mapsto D]$.

Podľa predchádzajúcej kapitoly existuje cpo monotónnych funkcií.

Funkcionál – τ je zobrazenie triedy funkcií $[D^n \mapsto D]$ do seba, t.j.

$$\tau \quad \in \quad [D^n \mapsto D] \mapsto [D^n \mapsto D].$$

Monotónny funkcionál – ak $f \sqsubseteq g$, potom $\tau[f] \sqsubseteq \tau[g]$.

Monotónnosť vzhľadom na viaceré výskyty ϕ – $\tau[\phi, \phi]$

ak $f \sqsubseteq g$ potom pre každé h platí $\tau[f, h] \sqsubseteq \tau[g, h]$ a $\tau[h, f] \sqsubseteq \tau[h, g]$.

Poznámka: Funkcionál je monotónny práve vtedy, keď je monotónny vzhľadom na všetky výskyty ϕ.

Spojitý funkcionál – pre každý reťazec $\{f_i\}_0^\infty : \tau[\sqcup_0^\infty f_i] \equiv \sqcup_0^\infty \tau[f_i]$;

definícia je korektná – obe najmenšie horné hranice existujú.

Príklady: spojitých resp. nespojitých funkcionálov:

1. $\tau[\phi]:\ \phi$

   Funkcionál je spojitý, lebo $\tau[\sqcup_0^\infty f_i] \equiv \sqcup_0^\infty f_i \equiv \sqcup_0^\infty \tau(f_i)$.

2. $\tau[\phi]:\ h$

   Funkcionál je spojitý, lebo $\tau[\sqcup_0^\infty f_i] \equiv h \equiv \sqcup_0^\infty \tau(f_i)$.

3. $\tau[\phi](x):$ if $x = 0$ then 1 else $\phi(x+1)$ fi

4. $\tau[\phi](x):$ if $x = 0$ then 1 else $\phi(x-1)$ fi

   Funkcionály z príkladov 3 a 4 sú spojité, pretože vzniknú kompozíciou monotónnych funkcií a funkčnej premennej (pozri nasledujúcu vetu).

5. $\tau[\phi](x):$ if $(\forall y \in N)[\phi(y) = y]$ then $\phi(x)$ else $\perp$ fi

   Funkcionál nie je spojitý. Uvažujme reťazec funkcií $\{f_i\}_0^\infty$ taký, že

   $$f_i(x):\quad \text{if } x < i \text{ then } x \text{ else } \perp \text{ fi.}$$

   $\tau[f_i] \equiv \Omega$, teda $\sqcup_0^\infty \tau[f_i] \equiv \Omega$. Ale $\sqcup_0^\infty f_i$ je identická funkcia, takže aj $\tau[\sqcup_0^\infty f_i]$ je identická. Funkcionál je monotónny, pretože ak $\Omega \sqsubseteq f$, potom $\tau[\Omega] \sqsubseteq \tau[f]$.

6. $\tau[\phi](x):$ if $\phi(x) \equiv \perp$ then 0 else $\perp$ fi

   Funkcionál nie je ani monotónny. Zvoľme funkciu $Z(x) \equiv 0$. Platí $\Omega \sqsubseteq Z$, ale $Z \equiv \tau[\Omega] \not\sqsubseteq \tau[z] \equiv \Omega$.

7. $\tau[\phi](x):$ if $\phi(x) = \perp$ then 0 else $\perp$ fi

   Funkcionál je spojitý podľa nasledujúcej vety.

5. Monotónne vs. spojité funkcie:

Lema 59 *Kompozícia dvoch monotónnych funkcií je monotónna funkcia.*

Veta 60 *Funkcionál τ zodpovedajúci termu t, ktorý je definovaný kompozíciou monotónnych funkcií a funkčnej premennej ϕ, je spojitý.*

Dôkaz: indukciou vzhľadom na konštrukciu t. V prípade, že $t = \phi$ alebo $t = h$ je dôkaz tvrdenia zrejmý. Pri indukcii rozlíšime dva prípady:

1. $t[\phi] = f(t_1[\phi], \cdots, t_n[\phi])$, kde f je nejaká monotónna funkcia. Na základe indukčného predpokladu spojitosti zodpovedajúcich funkcionálov $\tau_1 \cdots, \tau_n$ ukážeme spojitosť τ.

   - Nech $g \sqsubseteq h$, potom z monotónnosti τ_i vyplýva $\forall i:\ \tau_i[g] \sqsubseteq \tau_i[h]$. Odtiaľ z monotónnosti f dostaneme $\tau[g] = f(\tau_1[g], \cdots, \tau_n[g]) \sqsubseteq f(\tau_1[h], \cdots, \tau_n[h]) = \tau[h]$.
   - Pretože $\forall i:\ f_i \sqsubseteq \sqcup_0^\infty f_i$, z monotónnosti τ a f vyplýva $\tau[f_i] \sqsubseteq \tau[\sqcup_0^\infty f_i]$ a teda aj $\sqcup_0^\infty \tau[f_i] \sqsubseteq \tau[\sqcup_0^\infty f_i]$.

   Opačná inklúzia, a teda spojitosť τ, sa dokáže nasledovným spôsobom. Nech $v \in D_n$, potom z definície τ a indukčného predpokladu dostaneme

   $$\tau[\sqcup_0^\infty f_i](v) \equiv f(\tau_1[\sqcup_0^\infty f_i](v), \cdots, \tau_n[\sqcup_0^\infty f_i](v)) \equiv f((\sqcup_0^\infty \tau_1[f_i])(v), \cdots, (\sqcup_0^\infty \tau_n[f_i])(v)).$$

   Z existencie najmenšej hornej hranice pre postupnosť monotónnych funkcií vyplýva, že existuje také prirodzené číslo k, že pre j, $1 \le j \le n$ platí $(\sqcup_0^\infty \tau_j[f_i])(v) \equiv \tau_j[f_k](v)$. Využitím tohoto faktu dostávame

   $f((\sqcup_0^\infty \tau_1[f_i])(v), \cdots, (\sqcup_0^\infty \tau_n[f_i])(v)) \equiv f(\tau_1[f_k](v), \cdots, \tau_n[f_k](v)) \equiv f(\tau_1[f_k], \cdots, \tau_n[f_k])(v)$
   $\equiv \tau[f_k](v) \sqsubseteq (\sqcup_0^\infty \tau[f_i])(v)$.

   Dokázali sme $\tau[\sqcup_0^\infty f_i](v) \sqsubseteq (\sqcup_0^\infty \tau[f_i])(v)$ pre ľubovoľné v, a teda $\tau[\sqcup_0^\infty f_i] \sqsubseteq \sqcup_0^\infty \tau[f_i]$.

2. $t[\phi] = \phi(t_1[\phi], \cdots, t_n[\phi])$, kde ϕ je funkčná premenná. Opť treba dokázať, že keď sú zodpovedajúce funkcionály $\tau_1, \cdots, \tau_n$ spojité, má túto vlastnosť aj τ. Dôkaz prebieha analogicky ako v prípade 1. Namiesto monotónnosti f sa využíva monotónnosť f_i a $\sqcup_0^\infty f_i$.

Lema 61 *Každá spojitá funkcia je monotónna.*

Dôkaz: Nech $x \sqsubseteq y$, potom

$$f(x) \sqsubseteq \sqcup\{f(x), f(y)\} =_{spoj.} f(\sqcup\{x, y\}) = f(y)$$

12.3 Pevné body funkcionálov

Pevný bod – funkcia $f \in [D^n \mapsto D]$ je pevným bodom funkcionálu τ ak $\tau[f] \equiv f$.

Najmenší pevný bod – f je najmenším pevným bodom τ ak pre každý iný pevný bod g toho istého funkcionálu τ platí $f \sqsubseteq g$.

Veta 62 (Kleene) *Každý spojitý funkcionál τ má (jediný) najmenší pevný bod*

$$f_\tau =_{df} \sqcup_0^\infty \tau^i[\Omega],$$

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

Dôkaz: keďže τ je spojitý, je aj monotónny funkcionál – t.j. $\{\tau^i[\Omega]\}_0^\infty$ je reťazec v cpo $[D^n \mapsto D]$. Označme $f_\tau =_{df} \sqcup_0^\infty \tau^i[\Omega]$.

- f_τ je pevný bod τ, pretože $\tau[f_\tau] \equiv \tau[\sqcup_0^\infty \tau^i[\Omega]] \equiv_{spoj} \sqcup_0^\infty \tau^{i+1}[\Omega] \equiv f_\tau$.
- f_τ je najmenší pevný bod τ, pretože pre ľubovoľný pevný bod g a všetky $i \ge 0$ platí $\tau^i[\Omega] \sqsubseteq g$ (indukciou vzhľadom na i s využitím predpokladu spojitosti τ). Odtiaľ $\sqcup_0^\infty \tau^i[\Omega] = f_\tau \sqsubseteq g$.

Konštrukcia riešenia rovnice – $\phi(\overline{x}) = t[\phi](\overline{x})$

- definovať postupnosť aproximácií $\tau^i[\Omega]$ pre všetky $i \ge 0$,

- zostrojiť najmenšiu hornú hranicu reťazca $\sqcup_0^\infty \tau^i[\Omega]$.

Príklady pevných bodov:

1. $\tau[\phi](x):$ if $x = 0$ then 1 else $x.\phi(x-1)$ fi

   Postupnosť zodpovedajúcich aproximácií je:

   $$\tau^i[\Omega](x): \text{ if } x < i \text{ then } x! \text{ else } \perp \text{ fi.}$$

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

   $\tau^0[\Omega] = \Omega$
   $\tau^1[\Omega](x) =$ if $x = 0$ then 1 else $x.\ \tau^0[\Omega](x-1)$ fi $\equiv$ if $x = 0$ then 1 else $\perp$ fi
   $\tau^2[\Omega](x) =$ if $x = 0$ then 1 else $x.\ \tau^1[\Omega](x-1)$ fi $\equiv$
   $\equiv$ if $x = 0$ then 1 else $x.$ if $x - 1 = 0$ then 1 else $\perp$ fi fi $\equiv$
   $\equiv$ if $x = 0$ then 1 else if $x = 1$ then x else $\perp$ fi fi $\equiv$ if $x < 2$ then $x!$ else $\perp$ fi
   $\tau^3[\Omega](x) =$ if $x = 0$ then 1 else $x.\ \tau^2[\Omega](x-1)$ fi $\equiv$
   $\equiv$ if $x = 0$ then 1 else $x.$ if $x - 1 < 2$ then $x!$ else $\perp$ fi fi $\equiv$
   $\equiv$ if $x = 0$ then 1 else if $x < 3$ then $x.x - 1!$ else $\perp$ fi fi $\equiv$ if $x < 3$ then $x!$ else $\perp$ fi

2. $\tau[\phi](x):$ if $x = 0$ then 1 else $\phi(x+1)$ fi

   Pevnými bodmi pre $n \in N_\perp$ sú funkcie $f_n(x):$ if $x = 0$ then 1 else n fi.

   $\tau[f_n](x) \equiv$ if $x = 0$ then 1 else if $x + 1 = 0$ then 1 else n fi fi $\equiv$ if $x = 0$ then 1 else n fi $\equiv f_n$

   Najmenším pevným bodom je $f_\tau(x):$ if $x = 0$ then 1 else $\perp$ fi.

3. $\tau[\phi](x):$ if $x > 100$ then $x - 10$ else $\phi(\phi(x+11))$ fi

   Najmenším pevným bodom τ je funkcia:

   $$f_\tau: \text{ if } x > 100 \text{ then } x - 10 \text{ else } 91 \text{ fi.}$$

4. $\tau[\phi](x):$ if $\phi(x) \equiv 0$ then 1 else 0 fi

   Funkcionál τ nie je monotónny a nemá žiadny pevný bod. Keby funkcionál mal pevný bod f, potom z $f(0) \equiv 0$ vyplýva $\tau[f](0) \equiv 1$. Podobne z $f(0) \not\equiv 0$ vyplýva $\tau[f](0) \equiv 0$, čo je v spore s predpokladom.

5. $\tau[\phi](x):$ if $\phi(x) \equiv 0$ then 0 else 1 fi

   Funkcionál τ má dva neporovnateľné pevné body, funkcie $g:\ 0$ a $h:\ 1$, nemá však najmenší pevný bod.

6. $\tau[\phi](x,y):$ if $x = y$ then $y + 1$ else $\phi(x, \phi(x-1, y+1))$ fi

   Funkcionál τ má tri pevné body, h_τ je najmenším z nich.

   $f_\tau(x,y):$ if $x = y$ then $y + 1$ else $x + 1$ fi
   $g_\tau(x,y):$ if $x \geq y$ then $x + 1$ else $y - 1$ fi
   $h_\tau(x,y):$ if $x \geq y \wedge (x - y$ párne) then $x + 1$ else $\perp$ fi

12.4 Systémy rekurzívnych definícií

Riešenie systému rekurzívnych definícií –

$$\begin{aligned} \phi_1(\overline{x}) &\Longleftarrow t_1[\phi_1, \cdots, \phi_n](\overline{x}) \\ &\vdots \\ \phi_n(\overline{x}) &\Longleftarrow t_n[\phi_1, \cdots, \phi_n](\overline{x}) \end{aligned}$$

Usporiadanie – $\langle f_1, \cdots, f_n\rangle \sqsubseteq \langle g_1, \cdots, g_n\rangle$ práve vtedy, keď $f_i \sqsubseteq g_i$ pre všetky i.

Monotónnosť – n–tica $\langle f_1, \cdots, f_n\rangle$ je monotónna, ak je monotónna každá funkcia f_i.

Funkcionál – funkcionál

$$\tau = \langle \tau_1[\phi_1, \cdots, \phi_n], \cdots, \tau_n[\phi_1, \cdots, \phi_n]\rangle$$

zobrazuje n–ticu funkcií $\langle f_1, \cdots, f_n\rangle$ do n–tice $\langle g_1, \cdots, g_n\rangle$.

Najmenší pevný bod – n–tica funkcií $\langle f_1, \cdots, f_n\rangle$ je najmenším pevným bodom systému rekurzívnych definícií ak f_i je najmenším pevným bodom rovnice $\phi_i = t_i[\phi_1, \cdots, \phi_n]$ – t.j. funkcionálu $\mathcal{V}[\![t_i[\phi_1, \cdots, \phi_n]]\!]$.

Spojitý funkcionál – funkcionál $\tau = \langle \tau_1, \cdots, \tau_n\rangle$ je spojitý, ak sú spojité všetky τ_i.

Veta 63 *Funkcionál τ_i, zodpovedajúci termu t_i zloženému z monotónnych funkcií a funkčných premenných $\phi_1, \cdots, \phi_n$, je spojitý.*

Veta 64 *Každý spojitý funkcionál τ má najmenší pevný bod*

$$f_\tau = \langle f_{\tau_1}, \cdots, f_{\tau_n}\rangle.$$

12.5 Sémantika rekurzívnych programov

1. Sémantika booleovských výrazov: $\mathcal{W}:\ Bexp \mapsto [D^n \mapsto W]$.

- $\mathcal{W}[\![p(t_1, \cdots, t_n)]\!] =_{df} \beta(\mathcal{V}[\![t_1]\!], \cdots, \mathcal{V}[\![t_n]\!])$,
  kde β je prirodzené rozšírenie interpretácie symbolu p.

2. Sémantika výrazov: $\mathcal{V}:\ Exp \mapsto [D^n \mapsto D]$.

- $\mathcal{V}[\![x]\!] =_{df} x$
- $\mathcal{V}[\![f_F(s_1, \cdots, s_n)]\!] =_{df} \pi(\mathcal{V}[\![s_1]\!], \cdots, \mathcal{V}[\![s_n]\!])$,
  kde π je prirodzené rozšírenie interpretácie symbolu f_F,
- $\mathcal{V}[\![$**if** b **then** t_1 **else** t_2 **fi**$]\!] =_{df}$ if $\mathcal{W}[\![b]\!]$ then $\mathcal{V}[\![t_1]\!]$ else $\mathcal{V}[\![t_2]\!]$ fi,
- $\mathcal{V}[\![\phi(s_1, \cdots, s_n)]\!] =_{df} f_t(\mathcal{V}[\![s_1]\!], \cdots, \mathcal{V}[\![s_n]\!])$,
  kde f_t je najmenší pevný bod rovnice $\phi(\overline{x}) = t[\phi](\overline{x})$, t.j. funkcionálu $\mathcal{V}[\![t[\phi](\overline{x})]\!]$.

3. Sémantika programu: s telom $\lambda x_1, \cdots, x_n.\ t_o[\phi]$

$$\mathcal{M}:\ Exp \mapsto [D^n \mapsto D]$$

- $\mathcal{M}[\![P]\!] =_{df} \mathcal{V}[\![t_o[\phi](x_1, \cdots, x_n)]\!]$.

12.6 Výpočet rekurzívnych programov

1. Výpočtová (operačná) sémantika: pre každý program $P:\ \{t_0[\boxed{\phi}](x_1, \cdots, x_n), \mathcal{R}\}$ (kde $\mathcal{R}$ je systém rekurzívnych definícií) a ľubovoľné vstupné hodnoty (valuáciu premenných) $d_1, \cdots, d_n$ treba definovať výpočtovú postupnosť termov $t_0, t_1, \cdots, t_n, \cdots$.

Krok výpočtu – term t_{i+1} dostaneme prepísaním termu t_i pomocou **zjednodušujúcich a substitučných** pravidiel:

- zjednodušenie podvýrazov na základe vlastností interpretovaných funkcií (napr. $1 + 4 \to 5,\ 0\,.\,x \to 0,\ 0 = 1 \to ff,\ true \wedge x \to x$),
- výber podmnožiny funkčných premenných v terme t_i na základe daného **výpočtového pravidla**,
- substitúcia funkčných premenných, vybraných výpočtovým pravidlom, pravými stranami zodpovedajúcich rekurzívnych definícií.

Zjednodušenie – predpokladáme, že zjednodušujúce pravidlá spajú podmienku **konfluentnosti** odvodení

$$\forall t, u, v:\ (t \to^* u\ \wedge\ t \to^* v)\quad \exists w:\ (u \to^* w\ \wedge\ v \to^* w).$$

Potom nie je podstatné, v akom poradí sa term zjednodušuje.

Výpočtové pravidlo – algoritmus (nezávislý od programu P), ktorého výsledkom je pre ľubovoľný term t konečná, **neprázdna** podmnožina výskytov funkčných premenných z t.

Ukončenie výpočtu – term t_i je výsledkom výpočtu, ak neobsahuje funkčné premenné a nedá sa upraviť zjednodušujúcimi pravidlami.

Nekonečné výpočty – nekonečná výpočtová postupnosť (opodstatuje zavedenie prirodzených rozšírení funkcií).

2. Výpočtové pravidlá:

$$\phi(x)\ \Longleftarrow\ \textbf{if}\ x > 100\ \textbf{then}\ x - 10\ \textbf{else}\ \phi(\phi(x + 11))\ \textbf{fi}$$

$$f_\tau:\ \text{if } x > 100 \text{ then } x - 10 \text{ else } 91 \text{ fi.}$$

pravidlo LIS – "volanie hodnotou" – vyberie najľavejší výskyt funkčnej premennej, ktorého podtermy neobsahujú funkčnú premennú (leftmost–innermost)

$\underline{\phi}(99)\ \to\ \phi(\underline{\phi}(110))\ \to\ \underline{\phi}(100)\ \to\ \phi(\underline{\phi}(111))\ \to\ \underline{\phi}(101)\ \to\ 91,$

pravidlo LOS – "volanie menom" – vyberie najľavejší výskyt funkčnej premennej (leftmost–outermost)

$\underline{\phi}(99)\ \to\ \underline{\phi}(\phi(110))\ \to\ \textbf{if}\ \underline{\phi}(110) > 100\ \textbf{then}\ \phi(110) - 10\ \textbf{else}\ \phi(\phi(\phi(110) + 11))\ \textbf{fi}\ \to \underline{\phi}(\phi(\phi(110) + 11))\ \to^*\ \cdots,$

pravidlo PIS – vyberie všetky výskyty funkčných premenných, ktorých podtermy neobsahujú funkčné premennú (parallel–innermost)

$\underline{\phi}(99)\ \to\ \phi(\underline{\phi}(110))\ \to\ \underline{\phi}(100)\ \to\ \phi(\underline{\phi}(111))\ \to\ \underline{\phi}(101)\ \to\ 91,$

pravidlo POS – vyberie všetky vonkajšie výskyty funkčnej premennej, t.j. výskyty, ktoré nie sú v podterme nejakej funkčnej premennej (parallel–outermost)

$\underline{\phi}(99)\ \to\ \underline{\phi}(\phi(110))\ \to\ \textbf{if}\ \underline{\phi}(110) > 100\ \textbf{then}\ \underline{\phi}(110) - 10\ \textbf{else}\ \underline{\phi}(\phi(\phi(110) + 11))\ \textbf{fi}\ \to^*,$

pravidlo FAS – vyberie všetky výskyty s voľným argumentom, t.j. tie, ktoré majú aspo jeden argument nezávislý od ϕ (free argument)

$\underline{\phi}(99)\ \to\ \phi(\underline{\phi}(110))\ \to\ \underline{\phi}(100)\ \to\ \phi(\underline{\phi}(111))\ \to\ \underline{\phi}(101)\ \to\ 91,$

pravidlo FS – vyberie všetky výskyty funkčnej premennej (full)

$\underline{\phi}(99)\ \to\ \underline{\phi}(\underline{\phi}(110))\ \to\ \underline{\phi}(\underline{\phi}(111))\ \to\ 91.$

3. Výpočtové pravidlá pre systémy rekurzívnych definícií: Výpočtové pravidlá pre systémy rekurzívnych definícií sú priamym zovšeobecnením výpočtových pravidiel pre rekurzívne definície s jednou funkčnou premennou. Každé prepisovacie pravidlo vyberá výskyty ϕ_i podľa popísaného kritéria a nezávisle od i. Uveďme, ktoré výskyty vyberú jednotlivé pravidlá pre term

$$\phi_1(0, \phi_2(1)) + \phi_1(\phi_2(2), \phi_2(3))$$

s dvoma funkčnými premennými.

pravidlo LIS – "volanie hodnotou" – $\phi_1(0, \underline{\phi_2}(1)) + \phi_1(\phi_2(2), \phi_2(3))$

pravidlo LOS – "volanie menom" – $\underline{\phi_1}(0, \phi_2(1)) + \phi_1(\phi_2(2), \phi_2(3))$

pravidlo PIS – $\phi_1(0, \underline{\phi_2}(1)) + \phi_1(\underline{\phi_2}(2), \underline{\phi_2}(3))$

pravidlo POS – $\underline{\phi_1}(0, \phi_2(1)) + \underline{\phi_1}(\phi_2(2), \phi_2(3))$

pravidlo FAS – $\underline{\phi_1}(0, \underline{\phi_2}(1)) + \phi_1(\underline{\phi_2}(2), \underline{\phi_2}(3))$

pravidlo FS – $\underline{\phi_1}(0, \underline{\phi_2}(1)) + \underline{\phi_1}(\underline{\phi_2}(2), \underline{\phi_2}(3))$

4. Operačná vstupno–výstupná sémantika:

Program – pozostáva z tela programu (vstupného termu $\phi(x_1,\cdots,x_n)$) a rekurzívnej definície funkčnej premennej ϕ

$$P:\ \{\phi(\overline{x});\ \phi(\overline{x}) \Longleftarrow t[\phi](\overline{x})\}.$$

Denotačný význam – najmenší pevný bod f_P funkcionálu τ, zodpovedajúceho termu t v programe P.

Operačný význam – funkcia vypočítaná programom P na základe pevne zvoleného pravidla r:

$$c_P^r(\overline{d}) = \begin{cases} t & \text{ak výpočtová postupnosť končí termom } t : \phi(\overline{d}) \to^*_{P,r} t \\ \perp & \text{ak je výpočtová postupnosť nekonečná.} \end{cases}$$

Symbolický výpočet – výpočet na úrovni rekurzívnych schém, uvažujeme len neinterpretované termy;

- pri výpočte dajú použiť iba substitučné pravidlá; označenie

$$s \to_{P,r} t \qquad s \to^*_{P,r} t \qquad\qquad s \to_P t \qquad s \to^*_P t.$$

"Interpretovaný" výpočet – v jazyku, definovanom triedou prípustných interpretácií ($t^{\mathcal{I}}$ označenie termu t v interpretácii $\mathcal{I}$):

- $\mathcal{L}_1$ – monotónnymi funkciami a paralelnou interpretáciou if–then–else–fi
  (if $\perp$ then x else x fi $= x$),
- $\mathcal{L}_2$ – prirodzene rozšírenými funkciami a sekvenčnou interpretáciou if–then–else–fi.

5. Výpočtový diagram:

Výpočtový diagram $\mathcal{D}_t$ – k termu t v programe P tvorí množina termov $\{s:\ t \to^*_P s\}$, čiastočne usporiadaná reláciou $u \trianglelefteq v$ práve vtedy, keď $u \to^*_P v$ (nekonečný strom).

Poznámka: $\mathcal{D}_t$ obsahuje všetky potenciálne možné symbolické výpočty programu P, pre ľubovoľný vstup a každé pravidlo r.

Príklad: uvažujme definíciu $t[\phi,\phi]$ s dvomi výskytmi premennej ϕ:

$$\phi \to_P t[\phi,\phi] \to_P \begin{cases} t[t[\phi,\phi],\phi] & \to_P \cdots \\ t[\phi,t[\phi,\phi]] & \to_P \begin{cases} t[t[\phi,\phi],t[\phi,\phi]] & \to_P \cdots \\ t[\phi,t[t[\phi,\phi],\phi]] & \to_P \cdots \\ t[\phi,t[\phi,t[\phi,\phi]]] & \to_P \cdots \\ \cdots & \end{cases} \\ t[t[\phi,\phi],t[\phi,\phi]] & \to_P \cdots \end{cases}$$

Lema 65 *Ak pre termy $u,v \in \mathcal{D}_t$ platí $u \trianglelefteq v$, potom pri každej (monotónnej) interpretácii $\mathcal{I}$ platí $u^{\mathcal{I}}[\Omega] \sqsubseteq v^{\mathcal{I}}[\Omega]$, t.j. pre $\forall\overline{d}$ platí $u^{\mathcal{I}}[\Omega](\overline{d}) \sqsubseteq v^{\mathcal{I}}[\Omega](\overline{d})$.*

Dôkaz: indukciou vzhľadom na konštrukciu termu s využítím monotónnosti interpretovaných funkcií ($\Omega \sqsubseteq t^{\mathcal{I}}[\Omega,\Omega] \Rightarrow \tau^{\mathcal{I}}[\Omega,\Omega] \sqsubseteq t^{\mathcal{I}}[t^{\mathcal{I}}[\Omega,\Omega],\Omega]$ *atď.*).

Cesta v $\mathcal{D}_t$ zodpovedajúca pravidlu r a vstupu $\overline{d}$ –
Nech $s_0^{\mathcal{I}}[\Omega] \sqsubseteq s_1^{\mathcal{I}}[\Omega] \sqsubseteq \cdots \sqsubseteq s_n^{\mathcal{I}}[\Omega] \sqsubseteq \cdots$ je interpretovaný reťazec, zodpovedajúci ceste $s_0 \to_{P,r} s_1 \to_{P,r} \cdots \to_{P,r} s_n \to^*_{P,r} \cdots$. Potom platí

$$c_P^r(\overline{d}) = \sqcup_0^\infty \{s_i^{\mathcal{I}}[\Omega](\overline{d})\}.$$

Kanonická cesta – cesta zodpovedajúca pravidlu FS, t.j. symbolickému výpočtu

$$\phi \to_{FS} t[\phi,\phi] \to_{FS} t[t[\phi,\phi],t[\phi,\phi]] \to^*_{FS} \cdots.$$

Interpretáciou kanonickej cesty dostaneme reťazec identický s Kleeneho charakterizáciou najmenšieho pevného bodu:

$$\Omega \sqsubseteq t^{\mathcal{I}}[\Omega] \sqsubseteq (t^{\mathcal{I}})^2[\Omega] \sqsubseteq \cdots \sqsubseteq (t^{\mathcal{I}})^n[\Omega] \sqsubseteq \cdots$$

12.7 Korektnosť výpočtových pravidiel

1. Korektné výpočtové pravidlo (fixpointové pravidlo): výpočtové pravidlo r je korektné, ak pre každý program P platí $f_P \equiv c_P^r$, t.j.

$$\forall\overline{d}:\ c_P^r(\overline{d}) \equiv f_P(\overline{d}).$$

Lema 66 *Pravidlá LIS, PIS, LOS nie sú v jazyku $\mathcal{L}_1$ korektné.*

Dôkaz: Uvažujme rekurzívny program s rekurzívnou definíciou

$$\phi(x,y) \Longleftarrow \textbf{if } x=0 \textbf{ then } 0 \textbf{ else } \phi(x+1,\phi(x,y)) * \phi(x-1,\phi(x,y)) \textbf{ fi}$$

a paralelnou interpretáciou operácie $*$ násobenia, t.j. $0*\perp = \perp * 0 = 0$. Všetky použité funkcie sú monotónne. Potom

$$f_P(x,y):\ \text{if } x \equiv \perp \text{ then } \perp \text{ else } 0 \text{ fi}$$

ale PIS, LIS a LOS výpočty sa neskončia a preto $c_P^{LIS}(1,0) = c_P^{PIS}(1,0) = c_P^{LOS}(1,0) = \perp$.

Poznámka: pre pravidlo POS (FAS, FS) kontrapríklad nefunguje:

$$\underline{\phi}(1,0) \to \underline{\phi}(0,\phi(1,0)) \to [\phi(3,\phi(2,\phi(1,0))) * \phi(1,\phi(2,\phi(1,0)))] * 0 \to^* 0.$$

Lema 67 *Pravidlá LIS a PIS nie sú v jazyku $\mathcal{L}_2$ korektné.*

Dôkaz: Pre rekurzívny program s definíciou

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

platí $f_P(x,y)$: if $x \geq 0$ then 1 else $\perp$ fi ale LIS a PIS výpočty sa neskončia a teda $\phi(1,0) \to^*_{P,LIS}\perp$ a $\phi(1,0) \to^*_{P,PIS}\perp$.

2. Vzťah medzi f_P a c_P^r:

Veta 68 *Pre každý rekurzívny program P a ľubovoľné výpočtové pravidlo r platí $c_P^r \sqsubseteq f_P$.*

Dôkaz: Uvažujme výpočtovú cestu $s_0[\phi], s_1[\phi], \cdots, s_n[\phi], \cdots$ v $\mathcal{D}_t$, zodpovedajúcu výpočtu pravidlom r a ľubovoľnému vstupu $\overline{d}$ a kanonickú cestu $t_0[\phi], t_1[\phi], \cdots, t_n[\phi], \cdots$. Zrejme pre všetky i platí $s_i \to_P^* t_i$. Potom podľa lemy o interpretácii termov výpočtového diagramu, usporiadaných reláciou $\trianglelefteq$, platí $s_i^{\mathcal{I}}[\Omega](\overline{d}) \sqsubseteq t_i^{\mathcal{I}}[\Omega](\overline{d})$. Takže $\forall i:\ s_i^{\mathcal{I}}[\Omega](\overline{d}) \sqsubseteq t_i^{\mathcal{I}}[\Omega](\overline{d}) \sqsubseteq \sqcup_0^\infty t_i^{\mathcal{I}}[\Omega](\overline{d}) =_{df} f_P(\overline{d})$. Odtiaľ $\sqcup_0^\infty s_i^{\mathcal{I}}[\Omega](\overline{d}) =_{df} c_P^r(\overline{d}) \sqsubseteq f_P(\overline{d})$ pre ľubovoľný vstup $\overline{d}$.

Dôsledok 69

- *ak jedna z výpočítaných funkcií je pevným bodom P, potom je to najmenší pevný bod,*
- *rekurzívny program nemôže mať viac ako jeden "vypočitateľný" pevný bod,*
- *ak má rekurzívny program pevný bod a dva rôzne výpočty pre daný vstup sa skončia, musia sa skončiť rovnakým výsledkom,*
- *ak $f_P(\overline{d}) \equiv \perp$ pre $\overline{d} \in M \subseteq D^n$, potom žiadny výpočet, začínajúci vstupom $\overline{d} \in M$ sa nemôže skončiť,*
- *ak $f_P \equiv \Omega$, potom žiadny výpočet programu P sa nemôže skončiť,*
- *ak $f_P \not\equiv \Omega$, potom aspo jedna z funkcií, použitá v P, nie je prirodzene rozšírená.*

12.8 Kritéria korektnosti pravidiel

Sémantické kritérium – pravidlo je korektné pre ľubovoľný program a danú monotónnu interpretáciu.

Syntaktické kritérium – pravidlo je korektné pre ľubovoľnú monotónnu interpretáciu a daný program.

1. Sémantické kritérium: využíva vlastnosti tried interpetácií.

Označenie – výskyty ϕ v terme s vybrané pravidlom r označíme ϕ_1, ostatné ϕ_2, t.j. $s[\phi] =_{nt} s[\phi/\phi_1, \phi/\phi_2]$.

Bezpečná substitúcia – substitúcia za výskyty ϕ_1 je bezpečná pri interpretácii $\mathcal{I}$, ak pre ľubovoľný interpretovaný term $s^{\mathcal{I}}$

$$s^{\mathcal{I}}[\Omega/\phi_1, \Omega/\phi_2](\overline{d}) \equiv s^{\mathcal{I}}[\Omega/\phi_1, f_P/\phi_2](\overline{d}) \equiv \perp .$$

Bezpečné pravidlo - používa iba bezpečné substitúcie.

Intuícia – niektoré výskyty nestačí nahradiť ani "najvčšou" dostupnou informáciou, pokiaľ nezískame lepšiu aproximáciu o ostatných výskytoch. Bezpečné pravidlo forsíruje vyhodnotenie argumentov, ktoré sú pre výpočet podstatné.

2. Bezpečné výpočtové pravidlá:

Lema 70 *V jazyku $\mathcal{L}_1$ je pravidlo POS bezpečné.*

Dôkaz: indukciou vzhľadom na konštrukciu zjednodušeného termu s. Keďže pre $s \in F^0$ lema evidentne platí, uvažujeme dva prípady:

1. $s = \underline{\phi}(s_1[\phi], \cdots, s_n[\phi])$, potom pri každej interpretácii $\mathcal{I}$

$$\Omega(s_1^{\mathcal{I}}[\Omega], \cdots, s_n^{\mathcal{I}}[\Omega])(\overline{d}) \equiv \Omega(s_1^{\mathcal{I}}[f_P], \cdots, s_n^{\mathcal{I}}[f_P])(\overline{d}) \equiv \perp .$$

2. $s = f(a_1 \cdots, a_k, \underline{\phi}(\overline{s_1[\phi]}), \cdots, \underline{\phi}(\overline{s_m[\phi]}))$, kde $f \in [D^n \mapsto D]$. Keďže s je zjednodušený term, musí platiť $f(a_1, \cdots, a_k, \perp, \cdots, \perp) = \perp$. (Predpokladajme, že $f(\overline{a}, \perp, \cdots, \perp) = d$, potom z monotónnosti f vyplýva, že pre $\forall b_i$ platí $f(\overline{a}, b_1, \cdots, b_m) = d$. Keďže výsledok $f(\overline{a}, b_1, \cdots, b_m)$ nezávisí od výpočtu, musel sa dať odvodiť (zjednodušovaním) skôr, na základe prvých argumentov $\overline{a}$. To je v spore s predpokladom, že s je zjednodušený.) Odtiaľ

$$f(\overline{a}, \Omega(\overline{s_1^{\mathcal{I}}[\Omega]}), \cdots, \Omega(\overline{s_m^{\mathcal{I}}[\Omega]}))(\overline{d}) \equiv f(\overline{a}, \Omega(\overline{s_1^{\mathcal{I}}[f_P]}), \cdots, \Omega(\overline{s_m^{\mathcal{I}}[f_P]}))(\overline{d}) \equiv \perp$$

Poznámka: Príklad $f(\overline{a}, \phi(s_1[\phi]), \cdots, \phi(s_m[\phi]))$ dokumentuje, prečo nie sú pravidlá LIS a PIS v $\mathcal{L}_2$ bezpečné.

Lema 71 *Pravidlo FAS je bezpečné v $\mathcal{L}_1$, ak f_P nie je konštantná funkcia.*

Dôkaz: pre prípad $\phi(\underline{\phi}(\overline{s_1}), \cdots, \underline{\phi}(\overline{s_n}))$, kde $\overline{s_i}$ neobsahujú ϕ. Aby

$$f_P(\Omega(\overline{s_1^{\mathcal{I}}}), \cdots, \Omega(\overline{s_n^{\mathcal{I}}}))(\overline{d}) \equiv \perp \equiv \Omega(\Omega(\overline{s_1^{\mathcal{I}}}), \cdots, \Omega(\overline{s_n^{\mathcal{I}}}))(\overline{d}),$$

musí platiť $f_P(\perp, \cdots, \perp) \equiv \perp$, t.j. f_P nesmie byť konštantou.

Lema 72 *V jazyku $\mathcal{L}_2$ je bezpečné pravidlo LOS.*

Dôkaz: zaujímavý je prípad termu **if** $b[\phi_{LOS}]$ **then** $s_1[\phi]$ **else** $s_2[\phi]$ **fi**. Pravidlo LOS sa najprv pokúša substituovať v podmienke a zo striktnosti b a sekvenčnej interpretácie if $\perp$ then x else y fi $\equiv \perp$ vyplýva, že pravidlo je bezpečné.

Poznámka: Posledný prípad naznačuje, prečo nemôže byť pravidlo LOS bezpečné pri paralelnej interpretácii if–then–else–fi.

3. Korektnosť bezpečných pravidiel

Veta 73 *Každé bezpečné výpočtové pravidlo je korektné.*

Dôkaz: sporom, t.j. predpokladáme, že pravidlo r je bezpečné ale napriek tomu platí $c_P^r \not\equiv f_P$. Potom $\exists \overline{d} \in D^n:\ c_P^r(\overline{d}) \equiv \perp$ ale $f_P(\overline{d}) \not\equiv \perp$ a teda podľa vety o pevnom bode $\exists m:\ (t^m)^{\mathcal{I}}[\Omega] \not\equiv \perp$.

Nech $s_0[\phi], \cdots, s_n[\phi], \cdots$ je cesta v diagrame $\mathcal{D}_t$, zodpovedajúca pravidlu r a vstupu $\overline{d}$. Keďže $c_P^r(\overline{d}) \equiv \perp$, máme dve možnosti:

1. cesta je konečná a $s_n^{\mathcal{I}}[\Omega](\overline{d}) \equiv s_n^{\mathcal{I}}[f_P](\overline{d}) \equiv \perp$. Na druhej strane platí $s_0^{\mathcal{I}}[f_P](\overline{d}) \equiv f_P(\overline{d}) \not\equiv \perp$, čo je v spore s predpokladom, pretože $s_0^{\mathcal{I}}[f_P](\overline{d}) \sqsubseteq s_n^{\mathcal{I}}[f_P](\overline{d})$.

2. cesta je nekonečná. Potom každý term $s_i[\phi]$ obsahuje konečný počet výskytov ϕ v hbke $\leq m$ a teda $\exists N$ také, že $s_{N+1}[\phi]$ vzniká z $s_N[\phi]$ len substitúciou za výskyty v hbke $> m$ (označme tieto výskyty ϕ_1).

   Potom všetky výskyty Ω v $s_N[\Omega/\phi_1, t^m[\Omega]/\phi_2]$ sú v hbke $> m$. Odtiaľ z monotónnosti príslušných termov vyplýva

   $$(t^m)^{\mathcal{I}}[\Omega] \sqsubseteq s_N^{\mathcal{I}}[\Omega/\phi_1, (t^m)^{\mathcal{I}}[\Omega]/\phi_2] \sqsubseteq s_N^{\mathcal{I}}[\Omega/\phi_1, f_P/\phi_2].$$

   Keďže $(t^m)^{\mathcal{I}}[\Omega](\overline{d}) \not\equiv \perp$, musí byť aj $s_N^{\mathcal{I}}[\Omega/\phi_1, f_P/\phi_2] \not\equiv \perp$, čo odporuje predpokladu, že r je bezpečné pravidlo.

4. Syntaktické kritérium: platí nezávisle od interpretácie (univerzálne pravidlá).

Syntaktická dominancia – pravidlo r syntakticky dominuje nad pravidlom POS, ak vo výpočtovom diagrame $\mathcal{D}_t$ platí

$$\forall s, u: \ s \to_{P,POS} u \quad \exists v: \ s \to^*_{P,r} v \ \wedge \ u \to^*_P v.$$

Príklad: Pravidlo FS je bezpečné a syntakticky dominuje nad POS.

5. Univerzálne korektné pravidlo:

Lema 74 *Nech $s, u, v \in \mathcal{D}_t$ také, že $s \to_{P,POS} u$ a $s \to^*_P v$. Potom existuje w s vlastnosťou $v \to_{P,POS} w$ a $u \to^*_P w$.*

Dôkaz: tvrdenie vyplýva priamo z vlastnosti

$$\forall s, u, v: \ s \to_{P,POS} u \ \wedge s \to_P v \quad \exists w: \ v \to_{P,POS} w \ \wedge u \to_P w.$$

Túto "jednokrokovú" verziu lemy dokážeme indukciou vzhľadom na štruktúru termu s. V prípade, že s neobsahuje ϕ, tvrdenie vyplýva z pôvodného predpokladu.

1. $s = g(s_1, \cdots, s_n)$ Ak $s \to_{P,POS} u$, potom $u = g(u_1, \cdots, u_n)$, pričom platí $s_i \to_{P,POS} u_i$. Pri kroku $s \to_P v$, kde $v = g(v_1, \cdots, v_n)$, musí platiť buď $s_i \to_P v_i$ alebo $s_i = v_i$. Na základe indukčného predpokladu $\exists w_i: \ v_i \to_{P,POS} w_i$ a $u_i \to_P w_i$. Potom stačí položiť $w = g(w_1, \cdots, w_n)$.

2. $s = \phi(s_1, \cdots, s_n)$ Ak $s \to_{P,POS} u$, potom $u = t[\overline{s}/\overline{x}]$. Ak $s \to_P v$, potom term v vznikol nahradením neprázdnej množiny výskytov ϕ.

   (a) Ak vonkajší výskyt ϕ medzi ne nepatrí, potom $v = \phi(v_1, \cdots, v_n)$, pričom buď $s_i \to_P v_i$ alebo $s_i = v_i$. Stačí položiť $w = t[\overline{v}/\overline{x}]$.

   (b) Prípad, keď vonkajší výskyt medzi ne nepatrí nechávame čitateľovi ako cvičenie.

Veta 75 *Ak v $\mathcal{D}_t$ programu P pravidlo r syntakticky dominuje pravidlu POS, potom je vzhľadom na program P univerzálne korektné.*

Dôkaz: uvažujme dve cesty v diagrame $\mathcal{D}_t$ s rovnakým vstupom $s_0 = v_0$:

$$v_0 \to^*_{P,r} v_1 \to^*_{P,r} \cdots \to^*_{P,r} v_n \to^*_{P,r} \cdots$$
$$s_0 \to_{P,POS} s_1 \to_{P,POS} \cdots \to_{P,POS} s_n \to_{P,POS} \cdots$$

Z predpokladu syntaktickej dominantnosti vyplýva:

$$\forall v_i, w_{i+1}: \ v_i \to_{P,POS} w_{i+1} \quad \exists v_{i+1}: \ v_i \to^*_{P,r} v_{i+1} \ \wedge w_{i+1} \to^*_P v_{i+1}.$$

Vlastnosť ciest v $\mathcal{D}_t$, dokázaná predchádzajúcou lemou:

$$\forall s_i, s_{i+1}, v_i: \ s_i \to_{P,POS} s_{i+1} \ \wedge s_i \to^*_P v_i \ \exists w_{i+1}: \ v_i \to_{P,POS} w_{i+1} \ \wedge s_{i+1} \to^*_P w_{i+1}.$$

Odtiaľ indukciou vzhľadom na cesty $\{s_i\}$ a $\{v_i\}$ platí $\forall i: \ s_i \to^*_P v_i$.

Podľa lemy o vlastnostiach interpretovaných odvodení (POS je korektné) $f_P \equiv \sqcup_0^\infty \{s_i^{\mathcal{I}}[\Omega]\} \sqsubseteq \sqcup_0^\infty \{v_i^{\mathcal{I}}[\Omega]\}$, takže $c_P^r \equiv f_P$ ($c_P^r \sqsubseteq f_P$ vyplýva z vety 68).

Poznámka: Opačné tvrdenie uvedenej vety platí len za predpokladu, že schéma $\mathcal{D}_t$ nie je beznádejná. Schéma je beznádejná, ak sú v $t^{\mathcal{I}}[\Omega]$ všetky listy Ω (potom $f_P \equiv \Omega$).

Kapitola 13

Nedeterministické programy a schémy

V tejto kapitole nám pôjde o zdôraznenie nasledujúcich troch myšlienok:

- abstrakciu triedy nedeterministických programov (triedou schém s "vyšším" stupom abstrakcie),
- vyjadrenie významu nedeterministických programov a schém vo formálnom matematickom jazyku.

Doteraz sme študovali riadiace štruktúry tzv. deterministických programov. Abstrakciou tejto triedy programov sú "deterministické' schémy, ktoré sme študovali v predchádzajúcich kapitolách. V niektorých programovacích jazykoch existujú konštrukcie, ktoré im dávajú "nedeterministický" charakter. Príkladom môže byť konštrukcia umožujúca nedeterministický výber pokračovania výpočtu (príkaz výberu – fork) alebo Dijkstrove "guarded commands".

Zatiaľ čo v prípade deterministických programov je ich vstupno–výstupný význam charakterizovaný funkciami, v triede nedeterministických programov charakterizujú vstupno–výstupný vzťah relácie. Pomerne priamočiaro sa dá ku každému nedeterministickému programu "skonštruovať" jeho vstupno–výstupná relácia. Okrem toho ukážeme, že takúto konštrukciu možno zovšeobecniť aj na úrove schém. Každú nedeterministickú programovú schému vieme vyjadriť ako relačnú schému nad tzv. abstraktnou relačnou algebrou.

Pri štúdiu riadiacich štruktúr programov je možné uplatniť aj abstraktnejší pohľad ako ten, ktorý sme formalizovali definovaním triedy štandardných schém. V triede schém, definovaných v tejto kapitole, sú základnými symbolmi symboly elementárnych – atomických príkazov (napr. priradenia). Predpokladáme, že atomické príkazy menia "nejakým" spôsobom stav výpočtu, nezaujíma nás však ani štruktúra stavu, ani ako sa táto zmena realizuje. Stav je teda chápaný ako monolitný objekt (nie sú známe, žiadne informácie o jeho štruktúre).

13.1 Syntax nedeterministických schém

Symboly – reprezentujúce atomické príkazy a atomické predikáty:

- atomické príkazy – $\{\mathcal{A}, \mathcal{A}_i, \cdots\}$,
- atomické predikáty – $\{p, q, \cdots\}$.

Príkazy – $\{S, S_i, \cdots\}$

$$S ::= \mathcal{A} \;|\; \textbf{skip} \;|\; \textbf{abort} \;|\; S_1; S_2 \;|\; S_1 \mid S_2 \;|\; \textbf{if } p \textbf{ then } S_1 \textbf{ else } S_2 \textbf{ fi} \;|\; \textbf{while } p \textbf{ do } S \textbf{ od}$$

Strohú definíciu syntaxe príkazov doplníme jednoduchým vysvetlením jednotlivých príkazov:

- $\mathcal{A}$ – elementárny príkaz,
- **skip** – prázdny príkaz,
- **abort** – príkaz prerušenia,
- $S_1; S_2$ – zložený príkaz,
- $S_1 \mid S_2$ – príkaz nedeterministického výberu,
- **if** p **then** S_1 **else** S_2 **fi** – podmienkový príkaz,
- **while** p **do** S **od** – príkaz cyklu.

Intuitívny význam väčšiny príkazov je známy. Pripomíname len, že $S_1 \mid S_2$ definuje nedeterministický výber jednej zo zložiek (nie obe!).

Schéma – príkaz S, uzavretý v syntaktických "zátvorkách" **begin, end**

$$N: \;\; \textbf{begin } S \textbf{ end} \;.$$

Triedu nedeterministických programových schém budeme označovať $\mathcal{N}$.

Príklad:

```
N: begin
     if q then A; skip
        else while p do A1; A2 | A od
     fi
   end
```

Cvičenie: Pozorný čitateľ si už iste povšimol určitú príbuznosť medzi takto definovanou triedou nedeterministických schém a triedou Janovových schém. Sformulujte, v čom spočíva podobnosť medzi týmito triedami schém a ako ovplyvní táto príbuznosť rozhodnuteľnosť základných problémov pre triedu nedeterministických schém.

13.2 Interpretácia nedeterministických schém

Doteraz sme programy spájali s funkciami. V triede nedeterministických programov však takáto charakterizácia nestačí. K danému vstupu môže existovať viac "zmysluplných" výstupov. Prirodzeným spôsobom vyjadrenia vstupno–výstupného významu programu je relácia.

Uvažujme nasledujúce relácie z triedy relácií nad neprázdnou množinou stavov V:

- binárne relácie nad V – $R \subseteq V \times V$,
- predikátové relácie – podmožiny jednotkovej relácie $I = \{\langle x, x\rangle : x \in V\}$, určené charakteristickou funkciou p:

$$p = \{\langle x, x\rangle : p(x) = 1\} \subseteq I,$$
$$\overline{p} = \{\langle x, x\rangle : p(x) = 0\} \subseteq I.$$

"Spriahnuté" predikátové relácie p a $\overline{p}$ sú vzhľadom k I navzájom komplementárne.

1. Interpretácia symbolov: Interpretáciou nedeterministickej programovej schémy je dvojica $\mathcal{I} = (V, i)$, definovaná

- oborom interpretácie, t.j. množinou stavov V,
- interpretačným morfizmom, ktorý priradí ku každému
  - predikátovému symbolu zodpovedajúcu predikátovú reláciu, t.j. $i(p) = \{\langle x, x\rangle : p(x) = 1\} \subseteq I$,
  - symbolu atomického príkazu reláciu nad množinou stavov, t.j. $i(\mathcal{A}) = A \subseteq V \times V$.

2. Relačná algebra nad množinou stavov V: Pri definícii významu interpretovaných nedeterministických programových schém sa budeme opierať o algebru relácií $\mathcal{R}el_V$ nad množinou stavov V, definovanú "štandardnými" relačnými operáciami:

Konštantné relácie:

- jednotková relácia: $I = \{\langle x, x\rangle : x \in V\}$,
- prázdna relácia: $E = \emptyset$,
- univerzálna relácia: $U = V \times V$,

Unárne operácie:

- konverzia: $\widetilde{R} = \{\langle x, y\rangle : \langle y, x\rangle \in R\}$,
- komplement: $\overline{R} = \{\langle x, y\rangle : \langle x, y\rangle \in U \wedge \langle x, y\rangle \notin R\}$,
- tranzitívny uzáver: $R^+ = R \cup R^2 \cup \cdots \cup R^i \cup \cdots = \cup_{i=1}^{\infty} R^i$, kde $R^{i+1} = R; R^i$,
- reflexívny a tranzitívny uzáver: $R^* = I \cup R^+$,

Binárne operácie:

- kompozícia: $R_1; R_2 = \{\langle x, y\rangle : \exists z \; xR_1z \wedge zR_2y\}$,

- zjednotenie: $R_1 \cup R_2 = \{\langle x, y\rangle : xR_1y \vee xR_2y\}$,
- prienik: $R_1 \cap R_2 = \{\langle x, y\rangle : xR_1y \wedge xR_2y\}$.

Poznámka: Upozorujeme, že symbol ; budeme používať v dvoch rôznych významoch, ako programový konštruktor resp. ako operáciu na reláciách. Predpokladáme, že z kontextu, v ktorom sa symbol použije, sa bude vždy dať určiť jeho jednoznačný význam.

Základné vlastnosti relačných operácií:

- $(R_1; R_2); R_3 = R_1; (R_2; R_3)$
- $R_1; (R_2 \cup R_3) = R_1; R_2 \cup R_1; R_3$
- $(R_1 \cup R_2); R_3 = R_1; R_3 \cup R_2; R_3$
- $R; R^* = \bigcup_{i=0}^{\infty} R; R^i$
- $R^*; R = \bigcup_{i=0}^{\infty} R^i; R$
- $R; R^* \cup I = R^*$
- $R; I = I; R = R$
- $R; E = E; R = E, \quad R \cup E = E \cup R = R$
- $R_1 \subseteq R_2 \Rightarrow R; R_1 \subseteq R; R_2, \quad R_1; R \subseteq R_2; R$

Vlastnosti predikátových relácií:

- $p \cap \overline{p} = E, \quad p \cup \overline{p} = I$
- $p; q = p \cap q$
- $\overline{\overline{p}} = p$

3. Význam nedeterministických programov: Uvažujme ľubovoľnú nedeterministickú programovú schému N a interpetáciu $\mathcal{I}$. Každý nedeterministický program $P = (N, \mathcal{I})$ potom možno vyjadriť ako reláciu R, pričom xRy práve vtedy, keď vstupný stav x je programom P transformovaný do výstupného stavu y. Nech $\mathcal{P_N}$ je trieda relačných programov, ktoré dostaneme "relačnými" interpretáciami nedeterministických schém $\mathcal{N}$. Sémantiku relačných programov vyjadríme zobrazením

$$\mathcal{S} : \; \mathcal{P_N} \; \mapsto \; \mathcal{R}el_V.$$

Príslušnú reláciu $R \in \mathcal{R}el_V$ skonštruujeme indukciou vzhľadom na konštrukciu zodpovedajúceho relačného programu $P = (N, \mathcal{I})$. Predpokladáme, že relácie R, R_1, R_2 zodpovedajú programovým segmentom S, S_1, S_2. Základnými objektami sú relácie $i(\mathcal{A}) \in \mathcal{R}el_V$, zodpovedajúce elementárnym príkazom $\mathcal{A}$.

- príkaz prerušenia **abort** preložíme do prázdnej relácie E,

- prázdny príkaz **skip** preložíme do jednotkovej relácie I,
- zložený príkaz $S_1; S_2$ preložíme pomocou kompozície relácií $R_1; R_2$,
- príkaz nedeterministického výberu $S_1 \mid S_2$ nahradíme zjednotením relácií $R_1 \cup R_2$,
- podmienkový príkaz **if** p **then** S_1 **else** S_2 **fi** preložíme do relačného výrazu $(p; R_1) \cup (\overline{p}; R_2)$,
- príkaz cyklu **while** p **do** S **od** vyjadríme relačným výrazom $(p; R)^*; \overline{p}$,
- programu **begin** S **end** priradíme reláciu R.

Pomerne priamočiaro úvahou sa dá overiť, že pre takto skonštruovanú reláciu $R \in \mathcal{R}el_V$ platí xRy práve vtedy, keď program $P \in \mathcal{P}_{\mathcal{N}}$ transformuje vstupný stav x do výstupného stavu y.

Definícia 76 *Hovoríme, že dva nedeterministické programy P_1, P_2 sú ekvivalentné, ak sú ekvivalentné relácie R_1, R_2, zodpovedajúce programom P_1 a P_2.*

4. Relačná charakterizácia zaujímavých riadiacich príkazov: Pre zaujímavosť uvádzame relačnú charakterizáciu niektorých známych riadiacich konštrukcií i menej známych Dijkstrových nedeterministických konštrukcií:

podmienka bez alternatívy:

$$\textbf{if } p \textbf{ then } S \textbf{ fi} \quad \equiv \quad \textbf{if } p \textbf{ then } S \textbf{ else skip fi} \quad \leadsto \quad p; R \cup \overline{p}$$

cyklus **repeat***:*

$$\textbf{repeat } S \textbf{ until } p \quad \leadsto \quad R; (\overline{p}; R)^*; p$$

tzv. jeden a pol cyklus:

$$\textbf{loop } S_1; \textbf{ when } p \textbf{ do exit }; S_2 \textbf{ pool} \quad \leadsto \quad R_1; (\overline{p}; R_2; R_1)^*; p$$

nedeterministická viacnásobná podmienka:

$$\textbf{if} \quad \Box p_1 \Rightarrow S_1 \quad \Box p_2 \Rightarrow S_2 \cdots \quad \Box p_n \Rightarrow S_n \quad \textbf{fi} \quad \leadsto$$

$$\leadsto \quad (p_1; R_1 \cup p_2; R_2 \cup \cdots \cup p_n; R_n) \cup \overline{p_1}; \overline{p_2}; \ldots; \overline{p_n}$$

nedeterministický viacnásobný cyklus:

$$\textbf{do} \quad \Box p_1 \Rightarrow S_1 \quad \Box p_2 \Rightarrow S_2 \cdots \quad \Box p_n \Rightarrow S_n \quad \textbf{od} \quad \leadsto$$

$$\leadsto \quad (p_1; R_1 \cup p_2; R_2 \cup \cdots \cup p_n; R_n)^*; \overline{p_1}; \overline{p_2}; \ldots; \overline{p_n}$$

Poznámka: Treba si uvedomiť, že $\overline{p_1}; \overline{p_2} \neq \overline{p_1; p_2}$.

13.3 Význam nedeterministických schém

1. Abstraktná relačná algebra:

Definícia 77 *Algebraickú štruktúru*

$$\mathcal{A}bs = \langle \mathcal{M}, ;, \cup, \cap, \widetilde{\ }, \overline{\ }, I, E, U \rangle$$

nazývame abstraktnou relačnou algebrou ak

- *$\mathcal{M}$ je ľubovoľná množina,*
- *$;, \cup, \cap$ sú binárne operácie nad podmnožinami $\mathcal{M}$,*
  $\widetilde{\ }, \overline{\ }$ sú unárne operácie nad podmnožinami $\mathcal{M}$,
  I, E, U sú nulárne operácie (konštanty),
- *$\langle \mathcal{M}, \cup, \cap, \overline{\ }, E, U \rangle$ je Booleova algebra s nulovým prvkom E a jednotkým prvkom U,*
- *pre všetky $R_M, S_M, T_M \subseteq \mathcal{M}$ majú operácie $;, \widetilde{\ }, I$ vlastnosti:*

$$(R_M; S_M); T_M = R_M; (S_M; T_M)$$
$$\widetilde{\widetilde{R_M}} = R_M$$
$$\widetilde{R_M; S_M} = \widetilde{S_M}; \widetilde{R_M}$$
$$R_M; I = R_M$$
$$(R_M; S_M) \cap T_M = E \implies (S_M; \widetilde{T_M}) \cap \widetilde{R_M} = E$$

V abstraktnej relačnej algebre platí celý rad zaujímavých vlastností.

Lema 78 *Pre všetky $R_M, S_M, T_M \subseteq \mathcal{M}$ platí:*

ak $R_M \subseteq S_M$, potom $\widetilde{R_M} \subseteq \widetilde{S_M}$, $R_M; T_M \subseteq S_M; T_M$, $T_M; R_M \subseteq T_M; S_M$,
$E; R_M = R_M; E = E$ a $I; R_M = R_M$,
$\widetilde{E} = E$, $\widetilde{I} = I$, $\widetilde{U} = U$,
$R_M; (S_M \cup T_M) = R_M; S_M \cup R_M; T_M$, $(S_M \cup T_M); R_M = S_M; R_M \cup T_M; R_M$,
$\widetilde{R_M \cup S_M} = \widetilde{R_M} \cup \widetilde{S_M}$, $\widetilde{R_M \cap S_M} = \widetilde{R_M} \cap \widetilde{S_M}$, $\overline{\widetilde{R_M}} = \widetilde{\overline{R_M}}$.

Lema 79 *Pre všetky $R_M, S_M, T_M \subseteq \mathcal{M}$ platí*

$$R_M; S_M \cap T_M = R_M; (\widetilde{R_M}; T_M \cap S_M) \cap T_M.$$

Lema 80 *Pre všetky $R_M, S_M, T_M \subseteq \mathcal{M}$ platí:*

ak $\widetilde{R_M}; R_M \subseteq I$, potom $R_M; (S_M \cap T_M) = R_M; S_M \cap R_M; T_M$,
ak $R_M \subseteq I$, potom $\widetilde{R_M} = R_M$,
$R_M = (R_M; U \cap I); R_M$, $\quad R_M; U \subseteq (R_M; U \cap I); U$, $R_M; U \cap I = R_M; \widetilde{R_M} \cap I$.

Priamočiaro sa dá dokázať, že relačná algebra $\mathcal{R}el_V$ spa vlastnosti z definície abstraktnej relačnej algebry $\mathcal{A}bs$. Na druhej strane, nie všetky vlastnosti relačnej algebry platia v abstraktnej relačnej algebre.

Veta 81 *Algebra relácií $\mathcal{R}el_V$ je abstraktnou relačnou algebrou.*

2. Sémantika relačných schém: Nech $\mathcal{N}$ je trieda nedeterministických programových schém. Sémantickým oborom pri popise významu nedeterministických schém bude abstraktná relačná algebra $\mathcal{A}bs$:

$$\mathcal{S}_\mathcal{N} \quad : \quad \mathcal{N} \quad \mapsto \quad \mathcal{A}bs.$$

Sémantikou programovej schémy bude abstraktná relácia R_M.

Sémantickú funkciu $\mathcal{S}_\mathcal{N}$ definujeme indukciou vzhľadom na konštrukciu schémy. Predpokladáme, že význam symbolu elementárneho príkazu $\mathcal{A}$ reprezentuje "abstraktná" reláca A_M a význam predikátového symbolu p reprezentuje dvojica "abstraktných" relácií $p_M, \overline{p_M}$ takých, že $p_M \cup \overline{p_M} = I$ a $p_M \cap \overline{p_M} = E$. Pre zjednodušenie zápisu používame nasledujúce dve skratky:

$$(p; R_1) \cup (\overline{p}; R_2) \equiv p \to R_1, R_2 \qquad\qquad (p; R)^*; \overline{p} \equiv p * R.$$

- $\mathcal{S}_\mathcal{N}\llbracket \mathcal{A} \rrbracket = A_M$
- $\mathcal{S}_\mathcal{N}\llbracket \mathbf{skip} \rrbracket = I$
- $\mathcal{S}_\mathcal{N}\llbracket \mathbf{abort} \rrbracket = E$
- $\mathcal{S}_\mathcal{N}\llbracket S_1; S_2 \rrbracket = \mathcal{S}_\mathcal{N}\llbracket S_1 \rrbracket; \mathcal{S}_\mathcal{N}\llbracket S_2 \rrbracket$
- $\mathcal{S}_\mathcal{N}\llbracket S_1 \mid S_2 \rrbracket = \mathcal{S}_\mathcal{N}\llbracket S_1 \rrbracket \cup \mathcal{S}_\mathcal{N}\llbracket S_2 \rrbracket$
- $\mathcal{S}_\mathcal{N}\llbracket \mathbf{if}\ p\ \mathbf{then}\ S_1\ \mathbf{else}\ S_2\ \mathbf{fi}\ \rrbracket = p_M \to \mathcal{S}_\mathcal{N}\llbracket S_1 \rrbracket, \mathcal{S}_\mathcal{N}\llbracket S_2 \rrbracket$
- $\mathcal{S}_\mathcal{N}\llbracket \mathbf{while}\ p\ \mathbf{do}\ S\ \mathbf{od}\ \rrbracket = p_M * \mathcal{S}_\mathcal{N}\llbracket S \rrbracket$
- $\mathcal{S}_\mathcal{N}\llbracket \mathbf{begin}\ S\ \mathbf{end}\ \rrbracket = \mathcal{S}_\mathcal{N}\llbracket S \rrbracket$

Príklad: Nedeterministickej schéme N z úvodného príkladu kapitoly zodpovedá abstraktná relácia

$$q_M \to (A_M; I), [(p_M; A_{1M}; (A_{2M} \cup A_M))^*; \overline{p_M}].$$

Ľahko sa môžeme presvedčiť, že pre ľubovoľnú nedeterministickú schému N a interpretáciu $\mathcal{I}$ skonštruujeme tú istú reláciu $R \in Rel_V$ dvomi rôznymi postupmi:

- interpretáciou schémy N a konštrukciou relácie R k programu $P = (N, \mathcal{I})$,
- interpretáciou abstraktnej relácie $\mathcal{S}_\mathcal{N}\llbracket N \rrbracket$, charakterizujúcej význam schémy N.

3. Vlastnosti nedeterministických schém v relačnom kalkule: V relačnom kalkule sa dajú vyjadriť mnohé "zaujímavé" vlastnosti relácií, napr.:

R *je ekvivalencia* – práve vtedy, keď platí $I \subseteq R$ (reflexívnosť), $R \subseteq \widetilde{R}$ (symetria), $R; R \subseteq R$ (tranzitívnosť),

R *je funkcia* – práve vtedy, keď platí $\widetilde{R}; R \subseteq I$ (t.j. $xRy \wedge xRz \Rightarrow y = z$),

R *je totálna* – práve vtedy, keď platí $I \subseteq R; \widetilde{R}$.

Cvičenie: Ukážte, že relácia $(R; \widetilde{R}) \cap I$ definuje množinu vstupov relácie R, pre ktoré existuje aspo jeden "konečný výsledok".

Podobne sa dajú charakterizovať základné vlastnosti nedeterministických programových schém.

Definícia 82 *Nedeterministické schémy N_1, N_2 sú ekvivalentné práve vtedy keď sú ekvivalentné zodpovedajúce abstraktné relácie, t.j.* $\mathcal{S}_\mathcal{N}\llbracket N_1 \rrbracket = \mathcal{S}_\mathcal{N}\llbracket N_2 \rrbracket$.

Definícia 83 *Nedeterministická schéma N diverguje práve vtedy, keď* $\mathcal{S}_\mathcal{N}\llbracket N \rrbracket = E$.

Pri definícii pojmu zastavenia predpokladáme, že pre každý vstup existuje aspo jeden výstup, t.j. v relačnom kalkule $\forall x \exists y\ xRy$.

Definícia 84 *Nedeterministická schéma N sa zastaví práve vtedy, keď* $I \subseteq \mathcal{S}_\mathcal{N}\llbracket N \rrbracket; \widetilde{\mathcal{S}_\mathcal{N}\llbracket N \rrbracket}$.

Cvičenie: Ukážte, že uvedené definície zodpovedajú v relačnom kalkule definovaným pojmom.

Na záver niekoľko príkladov dôkazu ekvivalencie relačných programových schém.

Príklad: Pre $R, R_1, R_2, R_3 \in \mathcal{A}bs$ platí:

1. $p \to R, R = R$

   $p; R \cup \overline{p}; R = (p \cup \overline{p}); R = I; R = R$
2. $(p \to R_1, R_2); R = p \to (R_1; R), (R_2; R)$

   $(p; R_1 \cup \overline{p}; R_2); R = p; R_1; R \cup \overline{p}; R_2; R$
3. $p \to (p \to R_1, R_2), R_3 = p \to R_1, R_3$

   $p; (p; R_1 \cup \overline{p}; R_2) \cup \overline{p}; R_3 = p; p; R_1 \cup p; \overline{p}; R_2 \cup \overline{p}; R_3 = p; p; R_1 \cup E; R \cup \overline{p}; R_3 = p; R_1 \cup \overline{p}; R_3$
4. $p * R = p \to (R; p * R), I$

   $p * R = (p; R)^*; \overline{p} = (p; R; (p; R)^* \cup I); \overline{p} = p; R; (p; R)^*; \overline{p} \cup I; \overline{p} = p; R; p * R \cup \overline{p}$

Cvičenie: Dokážte, že pre $R, R_1, R_2 \in \mathcal{A}bs$ platia nasledujúce vlastnosti:

- $p * (p * R) = p * R$
- $(p_1 \cup p_2) * R = p_1 * R; p_2 * (R; p_1 * R)$

13.4 Správnosť programov v relačnom kalkule

1. Relačné induktívne formuly: Čiastočnú správnosť programu P vzhľadom na vstupnú podmienku p a výstupnú podmienku q vyjadruje jednoduchá relačná formula

$$p; P \ \subseteq \ P; q$$

Nasledujúce vlastnosti relácií sú ekvivalentným vyjadrením induktívnej formuly $p; P \ \subseteq \ P; q$:

- $\forall x, y[xpx \;\wedge\; xPy \;\Rightarrow\; yqy]$,
- $\forall y[\exists x(xpx \;\wedge\; xPy) \;\Rightarrow\; yqy]$,
- $\forall x[xpx \;\Rightarrow\; \forall y(xPy \;\Rightarrow\; yqy)]$.

Označme

$$(p \circ P)(x) \;\equiv\; \exists y(ypy \;\wedge\; yPx) \qquad\qquad (P \to q)(x) \;\equiv\; \forall y(xPy \;\Rightarrow\; yqy).$$

Ľahko sa presvedčíme, že relačné operátory $p \circ P$ a $P \to q$ zodpovedajú najsilnejšej výstupnej, resp. najslabšej vstupnej podmienke. Potrebné výsledky sumarizuje nasledujúca lema.

Lema 85

- $p; P \subseteq P; (p \circ P)$, $\qquad (P \to q); P \subseteq P; q$,
- *for all* p, q, *if* $p; P \subseteq P; q$ *then* $(p \circ P) \subseteq q$ *and* $p \subseteq (P \to q)$,
- $p \circ P = \bigcap\{q \mid p; P \subseteq P; q\}$, $\qquad P \to q = \bigcup\{p \mid p; P \subseteq P; q\}$.

Cvičenie: Dokážte tvrdenia lemy na základe vlastností v relačnom kalkule.

2. Inferenčný systém v relačnom kalkule:

Pravidlo kompozície – $p; P_1 \subseteq P_1; q \quad q; P_2 \subseteq P_2; r \;\vdash\; p; (P_1; P_2) \subseteq (P_1; P_2); r$

Pravidlo alternatívy – $p; b; P_1 \subseteq P_1; q \quad p; \overline{b}; P_2 \subseteq P_2; q \;\vdash\; p; (b \to P_1, P_2) \subseteq (b \to P_1, P_2); q$

Pravidlo cyklu – $p; b; P \subseteq P; p \;\vdash\; p; (b * P) \subseteq (b * P); \overline{b}; p$

Pravidlo následku – $p \subseteq p_1 \quad q_1 \subseteq q \quad p_1; P \subseteq P; q_1 \;\vdash\; p; P \subseteq P; q$