řečeno, chování programu nás zajímá pouze pro ty vstupní hodnoty z $D_{\bar{x}}$, které splňují predikát $\varphi(\bar{x})$. Ve speciálním případě, kdy připouštíme, že libovolný prvek z $D_{\bar{x}}$ může být užit jako vstupní hodnota, klademe $\varphi(\bar{x})$ rovno T: $\varphi(\bar{x})$ je pak *pravda* pro všechny prvky z $D_{\bar{x}}$.

2. Totální predikát $\psi(\bar{x}, \bar{z})$ nad $D_{\bar{x}} \times D_{\bar{z}}$, nazývaný *výstupní predikát*; popisuje vztah, který musí splňovat výstupní hodnoty vzhledem ke vstupním hodnotám v okamžiku skončení výpočtu.

Pomocí těchto predikátů definujeme základní pojmy:

1. Program P *končí pro* φ, jestliže program končí pro všechny vstupní hodnoty $\bar{\xi}$, pro něž je $\varphi(\bar{\xi})$ *pravda*.

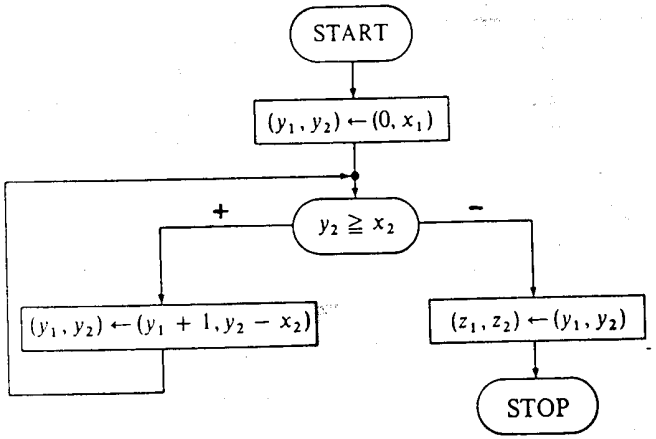
2. Program P je *parciálně korektní vzhledem k* φ *a* ψ, jestliže pro každou hodnotu $\bar{\xi}$, pro niž $\varphi(\bar{\xi})$ je *pravda* a pro niž P končí, je i $\psi(\bar{\xi}, P(\bar{\xi}))$ *pravda*.

3. Program P je *totálně korektní vzhledem k* φ a ψ, jestliže pro každou hodnotu $\bar{\xi}$, pro niž $\varphi(\bar{\xi})$ je *pravda*, program P končí a při tom $\psi(\bar{\xi}, P(\bar{\xi}))$ je *pravda*.

V případě parciální korektnosti se tedy nezajímáme o otázku ukončení programu, při vyšetřování totální korektnosti je tato otázka podstatná. *Verifikovat program* vzhledem k danému vstupnímu predikátu $\varphi(\bar{x})$ a výstupnímu predikátu $\psi(\bar{x}, \bar{z})$ znamená prokázat jeho totální korektnost vzhledem k φ a ψ. Obvykle je vhodné verifikovat program ve dvou krocích: v prvním dokázat parciální korektnost vzhledem k φ a ψ, v druhém ukázat, že program končí pro φ.[1])

Budeme se nyní zabývat metodami dokazování parciální korektnosti i ukončení programů. Ukažme nejprve možný postup na příkladě programu z obr. 3.1. Nejdříve dokážeme jeho parciální korektnost vzhledem ke vstupnímu predikátu

$$\varphi(x_1, x_2): x_1 \geqq 0 \wedge x_2 \geqq 0$$



Obr. 3.1. Program pro dělení celých čísel

[1]) O metodách přímého důkazu totální korektnosti bude zmínka v dodatku A. Pozn. překl.

(který vyjadřuje, že nás zajímá průběh programu v případě, že jak x_1 tak x_2 je nezáporné) a výstupnímu predikátu

$$\psi(x_1, x_2, z_1, z_2): x_1 = z_1 x_2 + z_2 \wedge 0 \leqq z_2 < x_2$$

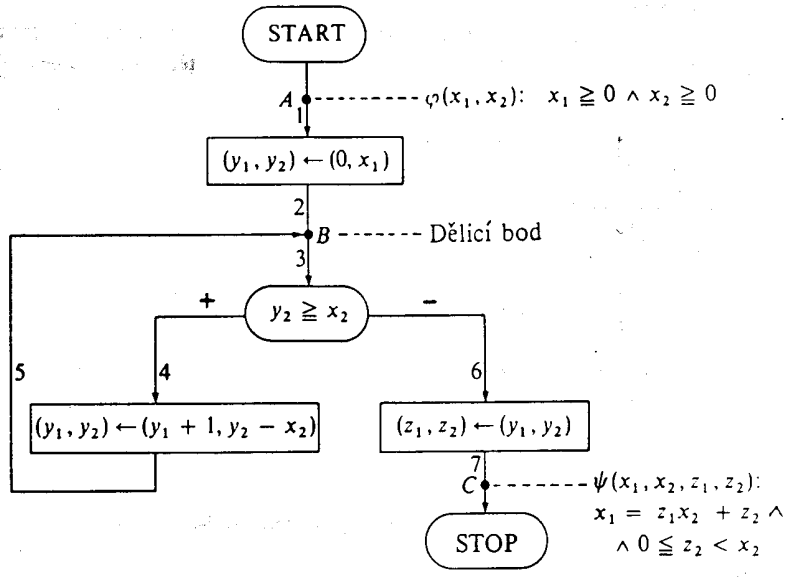
(který v podstatě definuje celočíselné dělení). Potom ukážeme, že program končí pro

$$\varphi'(x_1, x_2): x_1 \geqq 0 \wedge x_2 > 0.$$

Poněvadž tím prokážeme, že program je parciálně korektní vzhledem k φ a ψ a končí pro φ', máme tak dokázáno, že je totálně korektní vzhledem k φ' a ψ. Všimněme si rozdílu mezi φ a φ': v případě, že nás zajímá ukončení programu, je nutné vyloučit případ $x_2 = 0$ (pokud je $x_2 = 0$, program nekončí).

Parciální korektnost

Vyznačme v předloženém programu u příkazů START a STOP body A a C (viz obr. 3.2) a přiřaďme bodu A vstupní predikát φ a bodu C výstupní predikát ψ. Hlavní problém při verifikaci programů představují cykly. Cykl v našem programu lze zvládnout tím, že program „rozdělíme" bodem B, který rozloží každou cestu programem do tří složek: první je cesta z A do B (hrany 1 a 2); druhá je cesta z B zpět do B (hrany 3, 4 a 5); třetí je cesta z B do C (hrany 3, 6 a 7). Označíme tyto tři cesty po řadě α (START), β (cykl) a γ (STOP). Všechny končící běhy programu musí nejprve sledovat cestu α, pak několikrát (příp. 0-krát) projít cyklem β a konečně uzavřít běh přechodem na cestu γ. Všechny běhy programu jsou v jistém smyslu „pokryty" těmito třemi cestami.



Obr. 3.2. Program pro dělení celých čísel (s vstupními a výstupními predikáty)

Abychom dokázali částečnou korektnost programu, stanovíme především predikát $p(x_1, x_2, y_1, y_2)$, popisující vztah mezi proměnnými v bodě B. Vhodný je pro náš účel predikát[1])

$$x_1 = y_1 x_2 + y_2 \wedge y_2 \geqq 0.$$

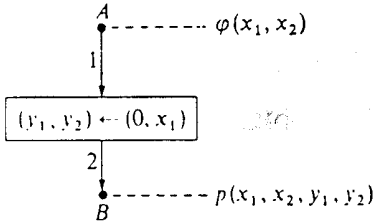
Po získání predikátu p se program rozpadá na tři cesty, z nichž každá začíná i končí predikátem. Parciální korektnost programu ověříme tak, že verifikujeme každou z těchto tří cest: tzn. pro každou z cest α, β, γ ukážeme, že je-li její vstupní predikát pravdivý pro jisté hodnoty vektorů $\bar{x}$ a $\bar{y}$, je po provedení jejích příkazů pravdivý i její výstupní predikát pro nové hodnoty vektorů $\bar{x}$ a $\bar{y}$.[2])

Verifikace cesty α spočívá v důkazu, že $p(x_1, x_2, y_1, y_2)$ je *pravda* při prvním vstupu do cyklu (za předpokladu, že je splněn vstupní predikát celého programu).[3]) K verifikaci cesty β je třeba ukázat, že $p(x_1, x_2, y_1, y_2)$ je *pravda*, když vstupujeme do cyklu po druhé, za předpokladu že týž predikát byl pravdivý při prvním vstupu do cyklu, dále, že je pravdivý při třetím vstupu za předpokladu, že byl pravdivý při druhém vstupu atd. K verifikaci cesty γ je třeba ukázat, že výstupní predikát cesty[4]) je pravdivý, je-li pravdivý predikát $p(x_1, x_2, y_1, y_2)$ při výstupu z cyklu. Jinak řečeno, verifikace cest α a β zaručí, že $p(x_1, x_2, y_1, y_2)$ je *pravda*, kdykoliv běh programu prochází dělicím bodem B[5]) – vždy pro průběžné hodnoty proměnných x_1, x_2, y_1 a y_2. Verifikace cesty γ pak zajistí, že výstupní predikát je pravdivý, jakmile program dojde do bodu C.

Zbývá verifikovat cesty α, β, γ. Verifikace každé cesty probíhá ve dvou krocích: Nejprve sestrojíme verifikační podmínku[6]) zapsanou pomocí daných predikátů a pak ji dokážeme.

Verifikační podmínku konstruujeme obvykle při zpětném procházení zvolené cesty tak, že si všímáme účinku všech příkazů, které na této cestě leží.

Cesta α. Uvažujme nejprve cestu α z obr. 3.2.



[1]) Smysl celého postupu bude objasněn vzápětí, pokud jde o volbu „vhodného" predikátu, musíme se zatím spolehnout na autora. Pozn. překl.

[2]) Poznamenejme, že jde opravdu o verifikaci, tj. o důkaz totální korektnosti, neboť žádná z cest sama o sobě již neobsahuje cykly. Pozn. překl.

[3]) Který je totožný se vstupním predikátem cesty α. Pozn. překl.

[4]) A tím i celého programu. Pozn. překl.

[5]) Predikát p je pravdivý v bodě B vždy, nezávisle na počtu průchodů cyklem; nazýváme ho proto někdy *invariant cyklu*. Pozn. překl.

[6]) *Verifikační podmínka* je predikát, který zachycuje požadovaný vztah mezi vstupním a výstupním predikátem (zvolené cesty). Formální definice bude podána v odst. 3.1.1. Pozn. překl.

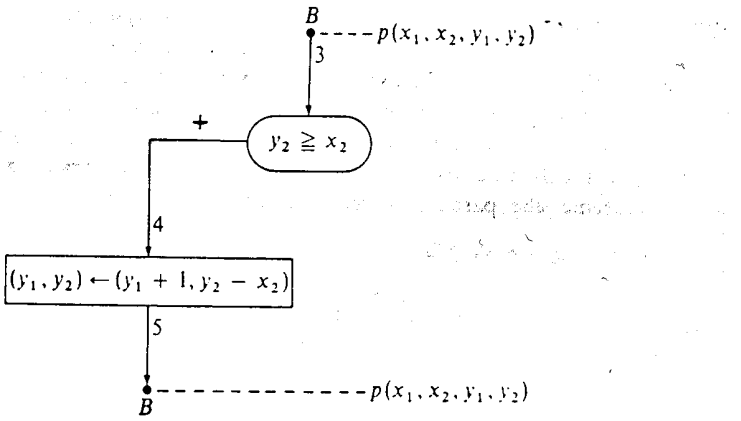
Položme si otázku: Co musí platit v bodě A, tj. před provedením příkazu $(y_1, y_2) \leftarrow (0, x_1)$, aby po jeho provedení platilo $p(x_1, x_2, y_1, y_2)$? Zřejmě je třeba, aby platilo $p(x_1, x_2, 0, x_1)$, tedy aby byl pravdivý predikát, vzniklý z $p(x_1, x_2, y_1, y_2)$ substitucí hodnoty 0 za všechny výskyty proměnné y_1 a x_1 za všechny výskyty proměnné y_2.[1])
Verifikační podmínka cesty tedy je

$$\varphi(x_1, x_2) \supset p(x_1, x_2, 0, x_1),$$

tj.

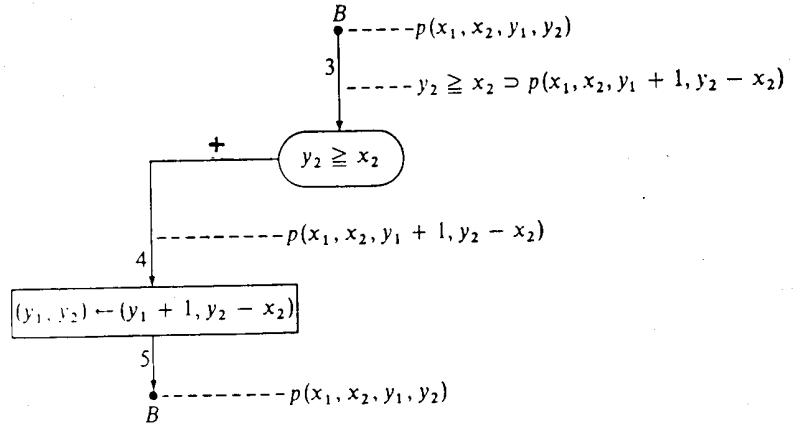
$$[x_1 \geqq 0 \wedge x_2 \geqq 0] \supset [x_1 = 0 \cdot x_2 + x_1 \wedge x_1 \geqq 0].$$

Cesta β. Při sestrojování verifikační podmínky cesty β musíme vzít v úvahu testovací příkaz; při sledování cesty β tento příkaz vždy shledá, že $y_2 \geqq x_2$, jak je vidět na tomto výseku programu:



Analogicky jako v předchozím případě zjistíme, že při procházení hranou 4 musí být pravdivý predikát $p(x_1, x_2, y_1 + 1, y_2 - x_2)$. Testovací příkaz sám o sobě nemění hodnoty proměnných, přesto však jeho provedení přináší cennou informaci: je zřejmé, že při procházení hranou 4 je vždy $y_2 \geqq x_2$. Položme si nyní vzhledem k testovacímu příkazu tutéž otázku, kterou jsme zodpověděli (již dvakrát) pro případ přiřazovacího příkazu: Co musí platit při procházení hranou 3, aby – kdykoliv je sledována větev „+", tj. kdykoliv $y_2 \geqq x_2$ – predikát $p(x_1, x_2, y_1 + 1, y_2 - x_2)$ byl pravdivý při průchodu hranou 4? Odpověď opět není složitá: Je třeba, aby $y_2 \geqq x_2 \supset p(x_1, x_2, y_1 + 1, y_2 - x_2)$. Úplnou analýzou cesty β tedy dostáváme:

[1]) Po provedení uvažovaného příkazu mají totiž oba predikáty týž význam. Pozn. překl.



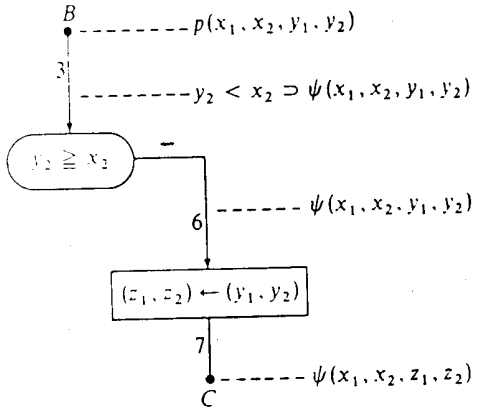
Celkem je tedy třeba dokázat verifikační podmínku

$$p(x_1, x_2, y_1, y_2) \supset [y_2 \geqq x_2 \supset p(x_1, x_2, y_1 + 1, y_2 - x_2)],$$

neboli

$$p(x_1, x_2, y_1, y_2) \wedge y_2 \geqq x_2 \supset p(x_1, x_2, y_1 + 1, y_2 - x_2)^1).$$

Cesta γ. Obdobnou analýzou cesty γ dostáváme:



Je tedy třeba dokázat verifikační podmínku

$$p(x_1, x_2, y_1, y_2) \supset [y_2 < x_2 \supset \psi(x_1, x_2, y_1, y_2)],$$

neboli

$$p(x_1, x_2, y_1, y_2) \wedge y_2 < x_2 \supset \psi(x_1, x_2, y_1, y_2).$$

[1] V této kapitole používáme konvence, podle níž výraz tvaru $A_1 \wedge A_2 \supset B$ zastupuje formuli $(A_1 \wedge A_2) \supset B$ a obecněji výraz $A_1 \wedge A_2 \wedge \ldots \wedge A_n \supset B$ formuli $(A_1 \wedge A_2 \wedge \ldots \wedge A_n) \supset B$.

Sestavili jsme tedy celkem tyto tři verifikační podmínky

$$\varphi(x_1, x_2) \supset p(x_1, x_2, 0, x_1), \qquad (\alpha)$$
$$p(x_1, x_2, y_1, y_2) \wedge y_2 \geqq x_2 \supset p(x_1, x_2, y_1 + 1, y_2 - x_2), \qquad (\beta)$$
$$p(x_1, x_2, y_1, y_2) \wedge y_2 < x_2 \supset \psi(x_1, x_2, y_1, y_2), \qquad (\gamma)$$

kde

$\varphi(x_1, x_2)$ je $x_1 \geqq 0 \wedge x_2 \geqq 0$,
$p(x_1, x_2, y_1, y_2)$ je $x_1 = y_1 x_2 + y_2 \wedge y_2 \geqq 0$,
$\psi(x_1, x_2, z_1, z_2)$ je $x_1 = z_1 x_2 + z_2 \wedge 0 \leqq z_2 < x_2$.

Čtenář se může sám přesvědčit, že pro libovolná celá čísla x_1, x_2, y_1 a y_2 jsou tyto verifikační podmínky pravdivé: program je tedy parciálně korektní vzhledem k φ a ψ.

Ukončení

Dosud jsme verifikovali náš program pouze parciálně: Dokázali jsme, že kdykoliv běh programu skončí (tj. provede se příkaz zastavení), výstupní predikát je pravdivý; vůbec jsme však nedokázali, že program skutečně skončí (že necykluje). K verifikaci programu pro dělení celých čísel je třeba ještě dokázat i tento fakt.

Ukážeme, že uvažovaný program končí pro každou dvojici vstupních hodnot x_1, x_2, pro kterou platí

$$\varphi'(x_1, x_2): x_1 \geqq 0 \wedge x_2 > 0$$

(opět připomínáme, že program nekončí, je-li $x_2 = 0$). Nejprve dokážeme, že predikát

$$q(x_1, x_2, y_1, y_2): y_2 \geqq 0 \wedge x_2 > 0$$

má tu vlastnost, že je pravdivý při každém průchodu programu bodem B. K tomu stačí dokázat tyto dvě verifikační podmínky:

$$\varphi'(x_1, x_2) \supset q(x_1, x_2, 0, x_1), \qquad (\alpha)$$
$$q(x_1, x_2, y_1, y_2) \wedge y_2 \geqq x_2 \supset q(x_1, x_2, y_1 + 1, y_2 - x_2), \qquad (\beta)$$

tj.

$$(x_1 \geqq 0 \wedge x_2 > 0) \supset (x_1 \geqq 0 \wedge x_2 > 0),$$
$$(y_2 \geqq 0 \wedge x_2 > 0 \wedge y_2 \geqq x_2) \supset (y_2 - x_2 \geqq 0 \wedge x_2 > 0).$$

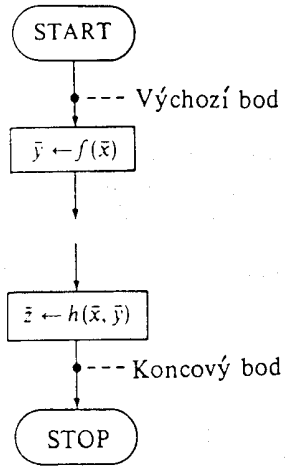
Obě podmínky jsou zjevně pravdivé.

Poněvadž máme nyní zaručeno, že v bodě B je vždy $x_2 > 0$, hodnota y_2 se postupně (jak procházíme cykly) snižuje. Máme také zaručeno, že $y_2 \geqq 0$, kdykoliv je program v bodě B. Klesající posloupnost kladných celých čísel samozřejmě nemůže být nekonečná – není proto ani možné, aby program donekonečna probíhal cyklem; jinak řečeno, program musí skončit. Ideu obsaženou v tomto důkazu zobecníme do metody důkazu ukončení programů v odst. 3.1.2.

3.1.1. Parciální korektnost

Zobecníme postup, s nímž jsme se právě seznámili na příkladě. Předpokládejme, že je dán program P, vstupní predikát φ a výstupní predikát ψ; při důkazu parciální korektnosti programu P vzhledem k φ a ψ postupujeme takto:

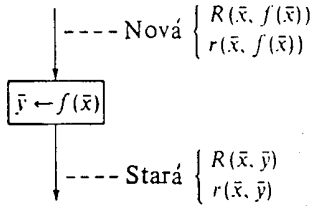
Krok 1. Dělicí body. V prvním kroku rozdělíme cykly programu vhodně volenými body na hranách diagramu, tzv. *dělicími body* tak, aby každý cykl obsahoval alespoň jeden dělicí bod. K množině takto získaných bodů přidáme *výchozí bod*, umístěný na hraně vycházející ze symbolu START a *koncové body*, ležící na hranách vedoucích k symbolům STOP (jeden bod na každé takové hraně).

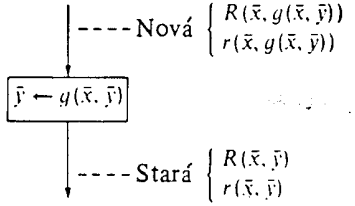


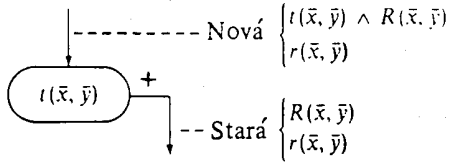
Zabýváme se pouze cestami, které začínají a končí v dělicích bodech a které neobsahují žádné další dělicí body. Každá taková cesta je konečná, poněvadž každý cykl obsahuje dělicí bod a celkový počet cest je také konečný. Nechť α je cesta vedoucí z děliciho bodu i do bodu j.[1]) V další diskusi používáme označení $R_\alpha(\bar{x}, \bar{y})$ pro predikát, udávající podmínku pro to, aby program prošel zvolenou cestou a dále označení $r_\alpha(\bar{x}, \bar{y})$ pro popis změny hodnot vektoru $\bar{y}$, vyvolané během průchodu programu cestou α. $R_\alpha(\bar{x}, \bar{y})$ je predikát nad $D_{\bar{x}} \times D_{\bar{y}}$ a $r_\alpha(\bar{x}, \bar{y})$ je funkce zobrazující $D_{\bar{x}} \times D_{\bar{y}}$ do $D_{\bar{y}}$. Oba výrazy jsou zapsány pomocí funkcí a predikátů (testů) použitých v příkazech cesty α.

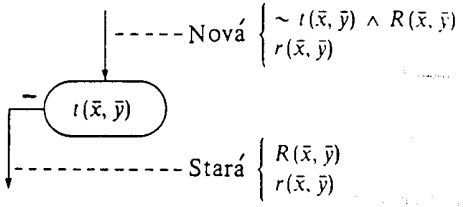
Jednoduchou metodou pro určení R a r je *metoda zpětných substitucí*. Položme nejprve $R(\bar{x}, \bar{y})$ rovno T a $r(\bar{x}, \bar{y})$ rovno $\bar{y}$; oba výrazy vztahujeme k dělicímu bodu j. Dále konstruujeme postupně „nová“ R a r pomocí „starých“ R a r tak, že se pohybujeme „pozpátku“ po cestě α směrem k bodu i. Poslední R a r, která takto získáme v bodě i jsou hledaná R a r. Pravidla pro konstrukci „nových“ R a r pomocí „starých“ R a r jsou:

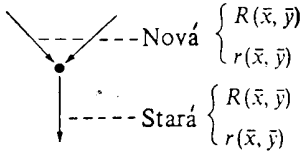
[1]) Přitom může být $i = j$, jak tomu bylo i v uvedeném příkladě.



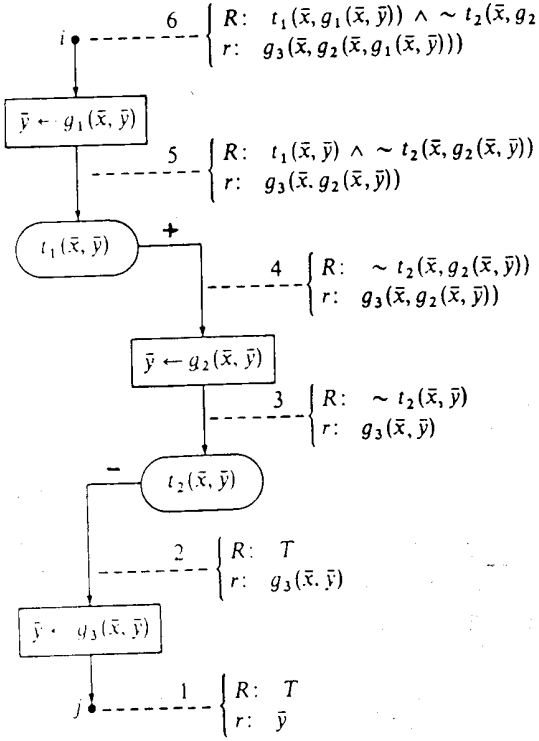








Podívejme se, jak pracuje metoda zpětných substitucí v případě cesty α z obr. 3.3. Začneme s hodnotou T v roli R a hodnotou $\bar{y}$ v roli r v dělicím bodě j



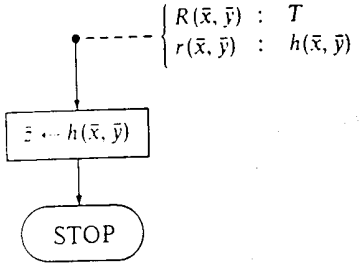
Obr. 3.3. Metoda zpětných substitucí

a postupujeme zpět k bodu i, kde nakonec dostaneme

$R_i(\bar{x}, \bar{y})$: $t_1(\bar{x}, g_1(\bar{x}, \bar{y})) \wedge \sim t_2(\bar{x}, g_2(\bar{x}, g_1(\bar{x}, \bar{y})))$,

$r_i(\bar{x}, \bar{y})$: $g_3(\bar{x}, g_2(\bar{x}, g_1(\bar{x}, \bar{y})))$.

Ve speciálním případě, kdy j je koncový bod, za r volíme $\bar{z}$ a za R volíme T; při prvním kroku zpět obdržíme



Ve speciálním případě, kdy i je výchozí bod, nevyskytuje se ani v $R_i(\bar{x}, \bar{y})$ ani v $r_i(\bar{x}, \bar{y})$ vektor $\bar{y}$; R_i je predikát nad $D_{\bar{x}}$ a r_i je funkce zobrazující $D_{\bar{x}}$ do $D_{\bar{y}}$.

Krok 2. Induktivní podmínky. V dalším kroku přiřadíme každému dělicímu bodu i daného programu predikát $p_i(\bar{x}, \bar{y})$ (nazývaný často *induktivní podmínka*), který by měl charakterizovat vztah mezi proměnnými v tomto bodě: p_i má tedy mít tu vlastnost, že kdykoliv program prochází bodem i, $p_i(\bar{x}, \bar{y})$ je *pravda* pro průběžné hodnoty $\bar{x}, \bar{y}$ vstupního vektoru a vektoru programu. K výchozímu bodu je takto přiřazen vstupní predikát $\varphi(\bar{x})$, ke všem koncovým bodům výstupní predikát $\psi(\bar{x}, \bar{z})$.[1])

Krok 3. Verifikační podmínky. Třetí krok spočívá v tom, že ke každé cestě vedoucí z dělicího bodu i do j sestrojíme *verifikační podmínku*

$$\forall \bar{x}\, \forall \bar{y}[p_i(\bar{x}, \bar{y}) \wedge R_\alpha(\bar{x}, \bar{y}) \supset p_j(\bar{x}, r_\alpha(\bar{x}, \bar{y}))].$$

Tato podmínka konstatuje, že kdykoliv induktivní podmínka p_i je *pravda* pro jisté hodnoty vektorů $\bar{x}$ a $\bar{y}$ (takové, že když s nimi zahájíme výpočet v bodě i, běh programu bude skutečně sledovat cestu α), je p_j *pravda* pro nové hodnoty vektorů $\bar{x}$ a $\bar{y}$ po průchodu programu cestou α.

Ve speciálním případě, že j je koncový bod, verifikační podmínka je

$$\forall \bar{x}\, \forall \bar{y}[p_i(\bar{x}, \bar{y}) \wedge R_\alpha(\bar{x}, \bar{y}) \supset \psi(\bar{x}, r_\alpha(\bar{x}, \bar{y}))]$$

a v případě, že i je výchozí bod, verifikační podmínka je

$$\forall \bar{x}[\varphi(\bar{x}) \wedge R_\alpha(\bar{x}) \supset p_j(\bar{x}, r_\alpha(\bar{x}))].$$

V závěrečném kroku dokážeme, že všechny takto získané verifikační podmínky jsou pravdivé. Jakmile to učiníme, máme zajištěnu pravdivost induktivní podmínky přiřazené k libovolnému dělicímu bodu pro hodnoty $\bar{x}$ a $\bar{y}$, s nimiž běh programu daným bodem prochází.[2]) Speciálně je zajištěno, že kdykoliv program dospěje k závěrečnému příkazu, je $\psi(\bar{x}, \bar{z})$ *pravda* pro průběžné hodnoty $\bar{x}$ a $\bar{z}$. Jinak řečeno, ověření verifikačních podmínek dokazuje parciální korektnost programu P vzhledem k φ a ψ.

Celý postup shrneme do tvaru tvrzení.

Věta 3.1 (Metoda induktivních podmínek – Floyd).

Pro daný program P, vstupní predikát $\varphi(\bar{x})$ a výstupní predikát $\psi(\bar{x}, \bar{z})$, sestrojte postupně

(1) *dělicí body všech cyklů programu;*

(2) *vhodné induktivní podmínky;*

(3) *verifikační podmínky.*

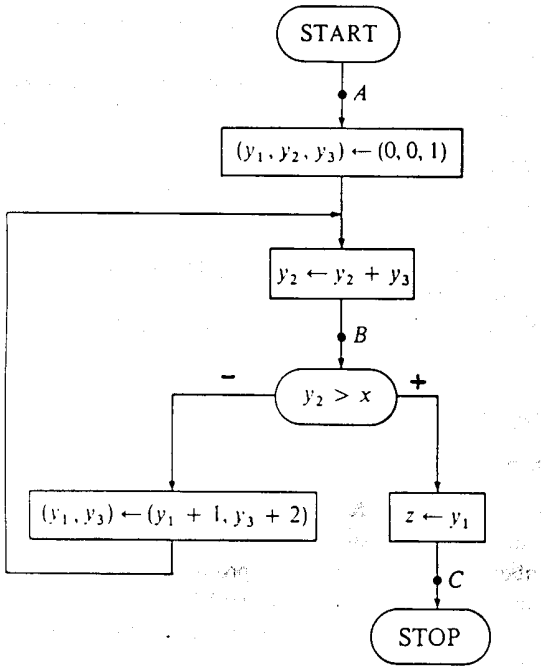
[1]) Všimněme si, že zatímco pro konstrukci predikátů R_α a funkcí r_α z předchozího kroku předložil autor jistý návod, o způsobu získávání induktivních podmínek nehovoří (srov. i s pozn.[1]) na str. 162). Později pochopíme, že to není náhoda či nedostatek výkladu. Pozn. překl.

[2]) Induktivní podmínky jsou tedy invarianty ve smyslu pozn.[5]) na str. 162. Pozn. překl.

Jsou-li všechny verifikační podmínky pravdivé, je program P parciálně korektní vzhledem k φ a ψ.

Všechny kroky tohoto postupu je možné realizovat mechanickými postupy s jedinou výjimkou, kterou tvoří krok 2: volba vhodných induktivních podmínek vyžaduje hlubokou znalost funkce a smyslu celého programu.

Ilustrujme metodu induktivních podmínek na několika příkladech.



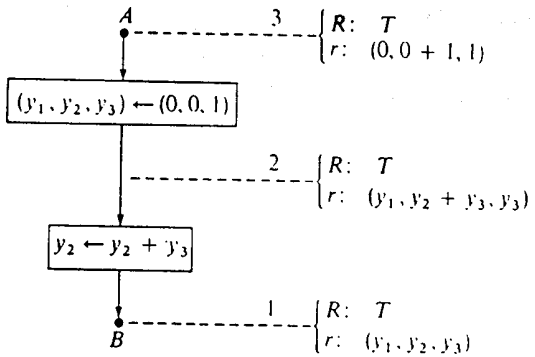
Obr. 3.4. Program P_1 pro výpočet $z = [\sqrt{x}]$

Příklad 3.1

Program P_1 z obr. 3.4 pracuje nad oborem celých čísel a pro každé přirozené číslo x spočítá $z = [\sqrt{x}]$, tj. výsledná hodnota proměnné z je největší přirozené číslo k takové, že $k \leqq \sqrt{x}$. Metoda výpočtu je založena na vztahu: $1 + 3 + 5 \ldots + (2n + 1) = (n + 1)^2$, který platí pro všechna $n \geqq 0$: n je zapamatováno jako hodnota proměnné y_1, liché číslo $2n + 1$ jako hodnota y_3 a součet $1 + 3 + 5 + \ldots + (2n + 1)$ jako hodnota y_2.

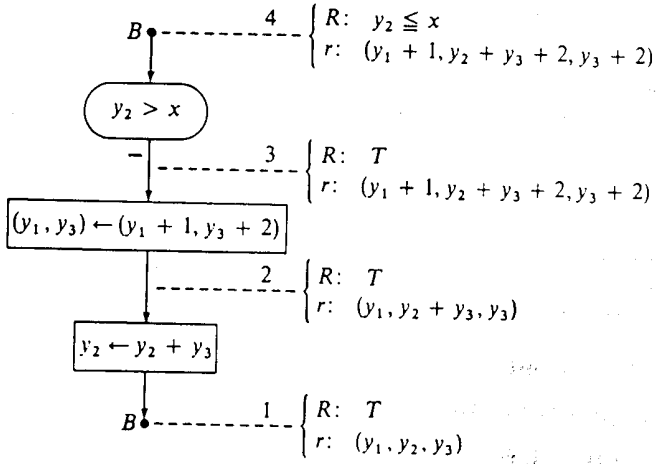
Dokážeme parciální korektnost programu vzhledem k vstupnímu predikátu $\varphi(x)$: $x \geqq 0$ a výstupnímu predikátu $\psi(x, z)$: $z^2 \leqq x < (z + 1)^2$. Volbou dělicího bodu B nejprve „přetneme" jediný cykl programu a dostaneme tak tři cesty, které je třeba vyšetřit. Metodou zpětných substitucí získáme pro každou z nich predikát R a funkci r:

Cesta α (z A do B):



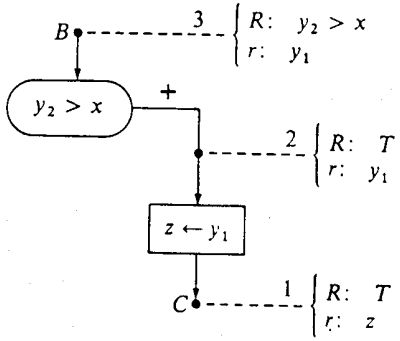
R_α je tedy T a r_α je $(0, 1, 1)$.

Cesta β (z B do B):



R_β je tedy $y_2 \leqq x$, r_β je $(y_1 + 1, y_2 + y_3 + 2, y_3 + 2)$.

Cesta γ (z B do C):

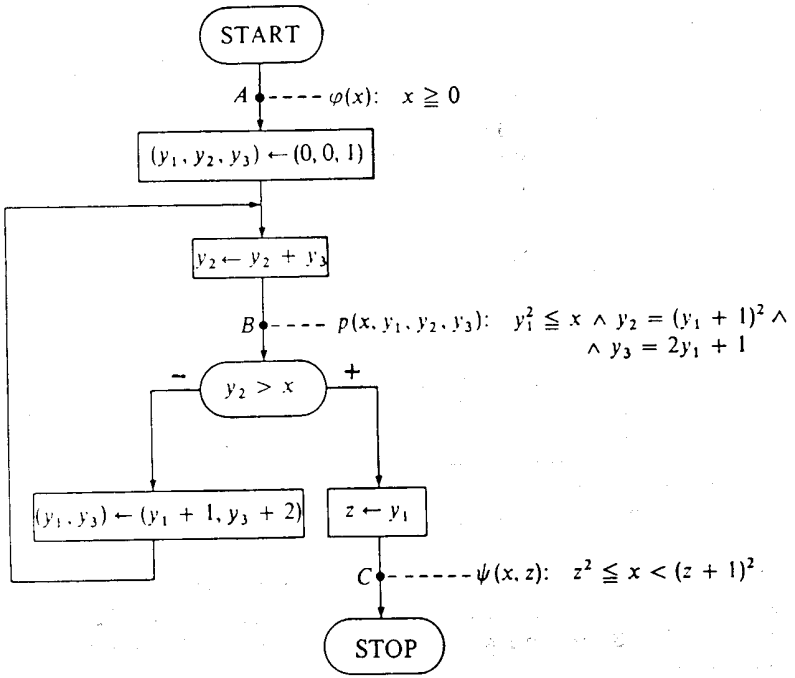


R_y je tedy $y_2 > x$, r_y je y_1.

Abychom mohli přistoupit k vlastnímu důkazu, je třeba přiřadit bodu B induktivní podmínku. Zvolme

$$p(x, y_1, y_2, y_3): y_1^2 \leqq x \wedge y_2 = (y_1 + 1)^2 \wedge y_3 = 2y_1 + 1.$$

Kromě toho přiřaďme výchozímu bodu A vstupní predikát $\varphi(x)$: $x \geqq 0$ a koncovému bodu C (viz obr. 3.5) výstupní predikát $\psi(x, z)$: $z^2 \leqq x < (z + 1)^2$.



Obr. 3.5. Program P_1 pro výpočet $z = [\sqrt{x}]$ (parciální korektnost)

Jde tedy o to, dokázat tři verifikační podmínky, které (pro libovolná celá čísla x, y_1, y_2 a y_3) tvrdí:

1. Pro cestu α je $[\varphi(x) \wedge T] \supset p(x, 0, 1, 1)$, tj.

$$x \geqq 0 \supset [0^2 \leqq x \wedge 1 = (0 + 1)^2 \wedge 1 = 2 \cdot 0 + 1].$$

2. Pro cestu β je $[p(x, y_1, y_2, y_3) \wedge y_2 \leqq x] \supset p(x, y_1 + 1, y_2 + y_3 + 2, y_3 + 2)$, tj.

$$[y_1^2 \leqq x \wedge y_2 = (y_1 + 1)^2 \wedge y_3 = 2y_1 + 1 \wedge y_2 \leqq x] \supset$$
$$\supset [(y_1 + 1)^2 \leqq x \wedge y_2 + y_3 + 2 = (y_1 + 2)^2 \wedge y_3 + 2 = 2(y_1 + 1) + 1].$$

3. Pro cestu γ je

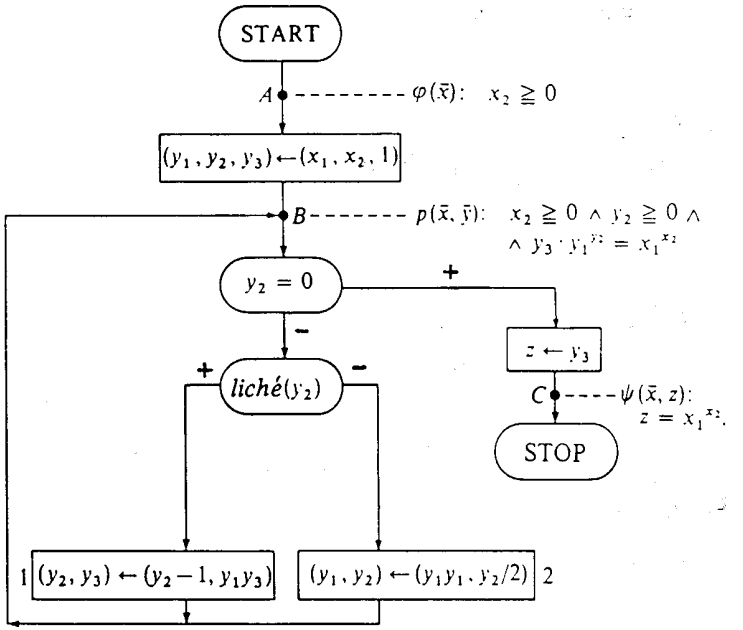
$$[p(x, y_1, y_2, y_3) \wedge y_2 > x] \supset \psi(x, y_1),$$

tj.

$$[y_1^2 \leqq x \wedge y_2 = (y_1 + 1)^2 \wedge y_3 = 2y_1 + 1 \wedge y_2 > x] \supset y_1^2 \leqq x < (y_1 + 1)^2.$$

Poněvadž všechny tři verifikační podmínky jsou pravdivé[1]), je daný program P_1 parciálně korektní vzhledem k vstupnímu predikátu $x \geqq 0$ a výstupnímu predikátu $z^2 \leqq x < (z + 1)^2$.

■



Obr. 3.6. Program P_2 pro výpočet $z = x_1^{x_2}$ (parciální korektnost)

Příklad 3.2

Program P_2 z obr. 3.6 pracuje nad oborem celých čísel a pro každé celé číslo x_1 a přirozené číslo x_2 spočítá $z = x_1^{x_2}$ (0^0 je definitoricky rovno 1). Výpočet se opírá o vzorce ($y_2 \geqq 0$):

$y_1^{y_2} = y_1 \cdot y_1^{y_2 - 1}$ pro y_2 liché,
$y_1^{y_2} = (y_1 \cdot y_1)^{y_2/2}$ pro y_2 sudé;

[1]) Ověření příslušných implikací je ve všech příkladech přenecháno na čtenáři jako elementární cvičení. Pozn. překl.

(které byly zvoleny na základě předpokladu, že se pracuje s číslem x_2 v dvojkovém zápisu).

Dokážeme, že program je parciálně korektní vzhledem k vstupnímu predikátu $\varphi(\bar{x})$: $x_2 \geqq 0$ a výstupnímu predikátu $\psi(\bar{x}, z)$: $z = x_1^{x_2}$. Volbou děliciho bodu B přetneme oba cykly programu; přiřadíme k němu induktivní podmínku

$$p(\bar{x}, \bar{y}): x_2 \geqq 0 \wedge y_2 \geqq 0 \wedge y_3 \cdot y_1^{y_2} = x_1^{x_2}.$$

Je třeba dokázat, že pro všechna celá čísla x_1, x_2, y_1, y_2 a y_3 platí verifikační podmínky:

1. Pro cestu α (z A do B):

   $$\varphi(\bar{x}) \supset p(\bar{x}, x_1, x_2, 1),$$

   tj.

   $$x_2 \geqq 0 \supset [x_2 \geqq 0 \wedge x_2 \geqq 0 \wedge 1 \cdot x_1^{x_2} = x_1^{x_2}].$$

2. Pro cestu β_1 (z B do B přes příkaz 1):

   $$[p(\bar{x}, \bar{y}) \wedge y_2 \neq 0 \wedge \mathit{liché}(y_2)] \supset p(\bar{x}, y_1, y_2 - 1, y_1 y_3),$$

   tj.

   $$[x_2 \geqq 0 \wedge y_2 \geqq 0 \wedge y_3 \cdot y_1^{y_2} = x_1^{x_2} \wedge y_2 \neq 0 \wedge \mathit{liché}(y_2)] \supset$$
   $$\supset [x_2 \geqq 0 \wedge y_2 - 1 \geqq 0 \wedge (y_1 y_3) \cdot y_1^{y_2 - 1} = x_1^{x_2}].$$

3. Pro cestu β_2 (z B do B přes příkaz 2):

   $$[p(\bar{x}, \bar{y}) \wedge y_2 \neq 0 \wedge \mathit{sudé}(y_2)] \supset p(\bar{x}, y_1 y_1, y_2/2, y_3),$$

   tj.

   $$[x_2 \geqq 0 \wedge y_2 \geqq 0 \wedge y_3 \cdot y_1^{y_2} = x_1^{x_2} \wedge y_2 \neq 0 \wedge \mathit{sudé}(y_2)] \supset$$
   $$\supset [x_2 \geqq 0 \wedge y_2/2 \geqq 0 \wedge y_3 \cdot (y_1 y_1)^{y_2/2} = x_1^{x_2}].$$

4. Pro cestu γ (z B do C):

   $$[p(\bar{x}, \bar{y}) \wedge y_2 = 0] \supset \psi(\bar{x}, y_3),$$

   tj.

   $$[x_2 \geqq 0 \wedge y_2 \geqq 0 \wedge y_3 \cdot y_1^{y_2} = x_1^{x_2} \wedge y_2 = 0] \supset y_3 = x_1^{x_2}.$$

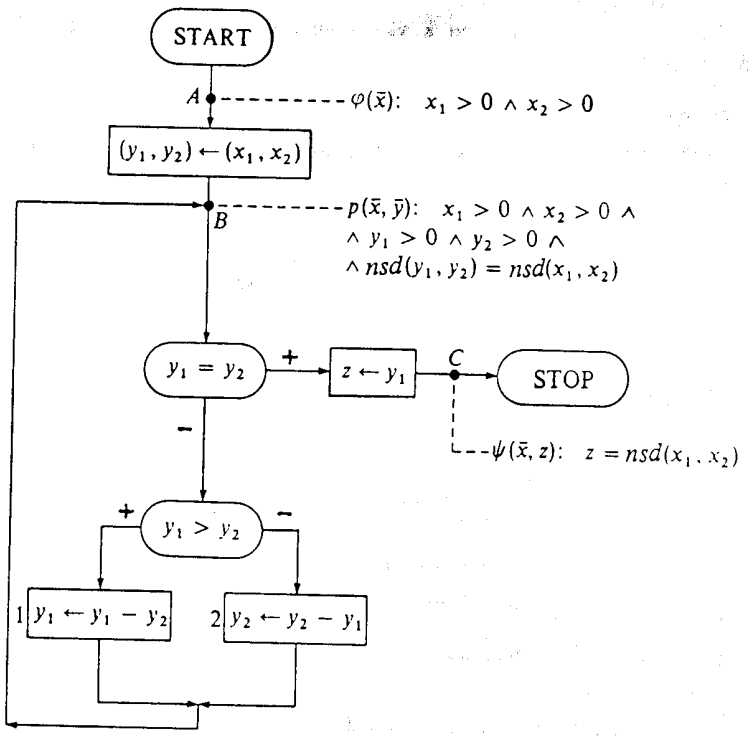
Poněvadž všechny čtyři verifikační podmínky jsou pravdivé, je daný program parciálně korektní vzhledem k vstupnímu predikátu $x_2 \geqq 0$ a výstupnímu predikátu $z = x_1^{x_2}$.

■

Příklad 3.3

Program P_3 z obr. 3.7 pracuje nad oborem celých čísel a ke každé dvojici kladných čísel x_1 a x_2 spočítá jejich největší společný dělitel $z = nsd(x_1, x_2)$; např. $nsd(14, 21) = 7$, $nsd(13, 21) = 1$. Výpočet se opírá o tyto vztahy:

je-li $y_1 > y_2$, je $nsd(y_1, y_2) = nsd(y_1 - y_2, y_2)$,
je-li $y_1 < y_2$, je $nsd(y_1, y_2) = nsd(y_1, y_2 - y_1)$,
je-li $y_1 = y_2$, je $nsd(y_1, y_2) = y_1 = y_2$.



Obr. 3.7. Program P_3 pro výpočet $z = nsd(x_1, x_2)$ (parciální korektnost)

Dokážeme, že program je parciálně korektní vzhledem k vstupnímu predikátu $\varphi(\bar{x})$: $x_1 > 0 \wedge x_2 > 0$ a výstupnímu predikátu $\psi(\bar{x}, z)$: $z = nsd(x_1, x_2)$. Volbou děliciho bodu B přetneme oba cykly programu a přiřadíme k němu induktivní podmínku

$$p(\bar{x}, \bar{y}): x_1 > 0 \wedge x_2 > 0 \wedge y_1 > 0 \wedge y_2 > 0 \wedge nsd(y_1, y_2) = nsd(x_1, x_2).$$

Verifikační podmínky, které je třeba dokázat (pro všechna celá čísla x_1, x_2, y_1, y_2) jsou:

1. Pro cestu α (z A do B):

   $$\varphi(\bar{x}) \supset p(\bar{x}, x_1, x_2),$$

   tj.

   $$[x_1 > 0 \wedge x_2 > 0] \supset [x_1 > 0 \wedge x_2 > 0 \wedge x_1 > 0 \wedge x_2 > 0 \wedge$$
   $$\wedge nsd(x_1, x_2) = nsd(x_1, x_2)].$$

2. Pro cestu β_1 (z B do B přes příkaz 1):

   $$[p(\bar{x}, \bar{y}) \wedge y_1 \neq y_2 \wedge y_1 > y_2] \supset p(\bar{x}, y_1 - y_2, y_2),$$

tj.

$$[x_1 > 0 \wedge x_2 > 0 \wedge y_1 > 0 \wedge y_2 > 0 \wedge nsd(y_1, y_2) = \\ = nsd(x_1, x_2) \wedge y_1 \neq y_2 \wedge y_1 > y_2] \supset [x_1 > 0 \wedge x_2 > 0 \wedge \\ \wedge y_1 - y_2 > 0 \wedge y_2 > 0 \wedge nsd(y_1 - y_2, y_2) = nsd(x_1, x_2)].$$

3. Pro cestu β_2 (z B do B přes příkaz 2)

$$[p(\bar{x}, \bar{y}) \wedge y_1 \neq y_2 \wedge y_1 \leqq y_2] \supset p(\bar{x}, y_1, y_2 - y_1),$$

tj.

$$[x_1 > 0 \wedge x_2 > 0 \wedge y_1 > 0 \wedge y_2 > 0 \wedge nsd(y_1, y_2) = \\ = nsd(x_1, x_2) \wedge y_1 \neq y_2 \wedge y_1 \leqq y_2] \supset [x_1 > 0 \wedge x_2 > 0 \wedge y_1 > 0 \wedge \\ \wedge y_2 - y_1 > 0 \wedge nsd(y_1, y_2 - y_1) = nsd(x_1, x_2)].$$

4. Pro cestu γ (z B do C):

$$[p(\bar{x}, \bar{y}) \wedge y_1 = y_2] \supset \psi(\bar{x}, y_1),$$

tj.

$$[x_1 > 0 \wedge x_2 > 0 \wedge y_1 > 0 \wedge y_2 > 0 \wedge nsd(y_1, y_2) = \\ = nsd(x_1, x_2) \wedge y_1 = y_2] \supset y_1 = nsd(x_1, x_2).$$

Poněvadž všechny čtyři verifikační podmínky jsou pravdivé, je daný program parciálně korektní vzhledem k vstupnímu predikátu $x_1 > 0 \wedge x_2 > 0$ a výstupnímu predikátu $z = nsd(x_1, x_2)$. ∎

3.1.2. Ukončení

Popíšeme nyní základní metodu důkazu ukončení programů. K tomu, abychom ukázali, že daný program nemůže procházet daným cyklem donekonečna, použijeme vhodným způsobem nějakou částečně uspořádanou množinu, v níž neexistuji nekonečné klesající posloupnosti. Nejčastěji se pro tyto účely používá množina přirozených čísel s jejím přirozeným uspořádáním $>$ (větší než). Poněvadž pro libovolné přirozené číslo n platí $n > \ldots > 2 > 1 > 0$, neexistují nekonečné klesající posloupnosti přirozených čísel. Všimněme si však, že nelze využít obdobně množinu všech celých čísel, neboť v ní existují nekonečné klesající posloupnosti – např. $n > \ldots > 2 > 1 > 0 > -1 > -2 > \ldots$. Naproti tomu není třeba vždy omezit se na množinu přirozených čísel; připomeňme proto několik obecných definic.

Částečně uspořádaná množina $(W, \prec)$ je dvojice sestávající z neprázdné množiny W a z binární relace $\prec$ nad W, splňující pro všechna $a, b, c \in W$ tyto podmínky:[1])

[1]) Podmínky 1 až 3 určují tzv. ostré částečné uspořádání. Termín „částečné uspořádání" obvykle označuje tranzitivní, antisymetrickou a reflexivní relaci. (Autor ho také v kap. 5 v tomto smyslu užívá; nezmiňuje se však, že ho již dříve užil v jiném smyslu.) Pozn. rec.

1. jestliže $a \prec b$ a $b \prec c$, pak $a \prec c$ (*tranzitivita*);
2. jestliže $a \prec b$, pak $b \not\prec a$ (*asymetrie*);
3. $a \not\prec a$ (*ireflexivita*).

Jak je zvykem, místo $a \prec b$ píšeme i $b \succ a$; $a \not\prec b$ znamená, že neplatí $a \prec b$. Částečné uspořádání nemusí být *úplné*, tj. mohou existovat $a, b \in W$ tak, že ani $a \prec b$, ani $b \prec a$.

Částečně uspořádanou množinu $(W, \prec)$, která neobsahuje nekonečné klesající posloupnosti $a_0 \succ a_1 \succ a_2 \succ \ldots$, nazveme *dobře fundovanou*[1]).

Příklady:

(a) Množina reálných čísel v intervalu $(0, 1)$ s přirozeným uspořádáním $<$ je částečně[2]) uspořádaná, není však dobře fundovaná (viz např. posloupnost $\frac{1}{2} > \frac{1}{3} > \frac{1}{4} > \ldots$).

(b) Množina všech celých čísel s přirozeným uspořádáním $<$ je částečně[2]) uspořádaná, není však dobře fundovaná (viz např. posloupnost $0 > -1 > -2 > \ldots$).

(c) Množina N všech přirozených čísel s přirozeným uspořádáním $<$ je dobře fundovaná.

(d) Množina Σ^* všech slov nad danou abecedou Σ s uspořádáním $\prec$ definovaným takto: $w_1 \prec w_2$, právě když w_1 je vlastní podslovo slova w_2[3]) je dobře fundovaná.

Předpokládejme nyní, že je dán program P a vstupní predikát φ. Důkaz toho, že program P končí pro φ, je možné vést tímto způsobem:

1. Zvolíme vhodnou dobře fundovanou množinu $(W, \prec)$.
2. Zvolíme dělicí body všech cyklů programu.
3. Ke každému dělicímu bodu i přiřadíme funkci $u_i(\bar{x}, \bar{y})$ zobrazující $D_{\bar{x}} \times D_{\bar{y}}$ do W, tj.

(*) $\forall \bar{x} \forall \bar{y} [u_i(\bar{x}, \bar{y}) \in W]$.

Jestliže pro každou cestu α z dělicího bodu i do dělicího bodu j takovou, že cesta neobsahuje další dělicí body a současně je tato cesta částí některého cyklu (tj. existuje i cesta z j do i) platí

(**) $\forall \bar{x} \forall \bar{y} \{\varphi(\bar{x}) \wedge R_\alpha(\bar{x}, \bar{y}) \supset [u_i(\bar{x}, \bar{y}) \succ u_j(\bar{x}, r_\alpha(\bar{x}, \bar{y}))]\}$,

pak program P končí pro φ.

[1]) Několik terminologických poznámek: Místo o částečně uspořádaných množinách se někdy hovoří prostě o uspořádaných množinách; *úplně* (též: *lineárně*) *uspořádaná množina* se též nazývá *řetězec*. Předpoklad neexistence nekonečné klesající posloupnosti se pak formuluje jako *podmínka konečnosti klesajících řetězců*; je ekvivalentní tzv. *minimální podmínce*. Běžná česká terminologie však nemá přímý název pro množiny, které tuto podmínku splňují; proto ve shodě s dříve zmíněnými zásadami překládáme anglický termín „well-founded set" doslovně. Pozn. překl.

[2]) Dokonce úplně. Pozn. překl.

[3]) To je právě když $w_2 = uw_1v$, kde alespoň jedno ze slov u, v je neprázdné. Pozn. překl.

Jinými slovy: Jestliže během libovolného výpočtu při přechodu od jednoho dělicího bodu k druhému nacházíme stále menší a menší prvky $u_i(\bar{x}, \bar{y})$ množiny W, pak ze skutečnosti, že ve W neexistuje nekonečná klesající posloupnost prvků plyne, že výpočet musí skončit.

Hlavním nedostatkem právě popsaného postupu je fakt, že podmínky (*) a (**) jsou silně omezující a jen stěží budeme schopni v praktických případech najít vhodný systém funkcí $u_i(\bar{x}, \bar{y})$, které je splňují. Problém je v požadavku, aby obě podmínky byly splněny pro všechna $\bar{x}$ a $\bar{y}$, zatímco obecně stačí, aby (*) a (**) bylo *pravda* jen pro ty hodnoty $\bar{x}$ a $\bar{y}$, kterých uvažované proměnné mohou skutečně nabýt během výpočtu. Nabízí se tu proto možnost využít vhodných induktivních podmínek a zesílit uvažovanou důkazovou metodu takto:

Krok 1. Zvolíme vhodnou množinu dělicích bodů, kterými přetneme všechny cykly programu. Ke každému dělicímu bodu i přiřadíme tzv. *„dobré" induktivní podmínky*[1]) $q_i(\bar{x}, \bar{y})$ tak, že:

(a) Pro každou cestu z výchozího bodu k dělicímu bodu j (takovou, že mezi nimi neleží žádný další dělicí bod) platí

$$\forall \bar{x}[\varphi(\bar{x}) \wedge R_z(\bar{x}) \supset q_j(\bar{x}, r_z(\bar{x}))].$$

(b) Pro každou cestu α z dělicího bodu i do dělicího bodu j (takovou, že mezi i a j neleží žádný další dělicí bod) platí

$$\forall \bar{x} \forall \bar{y}[q_i(\bar{x}, \bar{y}) \wedge R_\alpha(\bar{x}, \bar{y}) \supset q_j(\bar{x}, r_\alpha(\bar{x}, \bar{y}))].$$

Krok 2. Zvolíme vhodnou dobře fundovanou množinu $(W, \prec)$ a ke každému dělicímu bodu i přiřadíme částečnou funkci $u_i(\bar{x}, \bar{y})$ zobrazující $D_{\bar{x}} \times D_{\bar{y}}$ do W tak, že $u_i(\bar{x}, \bar{y})$ jsou tzv. *„dobré" funkce*, tj. že platí

$$\forall \bar{x} \forall \bar{y}[q_i(\bar{x}, \bar{y}) \supset u_i(\bar{x}, \bar{y}) \in W].$$

Krok 3. Dokážeme platnost *podmínky ukončení*, tj. dokážeme, že pro každou cestu α z dělicího bodu i do bodu j, takovou, že mezi i a j neleží žádný další dělicí bod a že cesta náleží nějakému cyklu; platí

$$\forall \bar{x} \forall \bar{y}\{q_i(\bar{x}, \bar{y}) \wedge R_\alpha(\bar{x}, \bar{y}) \supset [u_i(\bar{x}, \bar{y}) \succ u_j(\bar{x}, r_\alpha(\bar{x}, \bar{y}))]\}.$$

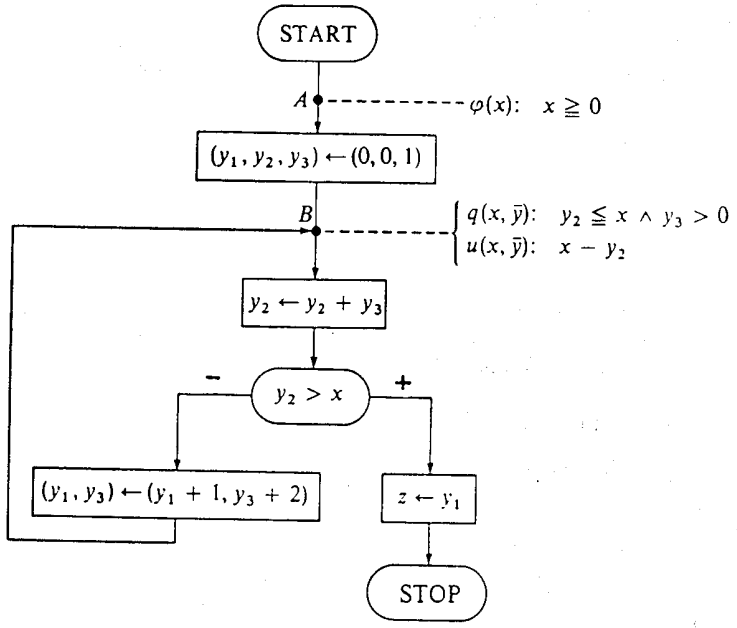
Celý postup shrneme do tvrzení:

Věta 3.2 (Metoda dobře fundovaných množin – Floyd).
Je-li dán program P a vstupní predikát $\varphi(\bar{x})$, sestrojte postupně

(1) *dělicí body všech cyklů programu a vhodné „dobré" induktivní podmínky;*
(2) *vhodnou dobře fundovanou množinu a „dobré" funkce;*
(3) *podmínky ukončení.*

Jsou-li všechny podmínky ukončení pravdivé, pak program P končí pro φ.

[1]) Poznamenejme, že volba dělicích bodů a dobrých induktivních podmínek nemusí být táž jako volba dělicích bodů a induktivních podmínek z důkazu parciální korektnosti uvažovaného programu. Pozn. překl.



Obr. 3.8. Program P_1 pro výpočet $z = [\sqrt{x}]$ (ukončení)

Příklad 3.4

Dokážeme, že program P_1 z př. 3.1 končí pro $\varphi(x)$: $x \geqq 0$. Zvolme dělicí bod B a induktivní podmínku $q(x, \bar{y})$: $y_2 \leqq x \wedge y_3 > 0$ tak, jak je ukázáno na obr. 3.8. Použijeme dobře fundované množiny $(N, <)$ přirozených čísel s přirozeným uspořádáním $<$. K dělicímu bodu B přiřadíme funkci $u(x, \bar{y})$: $x - y_2$.

Zajímají nás tyto dvě cesty:

Cesta α_1 (z A do B): R_{α_1} je T, r_{α_1} je $(0, 0, 1)$.

Cesta α_2 (z B do B): R_{α_2} je $y_2 + y_3 \leqq x$ a r_{α_2} je $(y_1 + 1, y_2 + y_3, y_3 + 2)$.

Důkaz rozčleníme do tří částí, v nichž ukážeme, že pro všechna celá čísla x, y_1, y_2 a y_3 platí:

1. $q(x, \bar{y})$ je dobrá induktivní podmínka.

Pro cestu α_1 je

$$\varphi(x) \supset q(x, 0, 0, 1),$$

tj.

$$x \geqq 0 \supset [0 \leqq x \wedge 1 > 0].$$

Pro cestu α_2 je

$$[q(x, y_1, y_2, y_3) \wedge y_2 + y_3 \leqq x] \supset q(x, y_1 + 1, y_2 + y_3, y_3 + 2).$$

tj.

$$[y_2 \leqq x \wedge y_3 > 0 \wedge y_2 + y_3 \leqq x] \supset [y_2 + y_3 \leqq x \wedge y_3 + 2 > 0].$$

2. $u(x, \bar{y})$ je dobrá funkce:

$$q(x, \bar{y}) \supset u(x, \bar{y}) \in N,$$

tj.

$$[y_2 \leqq x \wedge y_3 > 0] \supset x - y_2 \geqq 0.$$

3. Podmínka ukončení je splněna.

Pro cestu α_2 je

$$[q(x, \bar{y}) \wedge y_2 + y_3 \leqq x] \supset [u(x, y_1, y_2, y_3) > u(x, y_1 + 1, y_2 + y_3, y_3 + 2)],$$

tj.

$$[y_2 \leqq x \wedge y_3 > 0 \wedge y_2 + y_3 \leqq x] \supset [x - y_2 > x - (y_2 + y_3)]$$

(cestu α_1 neuvažujeme, protože není částí žádného cyklu).

Poněvadž všechna tři uvedená tvrzení jsou pravdivá, uvažovaný program končí pro všechna přirozená čísla x. ∎

Snadno se také ukáže, že program P_2 z př. 3.2 končí pro $\varphi(\bar{x})$: $x_2 \geqq 0$ [za $q_B(\bar{x}, \bar{y})$ vezměte $y_2 \geqq 0$ a nechť $u_B(\bar{x}, \bar{y})$ je y_2] a že program P_3 z př. 3.3 končí pro $\varphi(\bar{x})$: $x_1 > 0 \wedge x_2 > 0$ [za $q_B(\bar{x}, \bar{y})$ vezměte $y_1 > 0 \wedge y_2 > 0$ a nechť $u_B(\bar{x}, \bar{y})$ je $\max(y_1, y_2)$]. Rozebereme proto detailněji méně triviální příklad.

Příklad 3.5 (Knuth).

Program P_4 z obr. 3.9 spočítá také pro každou dvojici kladných čísel jejich největší společný dělitel $z = nsd(x_1, x_2)$. Ponecháme jako cvičení čtenáři, aby dokázal parciální korektnost programu P_4 vzhledem k vstupnímu predikátu $\varphi(\bar{x})$: $x_1 > 0 \wedge x_2 > 0$ a výstupnímu predikátu $\psi(\bar{x}, z)$: $z = nsd(x_1, x_2)$.[1]) Zde se soustředíme na důkaz toho, že P_4 končí pro φ.

Použijeme opět dobře fundované množiny $(N, <)$. Zvolíme dělicí body B a C a přiřadíme jim tyto predikáty a funkce:

$q_B(\bar{x}, \bar{y})$: $y_1 > 0 \wedge y_2 > 0$; $u_B(\bar{x}, \bar{y})$: y_1,

$q_C(\bar{x}, \bar{y})$: $y_1 > 0 \wedge y_2 > 0$; $u_C(\bar{x}, \bar{y})$: $y_1 + 2y_2$.

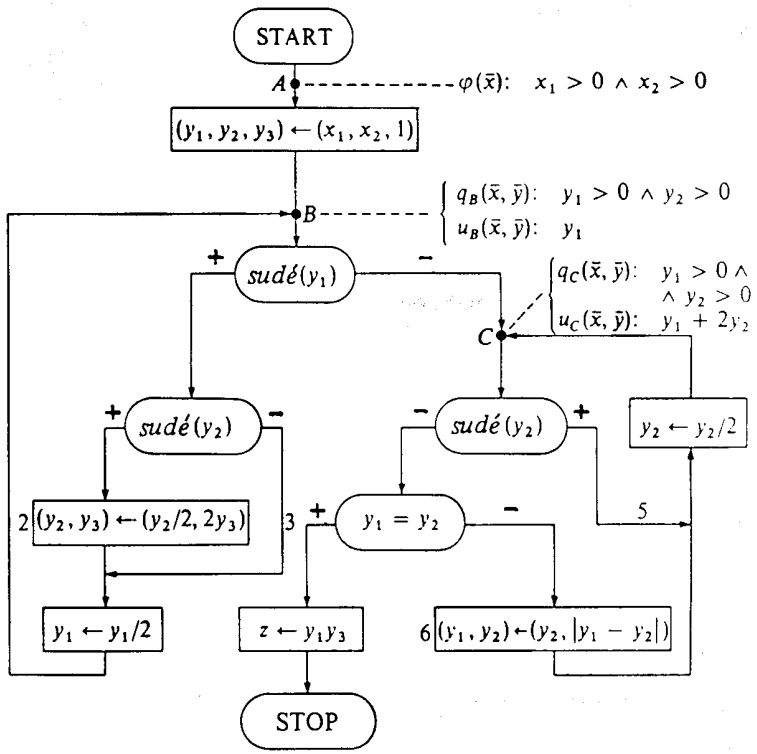
Zajímat nás bude celkem šest cest:

α_1 (z A do B): R_{α_1} je T, r_{α_1} je $(x_1, x_2, 1)$;

α_2 (z B do B přes příkaz 2): R_{α_2} je $sudé(y_1) \wedge sudé(y_2)$, r_{α_2} je $(y_1/2, y_2/2, 2y_3)$;

α_3 (z B do B přes příkaz 3): R_{α_3} je $sudé(y_1) \wedge liché(y_2)$, r_{α_3} je $(y_1/2, y_2, y_3)$;

[1]) Návod: Nechť $p_B(\bar{x}, \bar{y})$ je $x_1 > 0 \wedge x_2 > 0 \wedge y_1 > 0 \wedge y_3 . nsd(y_1, y_2) = nsd(x_1, x_2)$ a $p_C(\bar{x}, \bar{y})$ je $x_1 > 0 \wedge x_2 > 0 \wedge y_2 > 0 \wedge liché(y_1) \wedge y_3 . nsd(y_1, y_2) = nsd(x_1, x_2)$.



Obr. 3.9. Program P_4 pro výpočet $z = nsd(x_1, x_2)$ (ukončení)

α_4 (z B do C): R_{α_4} je $liché(y_1)$, r_{α_4} je (y_1, y_2, y_3);

α_5 (z C do C přes příkaz 5): R_{α_5} je $sudé(y_2)$, r_{α_5} je $(y_1, y_2/2, y_3)$;

α_6 (z C do C přes příkaz 6): R_{α_6} je $liché(y_2) \wedge y_1 \neq y_2$, r_{α_6} je $(y_2, |y_1 - y_2|, y_3)$.

Důkaz opět probíhá ve třech etapách: Pro všechna celá čísla x, y_1, y_2 a y_3 platí:

1. q_B a q_C jsou dobré induktivní podmínky. Dosadíme-li za q_B a q_C přiřazené predikáty, snadno dokážeme, že platí:

pro cestu α_1

$$\varphi(\bar{x}) \supset q_B(\bar{x}, x_1, x_2, 1);$$

pro cestu α_2

$$q_B(\bar{x}, \bar{y}) \wedge sudé(y_1) \wedge sudé(y_2) \supset q_B(\bar{x}, y_1/2, y_2/2, 2y_3);$$

pro cestu α_3

$$q_B(\bar{x}, \bar{y}) \wedge sudé(y_1) \wedge liché(y_2) \supset q_B(\bar{x}, y_1/2, y_2, y_3);$$

pro cestu α_4

$$q_B(\bar{x}, \bar{y}) \wedge liché(y_1) \supset q_C(\bar{x}, y_1, y_2, y_3);$$

pro cestu α_5

$$q_C(\bar{x}, \bar{y}) \wedge sudé(y_2) \supset q_C(\bar{x}, y_1, y_2/2, y_3);$$

pro cestu α_6

$$q_C(\bar{x}, \bar{y}) \wedge liché(y_2) \wedge y_1 \neq y_2 \supset q_C(\bar{x}, y_2, |y_1 - y_2|/2, y_3).$$

2. u_B a u_C jsou dobré funkce; opět lze snadno ukázat, že platí:

$$q_B(\bar{x}, \bar{y}) \supset u_B(\bar{x}, \bar{y}) \in N,$$

$$q_C(\bar{x}, \bar{y}) \supset u_C(\bar{x}, \bar{y}) \in N.$$

3. Podmínky ukončení jsou splněny.

Pro cestu α_2:

$$[q_B(x, y) \wedge sudé(y_1) \wedge sudé(y_2)] \supset [u_B(\bar{x}, y_1, y_2, y_3) > u_B(\bar{x}, y_1/2, y_2/2, 2y_3)].$$

Pro cestu α_3:

$$[q_B(\bar{x}, \bar{y}) \wedge sudé(y_1) \wedge liché(y_2)] \supset [u_B(\bar{x}, y_1, y_2, y_3) > u_B(\bar{x}, y_1/2, y_2, y_3)].$$

Pro cestu α_5:

$$[q_C(\bar{x}, \bar{y}) \wedge sudé(y_2)] \supset [u_C(\bar{x}, y_1, y_2, y_3) > u_C(\bar{x}, y_1, y_2/2, y_3)].$$

Pro cestu α_6:

$$[q_C(\bar{x}, \bar{y}) \wedge liché(y_2) \wedge y_1 \neq y_2] \supset [u_C(\bar{x}, y_1, y_2, y_3) > u_C(\bar{x}, y_2, |y_1 - y_2|/2, y_3)].$$

Také tato tvrzení jsou pravdivá, jak se snadno ověří dosazením přiřazených predikátů a funkcí za q_B, q_C, u_B a u_C; odtud plyne, že program P_4 končí pro každou dvojici kladných čísel x_1 a x_2. ∎

3.2. PROGRAMY S POLI

Pole je konstrukce sloužící k popisu většího počtu navzájem souvisejících proměnných. Chceme-li např. hovořit o 21 proměnných, není přehledné označit je po řadě písmeny A, B, C, ..., T, U; obvykle v takovém případě dáme přednost označení $S[0]$, $S[1]$, ..., $S[20]$, kde S považujeme za pole o 21 prvcích. Toto označení odpovídá použití indexů, běžnému z matematických textů: $S_0, S_1, \ldots, S_{20}$. Výraz typu $S[i + j]$ označuje tedy $(i + j)$-tý prvek $(0 \leqq i + j \leqq 20)$ uvažované množiny a závisí na momentální hodnotě výrazu $i + j$.

3.2.1. Parciální korektnost

V tomto odstavci dokážeme parciální korektnost několika programů, které užívají polí a rozebereme problémy, které zavedení pole přináší. Nebudeme však přitom řešit jeden specifický problém, který se týká programů s poli – nebudeme ověřovat, zda hodnota použitého indexu vždy leží v rozsahu pole. Tak např. použijeme-li v programu pole S s 21 prvky $S[0]$, $S[1]$, ..., $S[20]$ a přiřazovací příkaz $y \leftarrow S[i + j]$, bylo by třeba dokázat, že kdykoli dojde k provedení tohoto příkazu, je $0 \leqq i + j \leqq 20$, tj. průběžná hodnota indexu leží v rozsahu pole. Takový důkaz by bylo možno realizovat tím, že bychom přiřadili induktivní podmínku $0 \leqq i + j \leqq 20$ ke každé hraně směřující k uvedenému příkazu a dokázali příslušné verifikační podmínky.

Příklad 3.6

Program P_5 z obr. 3.10 uspořádá pole X sestávající z $n + 1$ $(n \geqq 0)$ reálných čísel $X[0]$, $X[1]$, ..., $X[n]$ tak, že výsledné hodnoty výstupního pole $Z[0]$, $Z[1]$, ..., $Z[n]$ budou tvořit rostoucí posloupnost. Prvky pole X jsou nejprve zapsány do pomocného pole S, ve kterém jsou pak prováděna potřebná přemístění. Všimněme si, že operace

$$(S[j], S[j + 1]) \leftarrow (S[j + 1], S[j])$$

zajišťuje výměnu čísel zapsaných v $S[j]$ a $S[j + 1]$. Na závěr jsou prvky pole S zapsány do pole Z.

Dokážeme, že (za předpokladu, že program končí):

1. Prvky pole Z jsou permutací prvků původního pole X. Tuto skutečnost zapíšeme výrazem $perm(X, Z, 0, n)$; predikát $perm(X, Z, k, l)$ obecněji tvrdí, že prvky $Z[k]$, $Z[k + 1]$, ..., $Z[l]$ jsou permutací prvků $X[k]$, $X[k + 1]$, ..., $X[l]$. (Predikát nabývá hodnoty *pravda*, kdykoliv $k \geqq l$.)

2. Prvky pole Z tvoří rostoucí posloupnost. Tuto skutečnost zapíšeme výrazem $usp(Z, 0, n)$; predikát $usp(Z, k, l)$ obecněji tvrdí, že prvky $Z[k]$, $Z[k + 1]$, ..., $Z[l]$ tvoří rostoucí posloupnost, tj. že $Z[k] \leqq Z[k + 1] \leqq \ldots \leqq Z[l]$. (Predikát nabývá hodnoty *pravda*, kdykoliv $k \geqq l$.)

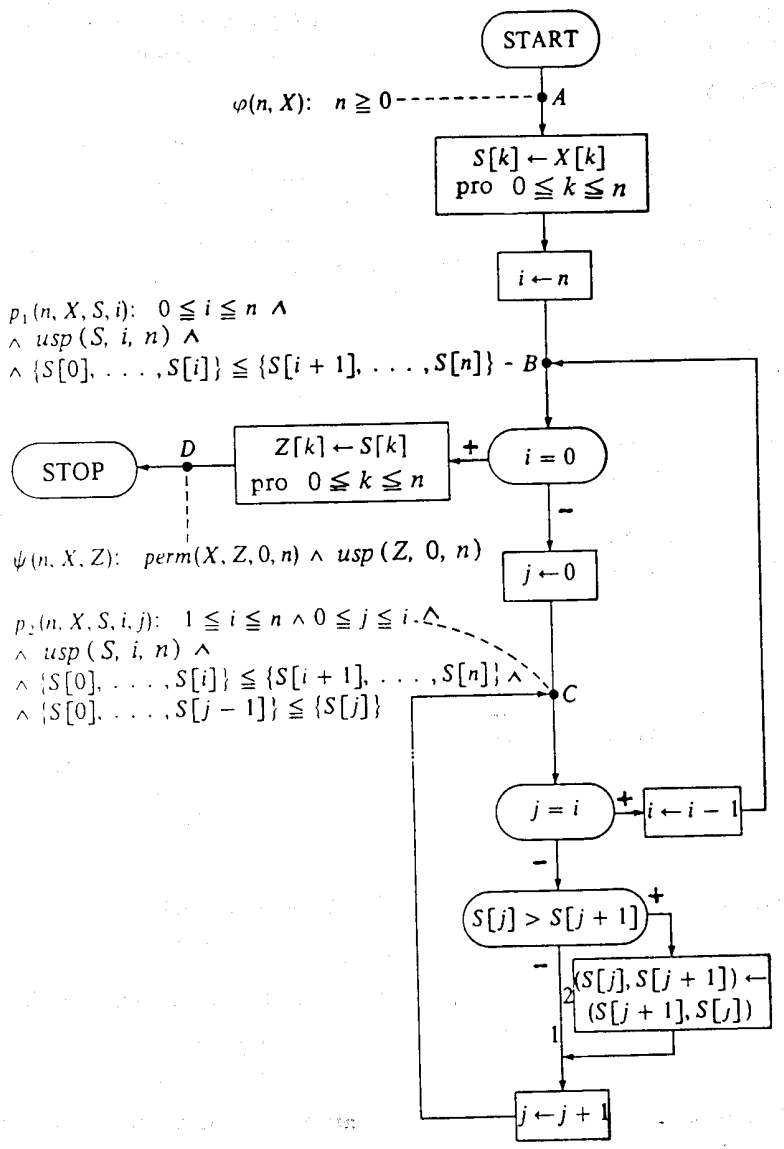
Dokazujeme tedy, že program je parciálně korektní vzhledem k vstupnímu predikátu

$$\varphi(n, X): \quad n \geqq 0$$

a výstupnímu predikátu

$$\psi(n, X, Z): \quad perm(X, Z, 0, n) \wedge usp(Z, 0, n).$$

Že $perm(X, Z, 0, n)$ je *pravda*, plyne okamžitě z toho, že jedinou používanou operací nad polem S je operace $(S[j], S[j + 1]) \leftarrow (S[j + 1], S[j])$, která mění pouze pořadí prvků v S.



Obr. 3.10. Program P_5 pro uspořádání prvků pole (parciální korektnost)

Abychom ukázali, že $usp(Z, 0, n)$ je *pravda*, zvolíme dělicí body B a C podle obr. 3.10 a přiřadíme k nim vhodné induktivní podmínky. Množinu všech prvků $S[m]$ takových, že $k \leqq m \leqq l$ označíme $\{S[k], \ldots, S[l]\}$ (tato množina je prázdná, jestliže $k > l$). Pro libovolné dvě množiny reálných čísel T_1 a T_2 pišme $T_1 \leqq T_2$, jestliže každý prvek z T_1 je menší nebo roven libovolnému prvku z T_2 (což je triviálně *pravda*, jsou-li množiny T_1 i T_2 prázdné).

Induktivní podmínky mohou být nyní vyjádřeny takto:

$p_1(n, X, S, i)$ v dělicím bodě B:

$$0 \leqq i \leqq n \wedge usp(S, i, n) \wedge \{S[0], \ldots, S[i]\} \leqq \{S[i + 1], \ldots, S[n]\};$$

$p_2(n, X, S, i, j)$ v dělicím bodě C:

$$1 \leqq i \leqq n \wedge 0 \leqq j \leqq i \wedge usp(S, i, n) \wedge \wedge \{S[0], \ldots, S[i]\} \leqq \{S[i + 1], \ldots, S[n]\} \wedge \{S[0], \ldots, S[j - 1]\} \leqq \{S[j]\}.$$

Verifikační podmínky, které je třeba dokázat, jsou:

1. Pro cestu α (z A do B):

$$\varphi(n, X) \supset p_1(n, X, X, n).$$

2. Pro cestu γ (z B do D):

$$[p_1(n, X, S, i) \wedge i = 0] \supset \psi(n, X, S).$$

3. Pro cestu β_1 (z B do C):

$$[p_1(n, X, S, i) \wedge i \neq 0] \supset p_2(n, X, S, i, 0).$$

4. Pro cestu β_2 (z C do C přes hranu 1):

$$[p_2(n, X, S, i, j) \wedge (S[j] \leqq S[j + 1]) \wedge j = i] \supset p_2(n, X, S, i, j + 1).$$

5. Pro cestu β_3 (z C do C přes příkaz 2):

$$[p_2(n, X, S, i, j) \wedge (S[j] > S[j + 1] \wedge j \neq i] \supset p_2(n, X, S^*, i, j + 1).$$

kde S^* označuje pole S po záměně hodnot $S[j]$ a $S[j + 1]$.

6. Pro cestu β_4 (z C do B):

$$[p_2(n, X, S, i, j) \wedge j = i] \supset p_1(n, X, S, i - 1).$$

Důkaz všech šesti tvrzení je snadný; ukažme např. platnost verifikační podmínky 6, která po příslušném rozpisu predikátů p_1 a p_2 zní:

$$[1 \leqq i \leqq n \wedge 0 \leqq j \leqq i \wedge usp(S, i, n) \wedge \wedge \{S[0], \ldots, S[i]\} \leqq \{S[i + 1], \ldots, S[n]\} \wedge \wedge \{S[0], \ldots, S[j - 1]\} \leqq \{S[j]\} \wedge j = i] \supset$$
$$\supset [0 \leqq i - 1 \leqq n \wedge usp(S, i - 1, n) \wedge \wedge \{S[0], \ldots, S[i - 1]\} \leqq \{S[i], \ldots, S[n]\}].$$

Především je zřejmé, že z předpokladu $1 \leqq i \leqq n$ vyplývá $0 \leqq i - 1 \leqq n$; z $usp(S, i, n)$ a $S[i-1] \leqq S[i]$ (je $S[j-1] \leqq S[j]$ a $j = i$) plyne $usp(S, i-1, n)$. Konečně z $\{S[0], \ldots, S[i]\} \leqq \{S[i+1], \ldots, S[n]\}$ a $\{S[0], \ldots, S[i-1]\} \leqq \{S[i]\}$ vyplývá $\{S[0], \ldots, S[i-1]\} \leqq \{S[i], \ldots, S[n]\}$.

■

Induktivní podmínky byly v př. 3.6 vyjádřeny celkem neformálním způsobem. K jejich zpřesnění můžeme využít aparátu, který nabízí predikátový počet; tak např. výraz

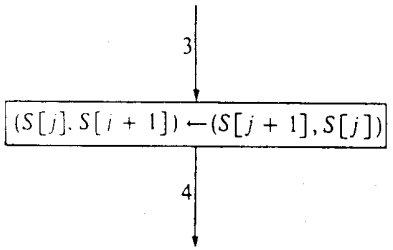
$$\{S[0], \ldots, S[j-1]\} \leqq \{S[j]\}$$

z podmínky p_2 můžeme formálněji vyjádřit ve tvaru

$$\forall k[0 \leqq k < j \supset S[k] \leqq S[j]].$$

Také verifikační podmínky byly vyjádřeny jen neformálně. Obecně je třeba věnovat zvláštní pozornost konstrukci verifikačních podmínek u programů, které ve svých přiřazovacích příkazech mění hodnotu některého prvku pole.

Vezměme např. přiřazovací příkaz, který zaměňuje hodnoty $S[j]$ a $S[j+1]$,



přičemž předpokládáme, že na hraně 4 je splněna podmínka

(*) $\forall k[0 \leqq k \leqq j \supset S[k] \leqq S[j+1]]$

(poznamenejme, že jde o konstrukci, vyskytující se ve verifikační podmínce 5 z př. 3.6). Konstrukce odpovídající podmínky přiřazené hraně 3 vyžaduje současnou výměnu $S[j]$ za $S[j+1]$ a $S[j+1]$ za $S[j]$. Tato substituce ovlivní hodnoty $S[k]$ a $S[j+1]$ v předložené podmínce (*): $S[k]$ bude nahrazeno výrazem[1])

$$if\ k = j\ then\ S[j+1]\ else\ (if\ k = j+1\ then\ S[j]\ else\ S[k]),$$

zatímco $S[j+1]$ je nahrazeno jednoduše přímo výrazem $S[j]$, protože je v předloženém tvrzení (*) uvedeno explicitně. Dostaneme tak

$$\forall k[0 \leqq k \leqq j \supset \\ \supset (if\ k = j\ then\ S[j+1]\ else\ (if\ k = j+1\ then\ S[j]\ else\ S[k])) \leqq S[j]].$$

[1]) Čtenář možná snadněji nahlédne přímo platnost výsledné podmínky (**) uvedené na konci tohoto rozboru. Pozn. překl.

Tento výraz lze uvážením dvou případů $k = j$ a $0 \leqq k < j$ (všimněme si, že z $0 \leqq k \leqq j$ vyplývá $k \neq j+1$) zjednodušit na výslednou podmínku

(**) $S[j+1] \leqq S[j] \wedge \forall k[0 \leqq k < j \supset S[k] \leqq S[j]]$.

Jiný možný přístup k práci s poli spočívá ve využití speciálních funkcí ω (*obsah*) a π (*přiřazení*) zavedených v práci [McCarthy 1962]. Pole, např. $S[0], S[1], \ldots$, je tu považováno za jedinou proměnnou S, jejíž hodnota zachycuje (kóduje) všechnu informaci zapsanou v poli, a to tak, že

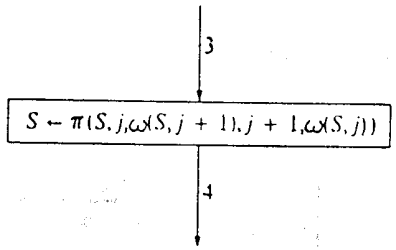
1. $\omega(S, i)$ udává hodnotu (obsah) prvku $S[i]$;
2. $\pi(S, i_1, x_1, i_2, x_2, \ldots, i_n, x_n)$, $n \geqq 1$, udává novou hodnotu proměnné S (novou hodnotu „kódu" reprezentujícího pole S) po současném provedení přiřazovacích příkazů $S[i_1] \leftarrow x_1; S[i_2] \leftarrow x_2; \ldots; S[i_n] \leftarrow x_n$.[1])

V této notaci tedy např. místo $S[i] \leftarrow S[j] + 2$ píšeme $S \leftarrow \pi(S, i, \omega(S, j) + 2)$.

Manipulace s takto zavedenými funkcemi využívá dvou axiómů (o nichž Kaplan [Kaplan 1968] dokázal, že dostatečně charakterizují obě funkce):

1. Je-li $i_1 \neq j, i_2 \neq j, \ldots, i_n \neq j$, je $\omega(\pi(S, i_1, x_1, \ldots, i_n, x_n), j) = \omega(S, j)$.
2. Je-li $i_k = j$, $1 \leqq k \leqq n$, je $\omega(\pi(S, i_1, x_1, \ldots, i_n, x_n), j) = x_k$.

Zapíšeme tvrzení z předcházejícího příkladu s použitím této notace. Přiřazovací příkaz pro výměnu $S[j]$ a $S[j+1]$ dostane tvar



a dané induktivní tvrzení přiřazené hraně 4 bude zapsáno takto:

$$\forall k[0 \leqq k \leqq j \supset \omega(S, k) \leqq \omega(S, j+1)].$$

Po provedení substituci, odpovídajících příkazu, dostaneme tvrzení pro hranu 3:

$$\forall k[0 \leqq k \leqq j \supset \omega(\pi(S, j, \omega(S, j+1), j+1, \omega(S, j)), k) \leqq \\ \leqq \omega(\pi(S, j, \omega(S, j+1), j+1, \omega(S, j)), j+1)].$$

Rozborem případů $k = j$ a $0 \leqq k < j$ můžeme tento výraz za použití axiómů 1 a 2 zjednodušit na

$$\omega(S, j+1) \leqq \omega(S, j) \wedge \forall k[0 \leqq k < j \supset \omega(S, k) \leqq \omega(S, j)].$$

[1]) Předpokládáme, že hodnoty $i_1, i_2, \ldots, i_n$ jsou vždy navzájem různé.

3.2.2. Ukončení

V odst. 3.1.2 jsme se seznámili s metodou, která pomocí tzv. dobře fundovaných množin umožňuje dokázat, že programy končí. Pro naše nynější účely využijeme této jednoduché vlastnosti dobře fundovaných množin:

Nechť $(W, \prec)$ je dobře fundovaná množina, W^n množina všech uspořádaných n-tic prvků z W a $\prec_n$ označuje lexikografické částečné uspořádání množiny W^n, v němž $\langle a_1, a_2, \ldots, a_n\rangle \prec_n \langle b_1, b_2, \ldots, b_n\rangle$, právě když $a_1 = b_1$, $a_2 = b_2$, ..., $a_{i-1} = b_{i-1}$ a $a_i \prec b_i$ pro nějaké i $(1 \leqq i \leqq n)$. Pak $(W^n, \prec_n)$ je také dobře fundovaná množina.

Příklady:

(a) Poněvadž množina $(N, <)$ přirozených čísel s přirozeným uspořádáním $<$ je dobře fundovaná, je dobře fundovaná i množina $(N^2, <_2)$ všech dvojic přirozených čísel. Vztah $\langle n_1, n_2\rangle <_2 \langle m_1, m_2\rangle$ platí právě když $n_1 < m_1$ nebo $(n_1 = m_1) \wedge (n_2 < m_2)$; tak např. je $\langle 1, 100\rangle <_2 \langle 2, 1\rangle$.

(b) Poněvadž abeceda $\Sigma = \{A, B, \ldots, Z\}$ s obvyklým uspořádáním $A \prec B \prec C \prec \ldots \prec Z$ je dobře fundovaná, je dobře fundovaná i množina $(\Sigma^3, \prec_3)$ všech slov o délce 3. Uspořádání $\prec_3$ tu představuje běžné abecední uspořádání; je např. $AKT \prec_3 BOJ \prec_3 BOL$.

Využijeme nyní dobře fundovaných množin tvaru $(W^n, \prec_n)$ k důkazu ukončení několika programů s poli.

Příklad 3.7

Vezměme program P_5 z př. 3.6 a dokažme, že končí pro $\varphi(n)$: $n \geqq 0$. Zvolme dobře fundovanou množinu $(N^2, <_2)$ a dělicí bod B (viz obr. 3.11), k němuž přiřadíme predikát

$$q(n, i, j): \quad 0 \leqq j \leqq i \wedge 1 \leqq i \leqq n$$

a funkci

$$u(i, j): \quad (i, i - j).$$

Důkaz probíhá ve třech krocích:

1. q je dobrá induktivní podmínka; snadno totiž dokážeme, že platí: Pro cestu z A do B:

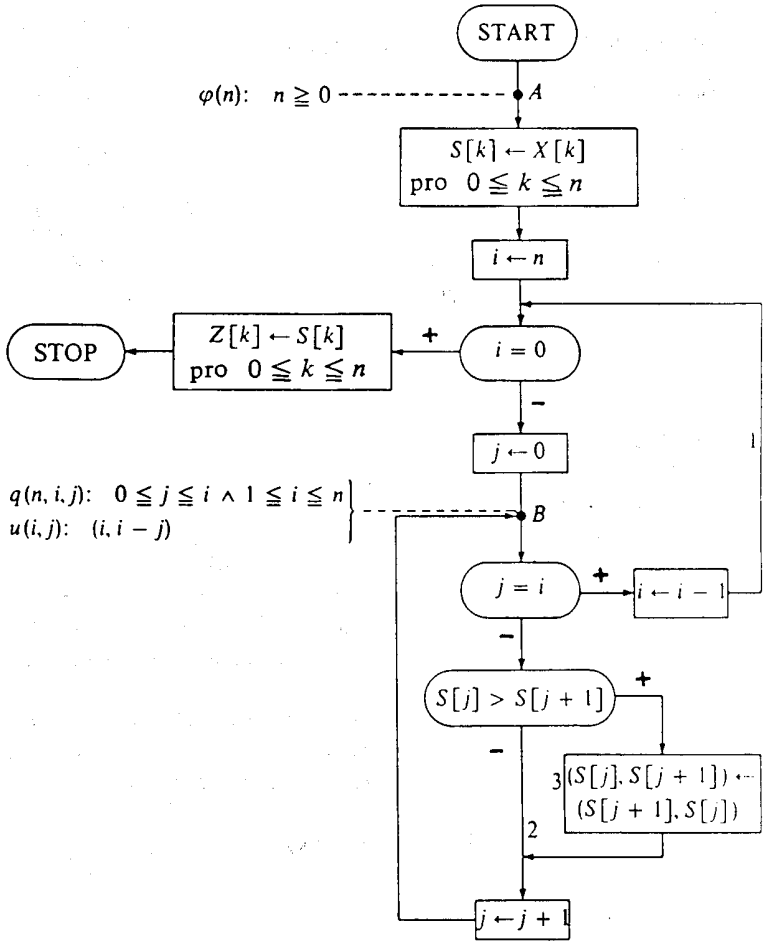
$$[\varphi(n) \wedge n \neq 0] \supset q(n, n, 0).$$

Pro cestu z B do B přes hranu 1:

$$[q(n, i, j) \wedge j = i \wedge i - 1 \neq 0] \supset q(n, i - 1, 0).$$

Pro cestu z B do B přes hranu 2 či příkaz 3:

$$[q(n, i, j) \wedge j \neq i] \supset q(n, i, j + 1).$$



Obr. 3.11. Program P_5 pro setřídění prvků pole (ukončení)

2. u je dobrá funkce; opět lze snadno ukázat, že platí

$$q(n, i, j) \supset u(i, j) \in N^2,$$

tj.

$$[0 \leqq j \leqq i \wedge 1 \leqq i \leqq n] \supset [0 \leqq i \wedge 0 \leqq i - j].$$

3. Podmínky ukončení jsou splněny.

Pro cestu z B do B přes hranu 1:

$$[q(n, i, j) \wedge j = i \wedge i - 1 \neq 0] \supset [u(i, j) >_2 u(i - 1, 0)],$$

tj.

$$[0 \leqq j \leqq i \wedge 1 \leqq i \leqq n \wedge j = i \wedge i - 1 \neq 0] \supset [(i, i - j) >_2 (i - 1, i - 1)].$$

Pro cestu z B do B přes hranu 2 nebo příkaz 3:

$$[q(n,i,j) \wedge j \neq i] \supset [u(i,j) >_2 u(i,j+1)],$$

tj.

$$[0 \leqq j \leqq i \wedge 1 \leqq i \leqq n \wedge j \neq i] \supset [(i,i-j) >_2 (i,i-(j+1))].$$

Také tyto podmínky jsou pravdivé pro všechna celá čísla i, j, n: první podmínka ukončení platí, protože levá složka v dvojici argumentů funkce u klesá z i na $i-1$ a pravá nás proto nemusí ani zajímat; druhá podmínka je splněna, protože levá složka zůstává nezměněna a pravá se zmenšuje z $i-j$ na $i-(j+1)$.

Program tedy končí při libovolném vstupním poli $X[0], \ldots, X[n]$, kde $n \geqq 0$; všimněme si, že důkaz se opírá pouze o hodnoty i, j a n a vlastní hodnoty pole X nepotřebuje brát v úvahu. ∎

Příklad 3.8

Program P_6 z obr. 3.12 vyhodnocuje determinant $z = |X|$ dvoudimenzionálního pole (matice) reálných čísel X řádu n ($n \geqq 1$) pomocí Gaussovy eliminační metody. Pro jednoduchost použijeme X jako vstupní pole i jako pole, v němž probíhá vlastní zpracování. Ukážeme, že program končí pro $\varphi(n)$: $n \geqq 1$, tj. pro každou matici X tvořenou n^2 reálnými čísly $X[i,j]$, kde $1 \leqq i, j \leqq n$ a $n \geqq 1$. Program používá proměnnou y k zápisu reálných čísel a celočíselné proměnné i, j, k. Předpokládáme, že dělení je definováno pro všechna reálná čísla (např. interpretuje $r/0$ jako $r/10^{-10}$).

K důkazu využijeme dobře fundované množiny $(N^3, <_3)$. Zvolíme dělicí body B, C a přiřadíme jim po řadě predikáty q_B, q_C a funkce u_B, u_C podle obr. 3.12.

Důkaz opět probíhá ve třech krocích: pro všechna celá čísla i, j, k, n je třeba dokázat:

1. q_B a q_C jsou dobré induktivní podmínky.

Pro cestu z A do B:

$$[\varphi(n) \wedge 1 \neq n] \supset q_B(n,2,1).$$

Pro cestu z B do B:

$$[q_B(n,i,k) \wedge k+1 \neq n \wedge i = n+1] \supset q_B(n,k+2,k+1).$$

Pro cestu z B do C:

$$[q_B(n,i,k) \wedge i \neq n+1] \supset q_C(n,i,n,k).$$

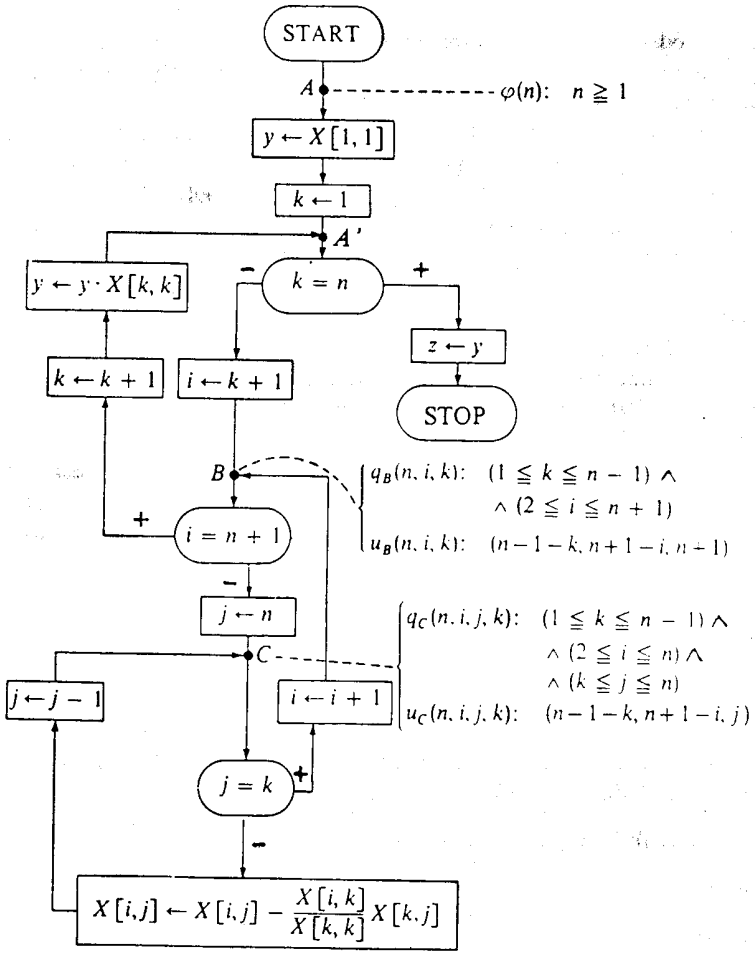
Pro cestu z C do B

$$[q_C(n,i,j,k) \wedge j = k] \supset q_B(n,i+1,k).$$

Pro cestu z C do C:

$$[q_C(n,i,j,k) \wedge j \neq k] \supset q_C(n,i,j-1,k).$$

2. u_B a u_C jsou dobré funkce.



Obr. 3.12. Program P_6 pro vyhodnocení determinantu $z = |X|$ (ukončení)

Pro dělicí bod B:

$$q_B(n,i,k) \supset u_B(n,i,k) \in N^3.$$

Pro dělicí bod C

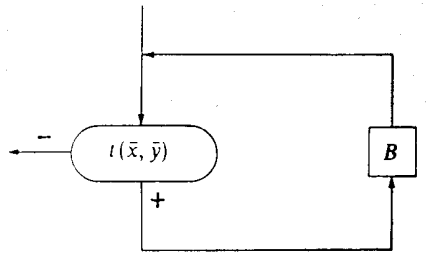
$$q_C(n,i,j,k) \supset u_C(n,i,j,k) \in N^3.$$

3. Podmínky ukončení jsou splněny.

Pro cestu z B do B:

$$[q_B(n,i,k) \wedge k+1 \neq n \wedge i = n+1] \supset$$
$$\supset [u_B(n,i,k) >_3 u_B(n,k+2,k+1)].$$

while $t(\bar{x}, \bar{y})$ do B



Jestliže výpočet končí (tím, že přejde k příkazu zastavení), výstupnímu vektoru $\bar{z}$ je přiřazena průběžná hodnota $h(\bar{x}, \bar{y})$ – označme ji např. $\bar{\zeta}$; řekneme, že v tomto případě je $P(\bar{\xi})$ definováno a píšeme $P(\bar{\xi}) = \bar{\zeta}$. V opačném případě, jestliže běh programu nikdy nekončí, říkáme, že $P(\bar{\xi})$ není definováno.

Příklad 3.10

1. Program P_1 z př. 3.1 pro výpočet druhé odmocniny může být přepsán do tvaru strukturovaného programu:

START
$(y_1, y_2, y_3) \leftarrow (0, 1, 1)$;
while $y_2 \leqq x$ **do** $(y_1, y_2, y_3) \leftarrow (y_1 + 1, y_2 + y_3 + 2, y_3 + 2)$;
$z \leftarrow y_1$
STOP

2. Program P_3 z př. 3.3 pro výpočet největšího společného dělitele lze přepsat do tvaru:

START
$(y_1, y_2) \leftarrow (x_1, x_2)$;
while $y_1 \neq y_2$ **do if** $y_1 > y_2$ **then** $y_1 \leftarrow y_1 - y_2$ **else** $y_2 \leftarrow y_2 - y_1$;
$z \leftarrow y_1$
STOP

3. Program P_4 z př. 3.5 pro výpočet největšího společného dělitele lze vyjádřit ve tvaru:

START
$(y_1, y_2, y_3) \leftarrow (x_1, x_2, 1)$;
while *sudé*(y_1) **do**
    begin
    if *sudé*(y_2) **do** $(y_2, y_3) \leftarrow (y_2/2, 2y_3)$;

    $y_1 \leftarrow y_1/2$
    end;
while *sudé*$(y_2) \vee y_1 \neq y_2$ **do**
    begin
    if *liché*(y_2) **do** $(y_1, y_2) \leftarrow (y_2, |y_1 - y_2|)$;
    $y_2 \leftarrow y_2/2$
    end;
$z \leftarrow y_1 y_3$
STOP

3.3.2. Parciální korektnost

K dokazování parciální korektnosti strukturovaných programů zavedl C. A. R. Hoare ([Hoare 1969]) tzv. *induktivní výrazy*, tj. zápisy tvaru

$$\{p(\bar{x}, \bar{y})\}\ B\{q(\bar{x}, \bar{y})\},$$

kde p a q jsou predikáty a B segment programu. Význam takového zápisu spočívá v tvrzení, že kdykoliv $p(\bar{x}, \bar{y})$ platí pro hodnoty $\bar{x}$, $\bar{y}$ v okamžiku bezprostředně předcházejícím provedení B a B končí, pak $q(\bar{x}, \bar{y})$ je pravdivé pro hodnoty $\bar{x}$, $\bar{y}$ po provedení B.

Dokázat, že program P je parciálně korektní vzhledem ke vstupnímu predikátu φ a výstupnímu predikátu ψ, tedy znamená odvodit

$$\{\varphi(\bar{x})\}\ P\{\psi(\bar{x}, \bar{z})\}.$$

K zvládnutí tohoto úkolu jsou k dispozici *verifikační pravidla*, tvořená jednak *axiómem přiřazení*, který popisuje změnu hodnot programových proměnných, vyvolanou užitím přiřazovacího příkazu, jednak *odvozovacími pravidly*, jejichž pomocí lze z induktivních výrazů pro malé segmenty sestavovat výrazy pro větší celky. Celý mechanismus aplikujeme tak, že nejprve stanovíme induktivní výraz $\{p\}\ B\{q\}$ pro každý přiřazovací příkaz programu podle axiómu přiřazení a potom sestavujeme postupně stále větší segmenty pomocí odvozovacích pravidel, dokud nezískáme požadovaný výraz $\{\varphi\}\ P\{\psi\}$.

Odvozovací pravidla, mezi něž počítáme níže uvedená pravidla 2 až 5, zapisujeme ve tvaru

$$\frac{\alpha_1}{\beta} \quad \text{nebo} \quad \frac{\alpha_1 \text{ a } \alpha_2}{\beta},$$

kde α_1 a α_2 jsou *antecedenty* (podmínky, za nichž je to které pravidlo možné použít) a β je *konsekvent* (induktivní výraz, který chceme odvodit). Každý z antecedentů je buď dříve odvozený induktivní výraz, popř. axióm přiřazení, nebo logický výraz, který je třeba dokázat zvlášť jakožto lemma.

Příklad 3.11

Dokažme ještě jednou parciální korektnost programu P_1 z př. 3.1 vzhledem ke vstupnímu predikátu $x \geqq 0$ a výstupnímu predikátu $z^2 \leqq x < (z+1)^2$. Jde o program (viz př. 3.10):

START
$(y_1, y_2, y_3) \leftarrow (0, 1, 1)$;
while $y_2 \leqq x$ **do** $(y_1, y_2, y_3) \leftarrow (y_1 + 1, y_2 + y_3 + 2, y_3 + 2)$;
$z \leftarrow y_1$
STOP

V důkazu bude figurovat predikát[1])

$$R(x, y_1, y_2, y_3) : (y_1^2 \leqq x) \wedge (y_2 = (y_1 + 1)^2) \wedge (y_3 = 2y_1 + 1).$$

Důkaz rozepíšeme do devíti kroků:

1. $x \geqq 0 \supset R(x, 0, 1, 1)$ — Lemma 1
2. $\{x \geqq 0\}$
   $(y_1, y_2, y_3) \leftarrow (0, 1, 1)$
   $\{R(x, y_1, y_2, y_3)\}$ — Pravidlo přiřazení (viz krok 1)
3. $R(x, y_1, y_2, y_3) \wedge y_2 \leqq x \supset R(x, y_1 + 1, y_2 + y_3 + 2, y_3 + 2)$ — Lemma 2
4. $\{R(x, y_1, y_2, y_3) \wedge y_2 \leqq x\}$
   $(y_1, y_2, y_3) \leftarrow (y_1 + 1, y_2 + y_3 + 2, y_3 + 2)$
   $\{R(x, y_1, y_2, y_3)\}$ — Pravidlo přiřazení (viz krok 3)
5. $\{R(x, y_1, y_2, y_3)\}$
   while $y_2 \leqq x$ **do** $(y_1, y_2, y_3) \leftarrow (y_1 + 1, y_2 + y_3 + 2, y_3 + 2)$
   $\{R(x, y_1, y_2, y_3) \wedge y_2 > x\}$ — Pravidlo cyklu (krok 4)
6. $\{x \geqq 0\}$
   $(y_1, y_2, y_3) \leftarrow (0, 1, 1)$;
   while $y_2 \leqq x$ **do** $(y_1, y_2, y_3) \leftarrow (y_1 + 1, y_2 + y_3 + 2, y_3 + 2)$
   $\{R(x, y_1, y_2, y_3) \wedge y_2 > x\}$ — Pravidlo sekvence (kroky 2 a 5)
7. $R(x, y_1, y_2, y_3) \wedge y_2 > x \supset y_1^2 \leqq x < (y_1 + 1)^2$ — Lemma 3
8. $\{R(x, y_1, y_2, y_3) \wedge y_2 > x\}$
   $z \leftarrow y_1$
   $\{z^2 \leqq x < (z+1)^2\}$ — Pravidlo přiřazení (krok 7)
9. $\{x \geqq 0\}$
   $(y_1, y_2, y_3) \leftarrow (0, 1, 1)$;
   while $y_2 \leqq x$ **do** $(y_1, y_2, y_3) \leftarrow (y_1 + 1, y_2 + y_3 + 2, y_3 + 2)$;
   $z \leftarrow y_1$
   $\{z^2 \leqq x < (z+1)^2\}$ — Pravidlo sekvence (kroky 6 a 8).

Poněvadž lemmata 1, 2, 3 jsou pravdivá, uvedené odvození dokazuje, že P_1 je parciálně korektní vzhledem k $x \geqq 0$ a $z^2 \leqq x < (z+1)^2$.

[1]) Srov. s indukční podmínkou p z př. 3.1. Pozn. překl.

Právě tak jako v metodě induktivních podmínek (odst. 3.1.2) celý důkaz stojí a padá s vhodnou volbou predikátu $R(x, y_1, y_2, y_3)$. Obecně je k jeho stanovení nutné proniknout k podstatě algoritmu, vyjádřeného v programu. Ideální by bylo, kdyby tyto predikáty programátor zapsal na příslušná místa v programu ve formě poznámek (uzavřených do složených závorek). Náš program pro výpočet druhé odmocniny by pak vypadal např. takto:

START
$\{x \geqq 0\}$
$(y_1, y_2, y_3) \leftarrow (0, 1, 1)$;
while $y_2 \leqq x$ **do** $\{(y_1^2 \leqq x) \wedge (y_2 = (y_1 + 1)^2) \wedge (y_3 = 2y_1 + 1)\}$
$(y_1, y_2, y_3) \leftarrow (y_1 + 1, y_2 + y_3 + 2, y_3 + 2)$;
$z \leftarrow y_1$
$\{z^2 \leqq x < (z+1)^2\}$
STOP

■

Příklad 3.12 ([Hoare 1961, 1971]).

V tomto příkladě uvažujeme program P_S, který přerovná prvky pole S sestávajícího z $n+1$ ($n > 0$) reálných čísel $S[0], \ldots, S[n]$ a najde přirozená čísla i, j tak, že $0 \leqq j < i \leqq n$ a

$$\forall a \forall b[(0 \leqq a < i \wedge j < b \leqq n) \supset S[a] \leqq S[b]].$$

Jinak řečeno, program rozmístí prvky pole S do dvou neprázdných podmnožin takových, že všechny prvky „dolní" z nich, totiž $S[0], \ldots, S[i-1]$, jsou menší nebo rovny všem prvkům druhé „horní" podmnožiny, tvořené prvky $S[j+1], \ldots, S[n]$ ($0 \leqq j < i \leqq n$)[1]). V programu je použito n jako vstupní proměnná, r jako programová proměnná a i, j jako výstupní proměnné. S je pro jednoduchost použito jako vstupní, programové i výstupní pole.

START
$\{n > 0\}$
$r \leftarrow S[podil(n, 2)]$; $(i, j) \leftarrow (0, n)$;
while $i \leqq j$ **do**
  begin while $S[i] < r$ **do** $i \leftarrow i + 1$;
    while $r < S[j]$ **do** $j \leftarrow j - 1$;
    if $i \leqq j$ **then begin** $(S[i], S[j]) \leftarrow (S[j], S[i])$;
      $(i, j) \leftarrow (i + 1, j - 1)$
    end
  end
$\{0 \leqq j < i \leqq n \wedge \forall a \forall b[(0 \leqq a < i \wedge j < b \leqq n) \supset S[a] \leqq S[b]]\}$
STOP

[1]) Platnost vztahu $j < i$ je zajištěna tím, že po skončení běhu je buď $j = i - 1$ nebo $j = i - 2$. Uvedené podmnožiny tedy buď na sebe „navazují" ($j + 1 = i$), nebo mají společný prvek ($S[i-1] = S[j+1]$). Pozn. překl.

Postupný rozklad na podmnožiny menších a větších prvků se uskutečňuje tak, že jednotlivé prvky pole jsou porovnávány s prostředním prvkem $r = S[podil(n, 2)]$[1]) a menší jsou zařazovány do dolní podmnožiny, zatímco větší jsou zařazovány do podmnožiny horní. Hodnota i je zpočátku 0, hodnota j je n; i je postupně zvyšováno, pokud platí $S[i] < r$, neboť v tomto případě patří porovnané prvky do dolní podmnožiny a není třeba je přemísťovat. Jakmile se objeví $S[i]$, které není menší než r (a nemůže proto zůstat na místě), zvyšování i je ukončeno. Analogicky je hodnota j postupně snižována, pokud $r < S[j]$ a tento proces se zastaví, jakmile se narazí na $S[j]$, které není větší než r. V tomto okamžiku jsou prvky $S[i]$ i $S[j]$ oba na špatném místě; k nápravě dojde tím, že je zaměníme. Je-li při tom $i \leqq j$, proces zvyšování i a snižování j pokračuje, pokud se nenajde další dvojice k výměně. Jakmile je $j < i$, úkol je vyřešen.

Chceme dokázat, že program je parciálně korektní, tj. že po jeho skončení (dojde-li k němu): (1) prvky pole S tvoří permutaci prvků původního pole; (2) je pravdivý predikát

$$(j < i) \wedge \forall a \forall b[(0 \leqq a < i \wedge j < b \leqq n) \supset S[a] \leqq S[b]].$$

Důkaz silnějšího tvrzení pro $0 \leqq j < i \leqq n$ přenecháme čtenáři (viz cv. 3.13).

Platnost podmínky (1) vyplývá z toho, že jedinou operací nad polem S je přiřazení $(S[i], S[j]) \leftarrow (S[j], S[i])$, které hodnoty prvků pole přeskupuje, ale nemění. Platnost vztahu (2) lze prokázat postupnou aplikací verifikačních pravidel. Zde se omezíme na to, že ukážeme, jak vypadají potřebné predikáty. Program opatřený vhodnými predikáty ve formě poznámek má tvar:[2])

START
$\{n > 0\}$
$r \leftarrow S[podil(n, 2)]; (i, j) \leftarrow (0, n);$
$\{0 \leqq i \wedge \forall a(0 \leqq a < i \supset S[a] \leqq r) \wedge$ (i-invariant)
$\wedge\ j \leqq n \wedge \forall b(j < b \leqq n \supset r \leqq S[b])\}$ (j-invariant)
while $i \leqq j$ **do**
  begin {i-invariant $\wedge$ j-invariant}
    while $S[i] < r$ **do** $i \leftarrow i + 1$;
    {i-invariant $\wedge$ j-invariant $\wedge$ $r \leqq S[i]$}
    while $r < S[j]$ **do** $j \leftarrow j - 1$;
    {i-invariant $\wedge$ j-invariant $\wedge$ $S[j] \leqq r \leqq S[i]$}
    if $i \leqq j$ **then begin** $(S[i], S[j]) \leftarrow (S[j], S[i])$;
      $(i, j) \leftarrow (i + 1, j - 1)$
    end
  end {i-invariant $\wedge$ j-invariant}
$\{j < i \wedge \forall a \forall b[(0 \leqq a < i \wedge j < b \leqq n) \supset S[a] \leqq S[b]]\}$
STOP ∎

[1]) $podil(n, 2)$ označuje podíl v celočíselném dělení čísla n dvěma.

[2]) Autor zde poprvé používá k označení některých částí predikátu termínu „invariant" (i-invariant, j-invariant). Viz pozn. pod čarou [3]) na str. 162. Pozn. překl.

3.3.3. Totální korektnost

Metoda verifikačních pravidel (věta 3.3) umožňuje dokázat pouze parciální korektnost strukturovaných programů. V práci [Manna, Pnueli 1973] byla tato metoda rozšířena na prostředek k důkazu totální korektnosti (včetně ukončení). Autoři zavedli označení

$$\{p(\bar{x}, \bar{y})\}\ B\{q(\bar{x}, \bar{y}, \bar{y}')\}^1)$$

k vyjádření tvrzení, že kdykoliv $p(\bar{x}, \bar{y})$ je pravdivé před provedením B, pak výpočet podle B končí a po jeho ukončení platí $q(\bar{x}, \bar{y}, \bar{y}')$, kde $\bar{x}$, $\bar{y}$ označuje hodnoty příslušných vektorů před provedením B, $\bar{y}'$ hodnoty vektoru programu po provedení B.

Za pomoci této notace mohou být formulována nová pravidla, tzv. *pravidla ukončení*, která zaručují, že ukončení segmentů se „dědí" z malých segmentů na větší segmenty. Jestliže pomocí těchto pravidel odvodíme

$$\{\varphi(\bar{x})\}\ P\{\psi(\bar{x}, \bar{z})\},$$

dokážeme tím, že P je totálně korektní vzhledem k φ a ψ.

Nejprve uvedeme pravidla ukončení pro přiřazovací příkazy, podmíněné příkazy a sekvence příkazů, jejichž význam je zřejmý. Komplikovanější případ, týkající se cyklů, ponecháme na závěr. Poněvadž $\bar{x}$ se během programu nemění, vypustíme ze zápisu predikátů ve všech pravidlech zmínku o $\bar{x}$; místo $p(\bar{x}, \bar{y})$ tedy píšeme stručněji $p(\bar{y})$, místo $q(\bar{x}, \bar{y}, \bar{y}')$ stručněji $q(\bar{y}, \bar{y}')$.

Pravidla ukončení:

1. *Pravidlo přiřazení:*

$$\frac{\forall \bar{y} \forall \bar{y}'[p(\bar{y}) \wedge \bar{y}' = f(\bar{y}) \supset q(\bar{y}, \bar{y}')]}{\{p(\bar{y})\}\ \bar{y} \leftarrow f(\bar{y})\ \{q(\bar{y}, \bar{y}')\}}$$

Toto pravidlo představuje v podstatě axióm, neboť k odvození induktivního výrazu využívá pouze logického výrazu.

2. *Pravidla podmínky:*

$$\frac{\{p(\bar{y}) \wedge t(\bar{y})\}\ B_1\{q(\bar{y}, \bar{y}')\} \quad \{p(\bar{y}) \wedge \sim t(\bar{y})\}\ B_2\{q(\bar{y}, \bar{y}')\}}{\{p(\bar{y})\}\ \textbf{if}\ t(\bar{y})\ \textbf{then}\ B_1\ \textbf{else}\ B_2\{q(\bar{y}, \bar{y}')\}}$$

a

$$\frac{\{p(\bar{y}) \wedge t(\bar{y})\}\ B\{q(\bar{y}, \bar{y}')\} \quad \forall \bar{y} \forall \bar{y}'[p(\bar{y}) \wedge \sim t(\bar{y}) \supset q(\bar{y}, \bar{y}')]}{\{p(\bar{y})\}\ \textbf{if}\ t(\bar{y})\ \textbf{do}\ B\{q(\bar{y}, \bar{y}')\}}$$

[1]) Těmto zápisům se v dalším výkladu opět říká induktivní výrazy. Pozn. překl.

celý důkazový postup? V našem konkrétním případě lze uvažovat takto. Předpokládejme, že místo predikátu $q(\bar{y}, \bar{y}')$ použijeme ve všech induktivních tvrzeních predikát

$$q(\bar{y}, \bar{y}') \wedge nsd(y_1, y_2) = nsd(y_1', y_2').$$

Není obtížné ověřit, že všechny uvažované kroky zůstanou v platnosti a že jsme tedy pro segment f schopni odvodit

$$\{y_1 > 0 \wedge y_2 > 0\}\, f\{y_1' = y_2' \wedge nsd(y_1, y_2) = nsd(y_1', y_2')\},$$

tj.

$$\{y_1 > 0 \wedge y_2 > 0\}\, f\{y_1' = nsd(y_1, y_2)\}.$$

Pro celý program tak dostáváme

$$\{x_1 > 0 \wedge x_2 > 0\}\, P_9\{z = nsd(x_1, x_2)\}.$$

■

Tato úvaha má obecnější platnost a lze ji shrnout do následujícího tvrzení:

Věta 3.5[1])

Předpokládejme, že jsme odvodili induktivní výraz

$$\alpha: \quad \{\varphi(\bar{x})\}\, P\{\psi(\bar{x}, \bar{z})\}.$$

Buď $S(\bar{y}, \bar{y}')$ reflexivní a tranzitivní relace, tj. nechť platí $\forall \bar{y}[S(\bar{y}, \bar{y})]$ a $\forall \bar{y}\forall \bar{y}'\forall \bar{y}''[S(\bar{y}, \bar{y}') \wedge S(\bar{y}', \bar{y}'') \supset S(\bar{y}, \bar{y}'')]$. Předpokládejme dále, že spolu s libovolným induktivním výrazem tvaru

$$\{p(\bar{y})\}\, \bar{y} \leftarrow g(\bar{x}, \bar{y})\, \{q(\bar{y}, \bar{y}')\},$$

použitým v odvození α, je možné odvodit i

$$\{p(\bar{y})\}\, \bar{y} \leftarrow g(\bar{x}, \bar{y})\, \{q(\bar{y}, \bar{y}') \wedge S(\bar{y}, \bar{y}')\}.$$

Pak je pro celý program P pravdivý i induktivní výraz

$$\alpha^+: \quad \{\varphi(\bar{x})\}\, P\{\psi(\bar{x}, \bar{z}) \wedge S(\bar{x}, \bar{z})\}.$$

Zamýšlíme-li tedy rozšířit odvozený induktivní výraz, stačí zkoumat přiřazovací příkazy daného programu. U př. 3.13 jsou jedinými přiřazovacími příkazy příkazy

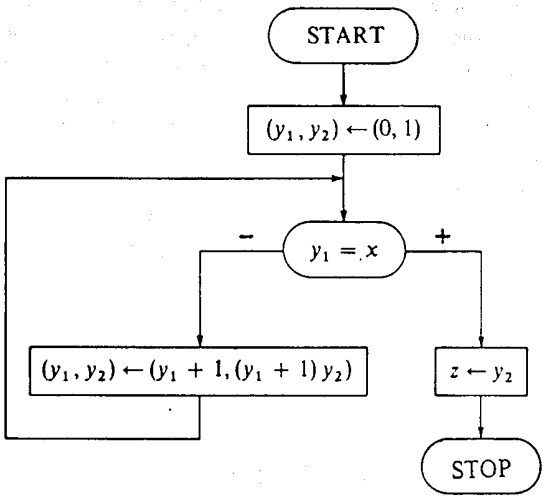
$$y_1 \leftarrow y_1 - y_2 \quad \text{a} \quad y_2 \leftarrow y_2 - y_1,$$

které zřejmě zachovávají funkci *nsd*.[2])

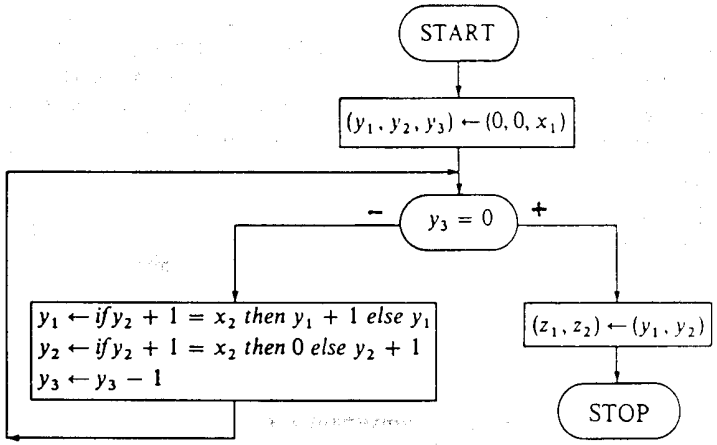
K důkazu věty 3.5 je třeba vyšetřit všechna pravidla ukončení různá od pravidla přiřazení a ukázat, že kdykoli je S zachováno jednotlivými složkami, je zachováno i většími segmenty.

[1]) Autor zde hovoří o „metateorému", aby zdůraznil, že jde o tvrzení o dané metodě, nikoliv o popis metody samé. Rozdíl se však nezdá být tak podstatný, aby ospravedlnil zavedení nečeského termínu. Pozn. překl.

[2]) To znamená, že hodnota $nsd(y_1, y_2)$ před i po provedení těchto příkazů je táž. Pozn. překl.



Obr. 3.14. Program pro výpočet $z = x!$

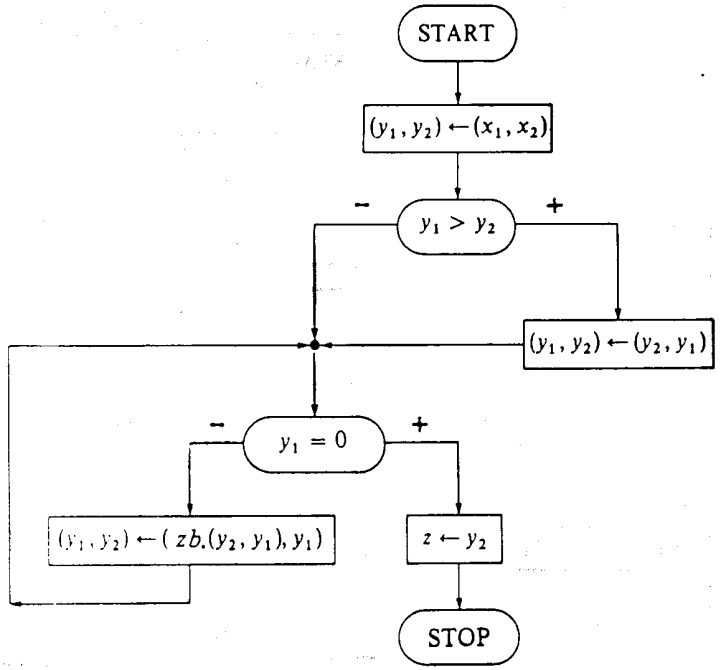


Obr. 3.15. Program pro výpočet podílu a zbytku

CVIČENÍ

Cv. 3.1. Dokažte, že program (nad oborem celých čísel) z obr. 3.14 pro výpočet faktoriálu je totálně korektní vzhledem k vstupnímu predikátu $\varphi(x)$: $x \geqq 0$ a výstupnímu predikátu $\psi(x, z)$: $z = x!$.

Cv. 3.2. Dokažte, že program (nad oborem přirozených čísel) z obr. 3.15 pro výpočet podílu a zbytku je totálně korektní vzhledem k vstupnímu predikátu $\varphi(\bar{x})$: $x_1 > 0 \wedge x_2 > 0$ a výstupnímu predikátu $\psi(\bar{x}, \bar{z})$: $0 \leqq z_2 < x_2 \wedge x_1 = z_1 x_2 + z_2$ $[z_1 = podíl(x_1, x_2), z_2 = zbytek(x_1, x_2)]$.



Obr. 3.16. Program pro výpočet $z = nsd(x_1, x_2)$

Cv. 3.3. Dokažte, že program (nad oborem přirozených čísel) z obr. 3.16 pro výpočet největšího společného dělitele čísel x_2 a x_2 je totálně korektní vzhledem k vstupnímu predikátu $\varphi(x_1, x_2)$: $x_1 > 0 \wedge x_2 > 0$ a výstupnímu predikátu $\psi(x_1, x_2, z)$: $z = nsd(x_1, x_2)$.

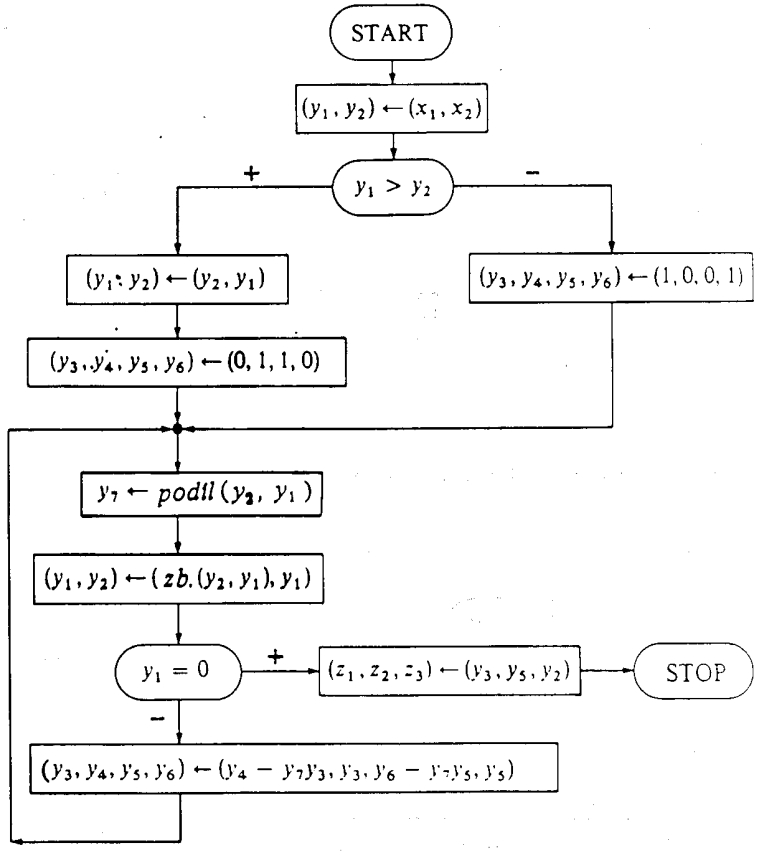
Cv. 3.4. ([Knuth 1968].) Dokažte, že program (nad oborem přirozených čísel) z obr. 3.17 pro výpočet největšího společného dělitele čísel x_1, x_2 a příslušných činitelů je totálně korektní vzhledem k vstupnímu predikátu $\varphi(\bar{x})$: $x_1 > 0 \wedge x_2 > 0$ a výstupnímu predikátu $\psi(\bar{x}, \bar{z})$: $z_3 = nsd(x_1, x_2) \wedge z_1 x_1 + z_2 x_2 = z_3$.

*Cv. 3.5. Dokažte, že program (nad oborem přirozených čísel) z obr. 3.18 pro výpočet McCarthyho „funkce 91" je totálně korektní vzhledem k vstupnímu predikátu $\varphi(x)$: T a výstupnímu predikátu $\psi(x, z)$: $z = \textit{if } x > 100 \textit{ then } x - 10 \textit{ else } 91$.[1])

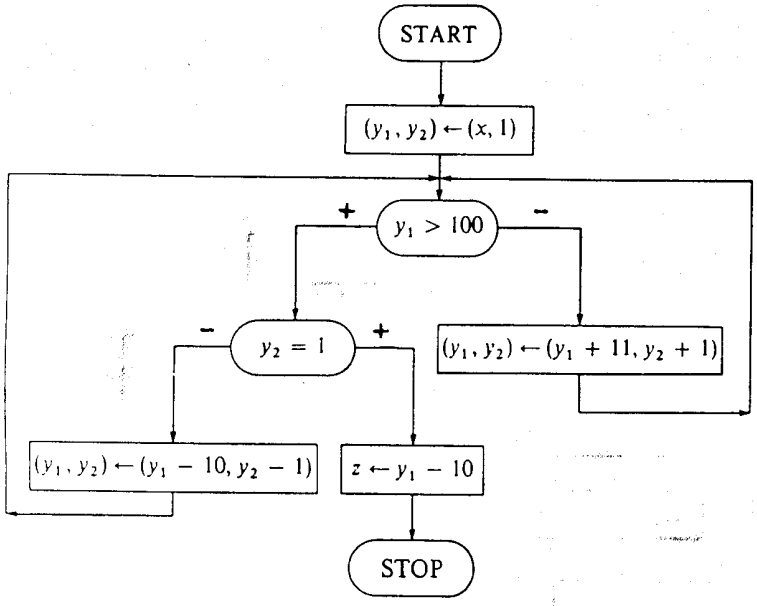
Cv. 3.6. Dokažte, že program (nad oborem celých čísel) z obr. 3.19, který rozhoduje, zda dané číslo x je prvočíslo, je totálně korektní vzhledem k predikátům $\varphi(x)$: $x \geqq 2$ a $\psi(x, z)$: $z \equiv \textit{prvočíslo}(x)$, kde $\textit{prvočíslo}(x)$ je *pravda*, právě když x je prvočíslo, tj. když $\forall k[2 \leqq k < x \supset \textit{zbytek}(x, k) \neq 0]$.[2])

[1]) Další vlastnosti této zajímavé funkce poznáme v kap. 5. Pozn. překl.

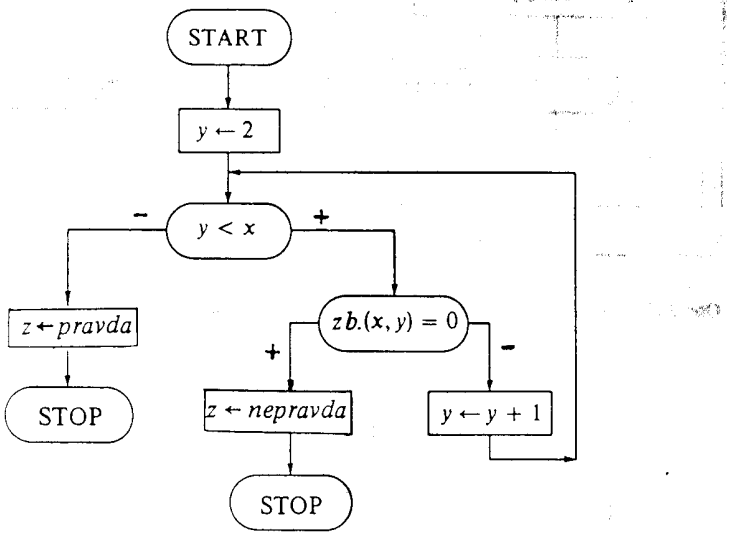
[2]) Poslední formule necharakterizuje ovšem prvočísla mezi všemi celými čísly, nýbrž jen pro $x \geqq 2$. Pozn. rec.



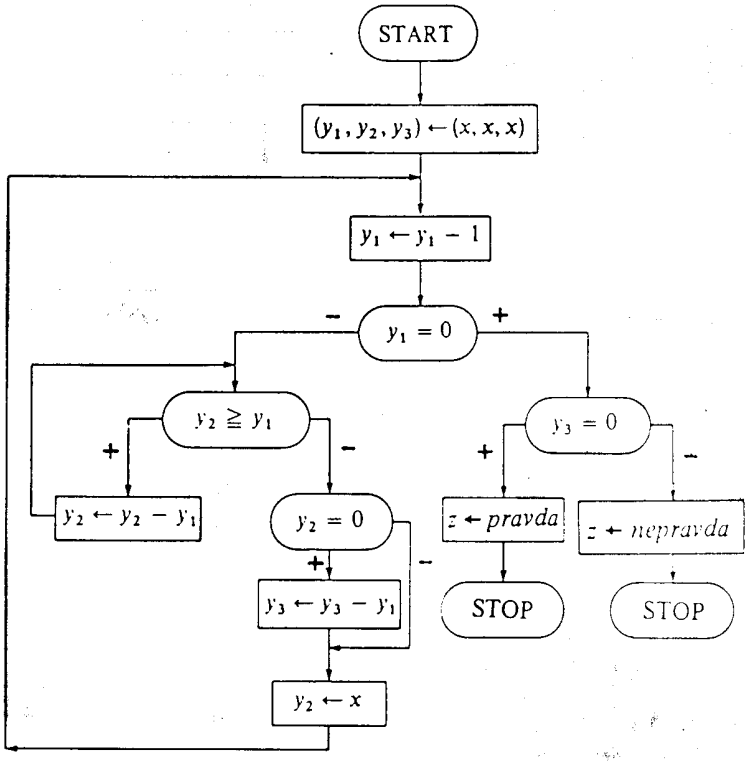
Obr. 3.17. Program pro výpočet $z_3 = nsd(x_1, x_2)$ a příslušných činitelů



Obr. 3.18. Program pro výpočet McCarthyho „funkce 91"



Obr. 3.19. Program rozhodující, zda dané číslo x je prvočíslo



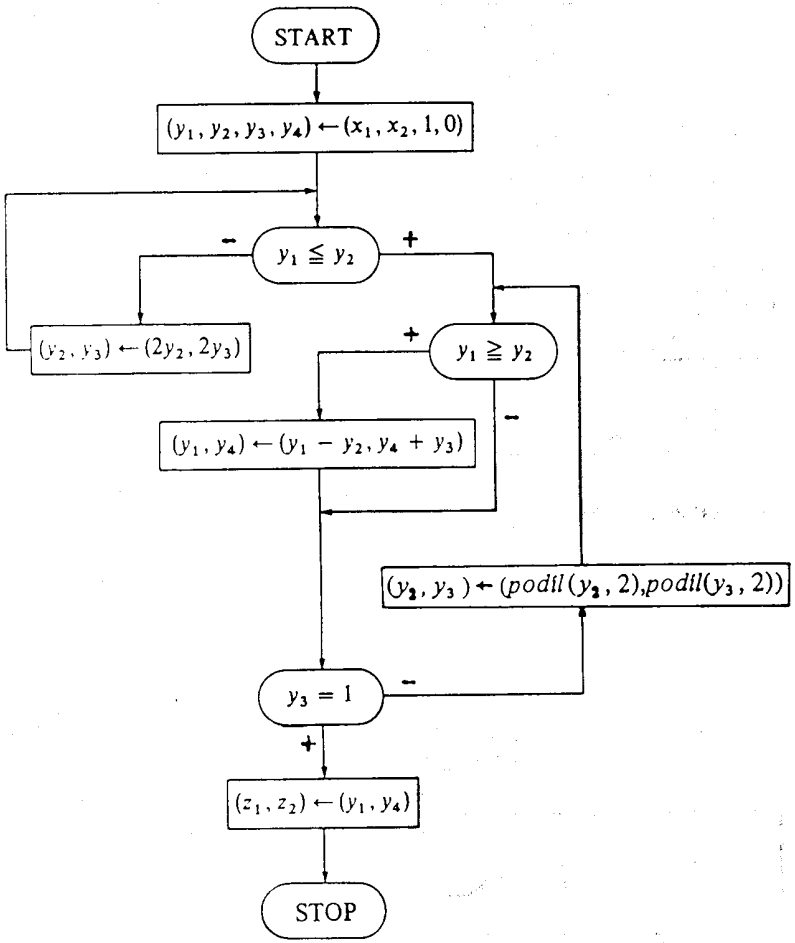
Obr. 3.20. Program rozhodující, zda dané číslo x je dokonalé

Cv. 3.7. Dokažte, že program (nad oborem přirozených čísel) z obr. 3.20, který rozhoduje, zda dané číslo x je dokonalé, je totálně korektní vzhledem k predikátům $\varphi(x)$: $x > 0$ a $\psi(x, z)$: „z je *pravda*, právě když x je dokonalé číslo." (*Číslo* x je *dokonalé*, je-li součtem všech svých dělitelů t, $1 \leqq t < x$. Tak např. 6 a 28 jsou dokonalá čísla, neboť $6 = 1 + 2 + 3$ a $28 = 1 + 2 + 4 + 7 + 14$.)

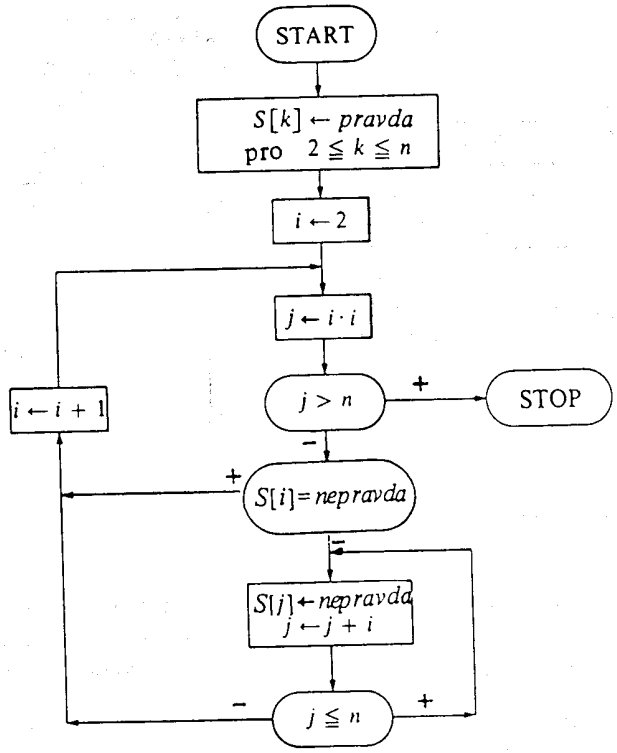
Cv. 3.8. Dokažte, že program (nad oborem celých čísel) z obr. 3.21 pro výpočet podílu a zbytku při dělení čísla x_1 číslem x_2[1]) je totálně korektní vzhledem k vstupnímu predikátu $\psi(\bar{x})$: $x_1 \geqq 0 \wedge x_2 > 0$ a výstupnímu predikátu $\psi(\bar{x}, \bar{z})$: $0 \leqq z_1 < x_2 \wedge x_1 = z_2 x_2 + z_1$.

Cv. 3.9. Program z obr. 3.22 nalezne všechna prvočísla od 2 do n, $n \geqq 2$, metodou Eratosthenova síta. Používá booleovské pole S o $n - 1$ prvcích $S[2]$, $S[3]$, ..., $S[n]$; každé $S[i]$ tedy nabývá vždy jedné ze dvou hodnot: *pravda*, *nepravda*. Dokažte, že program je totálně korektní vzhledem ke vstupnímu predikátu $\varphi(n)$: $n \geqq 2$ a výstupnímu predikátu $\psi(n, S)$: $\forall k[2 \leqq k \leqq n \supset S[k] \equiv prvočíslo(k)]$.

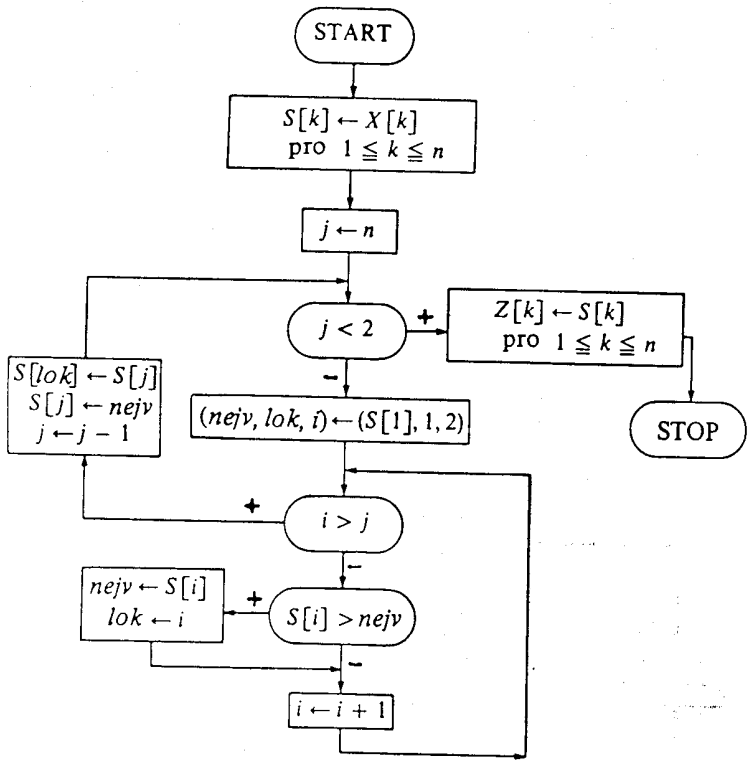
[1]) Jde o metodu používanou v aritmetických jednotkách počítačů, autor ji proto nazývá „hardware integer division". Pozn. překl.



Obr. 3.21. Program pro výpočet podílu a zbytku



Obr. 3.22. Program pro vyhledání všech prvočísel od 2 do n

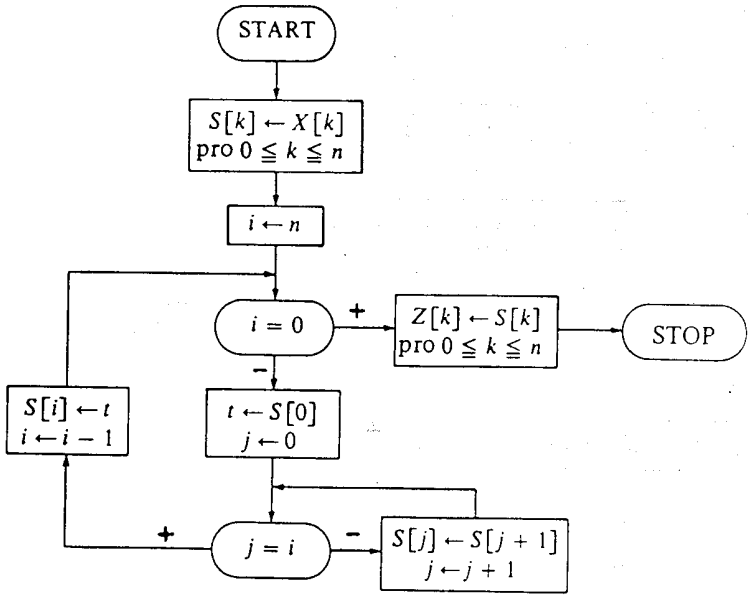


Obr. 3.23. Program pro uspořádání prvků pole

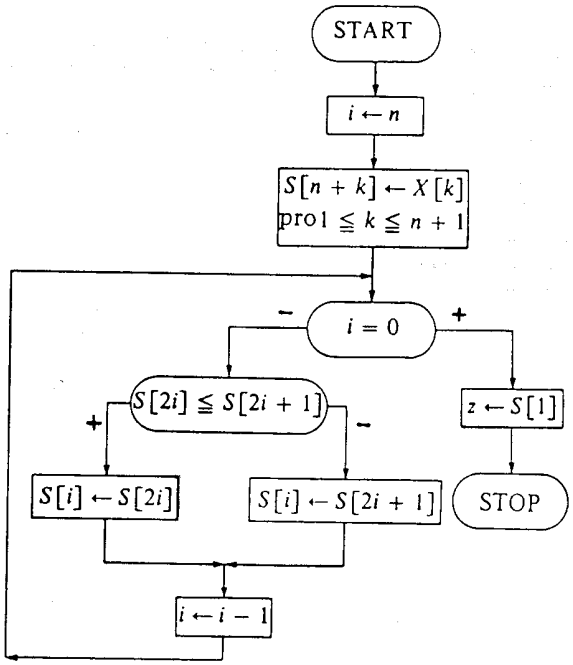
Cv. 3.10. Dokažte, že program z obr. 3.23 pro uspořádání daného pole $X[1], \ldots, X[n]$ reálných čísel ($n \geqq 1$) je totálně korektní vzhledem ke vstupnímu predikátu $\varphi(n, X)$: $n \geqq 1$ a výstupnímu predikátu $\psi(n, X, Z)$: $perm(X, Z, 1, n) \wedge$ $\wedge usp(Z, 1, n)$ (viz př. 3.6, odst. 3.2.1).

Cv. 3.11. Program z obr. 3.24 má na vstupu pole X o $n + 1$ prvcích $X[0], X[1], \ldots, X[n]$, $n \geqq 1$ a na výstupu vytváří pole Z o $n + 1$ prvcích $Z[0], Z[1], \ldots, Z[n]$, přičemž pole Z je tvořeno týmiž prvky jako pole X, tyto prvky jsou však seřazeny v opačném pořadí, tj. $Z[0] = X[n]$, $Z[1] = X[n - 1]$, $\ldots$ $Z[n] = X[0]$. Dokažte, že program je totálně korektní vzhledem k vstupnímu predikátu $\varphi(n, X)$: $n \geqq 0$ a výstupnímu predikátu $\psi(n, X, Z)$: $\forall k[0 \leqq k \leqq n \supset Z[k] =$ $= X[n - k]]$.

Cv. 3.12. Dokažte, že program z obr. 3.25 pro výpočet nejmenšího prvku daného pole reálných čísel $X[1], \ldots, X[n + 1]$, $n \geqq 0$, je totálně korektní vzhledem k vstupnímu predikátu $\varphi(n, X)$: $n \geqq 0$ a výstupnímu predikátu $\psi(n, X, z)$: $z =$ $= \min\{X[1], \ldots, X[n + 1]\}$.



Obr. 3.24. Program obracející pořadí prvků pole



Obr. 3.25. Program pro výpočet nejmenšího prvku pole

```
procedure TREESORT3 (M, n);
  value n: array M; integer n;
  comment Algoritmus uspořádává pole M[1:n], n ≧ 1. Nejlépe je jeho činnost
    patrná, představíme-li si M ve tvaru stromu, u něhož M[j ÷ 2] je otcem M[j]
    pro 1 < j ≦ n;
begin
  procedure výměna (x, y); real x, y;
    begin real t; t: = x; x: = y; y: = t
    end výměna;
  procedure síto(i, n); value i, n; integer i, n;
    comment M[i] „prosíváme" dolů směrem k listům toho podstromu stromu
      M[1: n], jehož je M[i] kořenem;
    begin real pom; integer j;
      pom: = M[i];
    cykl: j: = 2 × i;
      if j ≦ n then
      begin if j < n then
              begin if M[j + 1] > M[j] then j: = j + 1 end;
            if M[j] > pom then
              begin M[i]: = M[j]; i: = j; go to cykl end
      end;
      M[i]: = pom
    end síto;
  integer i;
  for i: = n ÷ 2 step − 1 until 2 do síto(i, n);
  for i: = n step − 1 until 2 do
    begin síto(1, i);
          comment M[j ÷ 2] ≧ M[j] pro 1 < j ≦ i;
          výměna (M[1], M[i]);
          comment M[i: n] je již uspořádáno;
    end
end TREESORT3.
```

Cv. 3.18.

(a) Nalezněte některá uspořádání, vzhledem k nimž jsou dobře fundované tyto množiny:

(i) Množina všech celých čísel [běžné uspořádání $<$ (menší než) nevede k dobře fundované množině!].

(ii) Množina všech racionálních čísel tvaru a/b, kde a i b jsou kladná celá čísla.

(b) Dokažte:

(i) Je-li $(W, \prec)$ dobře fundovaná množina, je dobře fundovaná i množina $(W^n, \prec_n)$, tj. množina n-tic prvků z W částečně uspořádaná lexikograficky[1]).

[1]) Viz odst. 3.2.2. Pozn. překl.

(ii) Jsou-li $(W_1, \prec_1)$ a $(W_2, \prec_2)$ dobře fundované množiny, je dobře fundovaná i množina $(W, \prec)$, kde

(1) $W = W_1 \cup W_2$,

(2) $a \prec b$, právě když $a, b \in W_1$ a $a \prec_1 b$
nebo $a, b \in W_2$ a $a \prec_2 b$
nebo $a \in W_1$ a $b \in W_2$.

Cv. 3.19.

(a) Dokažte, že platí obrácení věty 3.1 v tomto smyslu: Je-li P program, který je parciálně korektní vzhledem k φ a ψ, pak existují dělicí body a induktivní podmínky tak, že odpovídající verifikační podmínky jsou pravdivé.

(b) Dokažte, že platí obrácení věty 3.2 v tomto smyslu: je-li P program, který končí pro φ, pak existují dělicí body, dobré induktivní podmínky a dobré parciální funkce tak, že jsou splněny odpovídající podmínky ukončení.

Cv.3.20 (Dijkstra). Užitím metody verifikačních pravidel dokažte parciální korektnost strukturovaných programů (nad oborem celých čísel):

(a)
```
START
{x1 ≧ 0 ∧ x2 > 0}
(y1, y2) ← (x1, 0);
while x2 ≦ y1 do (y1, y2) ← (y1 − x2, y2 + 1);
(z1, z2) ← (y1, y2)
{0 ≦ z1 < x2 ∧ x1 = z2x2 + z1}
STOP
```

(b)
```
START
{x1 ≧ 0 ∧ x2 ≧ 0}
(y1, y2, y3) ← (x1, x2, 1);
while y2 ≠ 0 do
      begin if liché(y2) do (y2, y3) ← (y2 − 1, y3y1);
            (y1, y2) ← (y1y1, y2/2)
      end;
z ← y3
{z = x1^x2}
STOP
```

(c)
```
START
{x1 > 0 ∧ x2 > 0}
(y1, y2) ← (x1, x2);
  while y1 ≠ y2 do
      if y1 > y2 then y1 ← y1 − y2 else y2 ← y2 − y1;
z ← y1
{z = nsd(x1, x2)}
STOP
```

(d) START
$\{x_1 > 0 \wedge x_2 > 0\}$
$(y_1, y_2) \leftarrow (x_1, x_2)$;
while $y_1 \neq y_2$ **do**
begin while $y_1 > y_2$ **do** $y_1 \leftarrow y_1 - y_2$;
while $y_2 > y_1$ **do** $y_2 \leftarrow y_2 - y_1$
end;
$z \leftarrow y_1$
$\{z = nsd(x_1, x_2)\}$
STOP

(e) START
$\{x_1 \geqq 0 \wedge x_2 > 0\}$
$(y_1, y_2, y_3) \leftarrow (x_1, 0, x_2)$;
while $y_3 \leqq y_1$ **do** $y_3 \leftarrow 2y_3$;
while $y_3 \neq x_2$ **do**
begin $(y_2, y_3) \leftarrow (2y_2, y_3/2)$;
if $y_3 \leqq y_1$ **do** $(y_1, y_2) \leftarrow (y_1 - y_3, y_2 + 1)$
end;
$(z_1, z_2) \leftarrow (y_1, y_2)$
$\{0 \leqq z_1 < x_2 \wedge x_1 = z_2 x_2 + z_1\}$
STOP

(f) START
$\{x_1 > 0 \wedge x_2 > 0\}$
$(y_1, y_2, y_3, y_4) \leftarrow (x_1, x_2, x_2, 0)$;
while $y_1 \neq y_2$ **do**
begin while $y_1 > y_2$ **do** $(y_1, y_4) \leftarrow (y_1 - y_2, y_4 + y_3)$;
while $y_2 > y_1$ **do** $(y_2, y_3) \leftarrow (y_2 - y_1, y_3 + y_4)$
end;
$(z_1, z_2) \leftarrow (y_1, y_3 + y_4)$
$\{z_1 = nsd(x_1, x_2) \wedge z_2 = nsn(x_1, x_2)\}$
STOP

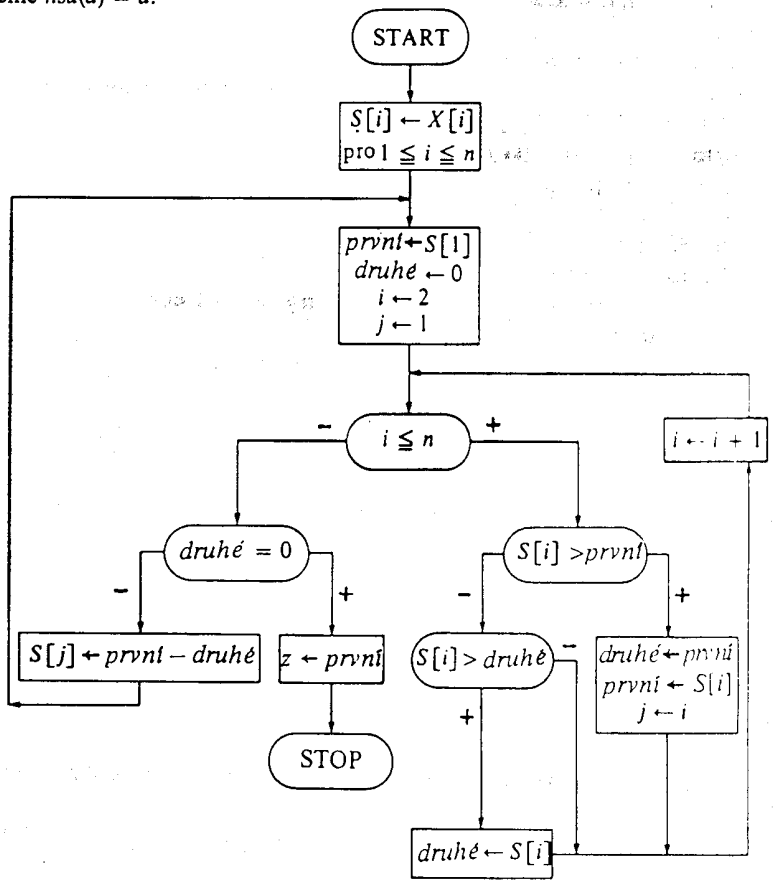
kde $nsn(x_1, x_2)$ je nejmenší společný násobek čísel x_1 a x_2, tzn. jejich součin dělený jejich největším společným dělitelem.

(g) Následující program nalezne dané číslo x v uspořádaném poli $A[1], \ldots, A[n]$, $n \geqq 1$.

START
$\{n \geqq 1 \wedge A[1] \leqq A[2] \leqq \ldots \leqq A[n] \wedge \exists i, 1 \leqq i \leqq n$, tak, že $x = A[i]\}$
$(y_1, y_2) \leftarrow (1, n)$;
while $y_1 \neq y_2$ **do**
begin $y_3 \leftarrow podíl(y_1 + y_2, 2)$;
if $x \leqq A[y_3]$ **then** $y_2 \leftarrow y_3$ **else** $y_1 \leftarrow y_3 + 1$
end;

$z \leftarrow y_1$
$\{1 \leqq z \leqq n \wedge x = A[z]\}$
STOP

***Cv. 3.21.** Dokažte, že program z obr. 3.27 nalezne největší společný dělitel pro libovolné pole nezáporných celých čísel $X[1], \ldots, X[n]$, $n \geqq 1$, tj. že $z = nsd(X[1], \ldots, X[n])$. Případné nuly budou ignorovány, takže např. $nsd(5, 0, 0, 15, 0, 7) = nsd(5, 15, 7)$ a $nsd(0, 5, 0) = nsd(5)$. Přitom pro každé nezáporné číslo a klademe $nsd(a) = a$.



Obr. 3.27. Program pro zjištění největšího společného dělitele prvků pole

***Cv.3.22** ([Duijvestijn 1972]). Dále uvedený program přerovná prvky pole $S[1], S[2], \ldots, S[n]$, $n \geqq 1$, v souhlase s danou permutací f množiny $\{1, 2, \ldots, n\}$, tj. zařídí, že $S_{výstupní}[i] = S_{vstupní}[f(i)]$ pro $1 \leqq i \leqq n$, a to bez dalších nároků na paměť. Dokažte, že program je úplně korektní vzhledem k vstupnímu predikátu

5. Pevné body programů

V této kapitole se budeme zabývat metodami dokazování vlastností rekurzivních programů. Při tom budeme uvažovat jen poměrně omezenou třídu programů odvozenou z konstrukcí známých programovacích jazyků typu Algol a Lisp. Příkladem rekurzivního programu nad oborem celých čísel je

$$P: \quad F(x, y) \Leftarrow \textit{if } x = y \textit{ then } y + 1 \textit{ else } F(x, F(x - 1, y + 1)).$$

Podstatný rozdíl mezi rekurzivními programy, které se chystáme vyšetřovat a mezi těmi, které byly zavedeny v odst. 4.4.1, je v tom, že nyní připustíme možnost, aby základní funkce i predikáty byly parciální, tj. nedefinovány pro některé hodnoty. Jde o zcela přirozené zobecnění, neboť parciální funkce reprezentují výsledky výpočtů, které mohou pro některé vstupní hodnoty končit, avšak pokračovat donekonečna pro jiné. Připustíme samozřejmě i mezní případy, totiž parciální funkce definované pro všechny hodnoty (tj. totální funkce) i funkce nedefinované pro vůbec žádnou hodnotu svého definičního oboru.

Podívejme se na tyto tři funkce:

$f_1(x, y)$: *if* $x = y$ *then* $y + 1$ *else* $x + 1$,
$f_2(x, y)$: *if* $x \geqq y$ *then* $x + 1$ *else* $y - 1$,
$f_3(x, y)$: *if* $(x \geqq y) \wedge (x - y$ sudé*)* *then* $x + 1$ *else* nedefinováno.

Mají zajímavou společnou vlastnost: Pro každou z nich, tj. pro každé i $(1 \leqq i \leqq 3)$, platí, že když nahradíme všechny výskyty symbolu F v programu P symbolem f_i, pak výrazy nalevo i napravo od znaménka $\Leftarrow$ označují tutéž parciální funkci; jinak řečeno, platí[1]):

$$f_i(x, y) \equiv \textit{if } x = y \textit{ then } y + 1 \textit{ else } f_i(x, f_i(x - 1, y + 1)).$$

Říkáme, že funkce f_1, f_2, f_3 jsou *pevnými body rekurzivního programu P.*

Funkce f_3 má mezi touto trojicí zvláštní postavení: Pro každou dvojici celých čísel $\langle x, y\rangle$ takovou, že $f_3(x, y)$ je definováno (tj. že $x \geqq y$ a $x - y$ je sudé), jsou definovány i hodnoty $f_1(x, y)$ a $f_2(x, y)$ a jsou rovny $f_3(x, y)$. Řekneme, že f_3 je *méně nebo stejně definována jako* f_1 a f_2 a píšeme $f_3 \sqsubseteq f_1$ a $f_3 \sqsubseteq f_2$. Lze ukázat, že f_3 je v tomto vztahu nejen vzhledem k funkcím f_1 a f_2, ale vzhledem ke všem

[1]) $A \equiv B$ je *pravda*, právě když A i B jsou definovány a $A = B$ nebo A i B jsou nedefinovány.

pevným bodům programu P a že je jedinou funkcí, která má tuto vlastnost: řekneme proto, že f_3 je *nejmenší (nejméně definovaný) pevný bod programu P*. Jeden z nejvýznamnějších výsledků teorie pevných bodů programu říká, že *každý rekurzivní program P má právě jeden nejmenší pevný bod*; označujeme ho f_P.

Je-li dán rekurzivní program P, pak nás samozřejmě zajímá, která parciální funkce je jím určena. Odpověď na tuto otázku není apriorně jednoznačná. Je možno zaujmout jedno ze dvou hledisek:

1. Hledisko pevného bodu: Program P určuje nejmenší pevný bod f_P.
2. Hledisko výpočtu: Program P určuje funkci C_P, kterou získáme výpočtem podle programu P při použití jistého výpočetního pravidla C.[1]) V této kapitole přijmeme hledisko pevného bodu a v tomto smyslu budeme vyšetřovat vlastnosti rekurzivních programů. Toto stanovisko se zdá být opodstatněné, poněvadž mnoho praktických realizací rekurzivních programů[2]) skutečně vede k pevnému bodu.

5.1. FUNKCE A FUNKCIONÁLY

Budeme uvažovat n-ární parciální funkce $(n \geqq 0)$ zobrazující obor $D_1^n = D_1 \times D_1 \times \ldots \times D_1$ do oboru hodnot D_2; pro každou n-tici prvků $\langle a_1, a_2, \ldots, a_n\rangle \in D_1^n$ je $f(a_1, a_2, \ldots, a_n)$ buď nějaký prvek $b \in D_2$, anebo je nedefinováno. Například funkce podíl x/y zobrazující $R \times R$ do R (R je množina reálných čísel) se obvykle považuje za nedefinovanou pro všechny dvojice $\langle x, y\rangle$, kde $y = 0$. Nulární parciální funkce bez argumentů, tj. případ $n = 0$, je buď konstanta z D_2, anebo nedefinováno[3]). Za speciální případ budeme považovat i n-ární parciální predikáty, neboť jde v podstatě o n-ární parciální funkce zobrazující D_1^n do množiny $\{pravda, nepravda\}$[4]).

Při výstavbě teorie parciálních funkcí je výhodné zavést speciální prvek ω, reprezentující hodnotu *nedefinováno*. Předpokládáme při tom $\omega \notin D_2$ a klademe $D_2^* = D_2 \cup \{\omega\}$. Každou n-ární parciální funkci f zobrazující D_1^n do D_2 můžeme chápat jako totální funkci zobrazující D_1^n do D_2^*: Je-li f pro $\langle a_1, \ldots, a_n\rangle \in D_1^n$ nedefinováno, položíme $f(a_1, \ldots, a_n) = \omega$. Například funkci podíl chápeme jako zobrazení z $R \times R$ do R^*, přičemž pro všechna $x \in R$ je $x/0 = \omega$.

[1]) Jedno z možných výpočetních pravidel bylo použito při definici výpočetní posloupnosti v odst. 4.4.1, další budou zavedena v odst. 5.2.1. Již nyní je však třeba smířit se se zajímavým, ale možná překvapivým faktem, že výpočet podle daného rekurzivního programu lze definovat různými způsoby (zejména záleží na pořadí, v jakém jsou nahrazovány výskyty funkčních proměnných F_i) a že podle toho se mohou lišit i výsledky výpočtů. Pozn. překl.

[2]) To je volba výpočetních pravidel zabudovaných v kompilátorech ap. Pozn. překl.

[3]) Viz dále (srov. též čl. 4.4.1). Pozn. rec.

[4]) Pro jednoduchost výkladu se omezíme na funkce z D_1^n do D_2; všechny výsledky kapitoly však platí i pro obecný případ zobrazení z $A_1 \times A_2 \times \ldots \times A_n$ do $B_1 \times B_2 \times \ldots \times B_m$ $(n \geqq 0,\ m \geqq 1)$. Některé důkazy by však bylo třeba modifikovat. Jedinou výjimku v tomto omezení tvoří funkce *if-then-else* uvedená v př. 5.3.

Timto formálním obratem však není odstraněna hlavní potíž: co pokládat za hodnotu funkce, jejíž některé argumenty nejsou definovány. Taková situace může zcela přirozeně vzniknout, uvažujeme-li kompozici funkcí. Tak např. ke stanovení hodnoty $y/(x/0)$ potřebujeme znát hodnotu y/ω. Musíme proto rozšířit každou n-ární funkci zobrazující D_1^n do D_2^+ na n-ární funkci zobrazující $(D_1^+)^n = D_1^+ \times \ldots \times D_1^+$ do D_2^+, kde $D_1^+ = D_1 \cup \{\omega\}$.

Toto rozšíření je možné provést mnoha způsoby. Ukazuje se však, že všechna „rozumná" rozšíření vedou vždy k monotónním funkcím. V odst. 5.1.1 se budeme proto zabývat definicí a studiem třídy n-árních monotónních funkcí z $(D_1^+)^n$ do D_2^+; tuto třídu označíme $[(D_1^+)^n \to D_2^+]$. V odst. 5.1.2 zavedeme pojem funkcionálů zobrazujících $[(D_1^+)^n \to D_2^+]$ do $[(D_1^+)^n \to D_2^+]$; každý takový funkcionál τ přiřazuje každé funkci z $[(D_1^+)^n \to D_2^+]$ opět nějakou funkci z $[(D_1^+)^n \to D_2^+]$. Mezi funkcionály mají významnou roli ty, které jsou monotónní, resp. spojité[1]); i tyto pojmy budou definovány v odst. 5.1.2. V odst. 5.1.3 se seznámíme s pojmem pevného bodu a nejmenšího pevného bodu funkcionálu a tzv. Kleeneovou větou o tom, že každý spojitý funkcionál má (jediný) nejmenší pevný bod; tento výsledek bude východiskem našich úvah ve zbytku kapitoly.

5.1.1. Monotónní funkce

Částečné uspořádání množiny D^+. Abychom mohli definovat pojem monotonie, je třeba zavést na každém rozšířeném oboru $D^+ = D \cup \{\omega\}$ částečné uspořádání[2]) $\sqsubseteq$; toto uspořádání má odpovídat pojmu *„je méně nebo stejně definováno jako"*, a proto klademe

$$\omega \sqsubseteq d \text{ a } d \sqsubseteq d \text{ pro všechna } d \in D^+.$$

Je zřejmé, že dva různé prvky a, b z množiny D jsou v uspořádání $\sqsubseteq$ *nesrovnatelné*, tj. neplatí ani $a \sqsubseteq b$, ani $b \sqsubseteq a$. Částečné uspořádání $\sqsubseteq$ přeneseme i na množinu $(D^+)^n$:

$$\langle a_1, \ldots, a_n \rangle \sqsubseteq \langle b_1, \ldots, b_n \rangle \text{ právě když } a_i \sqsubseteq b_i \text{ pro všechna } i,\ 1 \leqq i \leqq n.$$

Příklad 5.1

1. Je-li $D = \{a, b\}$, pak $D^+ = \{a, b, \omega\}$ a v částečném uspořádání $\sqsubseteq$ jsou srovnatelné tyto prvky:

$$\omega \sqsubseteq \omega, \omega \sqsubseteq a, \omega \sqsubseteq b, a \sqsubseteq a, b \sqsubseteq b.$$

[1]) Náš zájem o takové funkcionály je motivován snahou vytvořit aparát, jehož pomocí by bylo možné exaktně definovat význam a zkoumat vlastnosti rekurzivních programů (s parciálními základními predikáty a funkcemi). Funkcionály takto využité v čl. 5.2 mají roli podobnou interpretovaným termům z definice rekurzivních schémat (odst. 4.4.1), s nimiž sdílejí i označení. Pozn. překl.

[2]) Připomeňme, že binární relace $\sqsubseteq$ je *částečným uspořádáním*, je-li reflexivní (tj. $\forall a[a \sqsubseteq a]$), antisymetrická (tj. $\forall a \forall b[a \sqsubseteq b \wedge b \sqsubseteq a \sqsupseteq a = b]$) a tranzitivní (tj. $\forall a \forall b \forall c[a \sqsubseteq b \wedge b \sqsubseteq c \sqsupset a \sqsubseteq c]$). Jak je zvykem, píšeme $a \sqsubset b$, jestliže $a \sqsubseteq b$ a $a \neq b$ (a není identické s b) a $a \not\sqsubseteq b$, jestliže $a \sqsubseteq b$ není *pravda*. (Na rozdíl od částečného uspořádání zavedeného v odst. 3.1.2 jde zde o „neostré", reflexivní částečné uspořádání. Pozn. rec.)

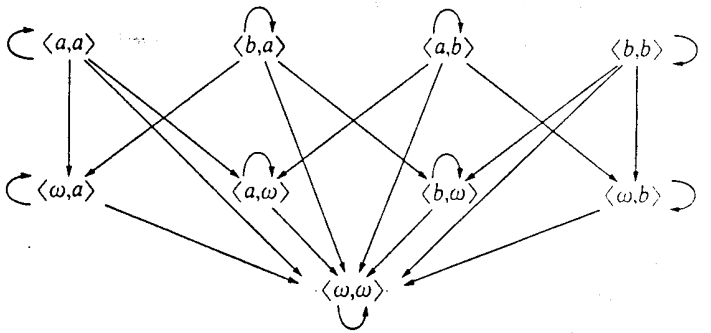
2. Je-li $D = \{a, b\}$, pak $D \times D = \{\langle a, a\rangle, \langle a, b\rangle, \langle b, a\rangle, \langle b, b\rangle\}$ a

$$D^+ \times D^+ = \{\langle \omega, \omega\rangle, \langle \omega, a\rangle, \langle a, \omega\rangle, \ldots, \langle a, b\rangle, \langle b, a\rangle, \langle b, b\rangle\}.$$

V částečném uspořádání $\sqsubseteq$ množiny $D^+ \times D^+$ je

$\langle \omega, \omega\rangle \sqsubseteq \langle \omega, \omega\rangle, \langle \omega, a\rangle, \langle a, \omega\rangle, \ldots, \langle a, b\rangle, \langle b, a\rangle, \langle b, b\rangle,$
$\langle a, \omega\rangle \sqsubseteq \langle a, \omega\rangle, \langle a, a\rangle, \langle a, b\rangle,$
$\langle \omega, a\rangle \sqsubseteq \langle \omega, a\rangle, \langle a, a\rangle, \langle b, a\rangle,$
$\langle \omega, b\rangle \sqsubseteq \langle \omega, b\rangle, \langle a, b\rangle, \langle b, b\rangle,$
$\langle b, \omega\rangle \sqsubseteq \langle b, \omega\rangle, \langle b, a\rangle, \langle b, b\rangle,$
$\langle a, a\rangle \sqsubseteq \langle a, a\rangle, \quad \langle a, b\rangle \sqsubseteq \langle a, b\rangle,$
$\langle b, a\rangle \sqsubseteq \langle b, a\rangle, \quad \langle b, b\rangle \sqsubseteq \langle b, b\rangle.$

Částečné uspořádání lze popsat i diagramem, v němž $B \sqsubseteq A$ je zachyceno jako $A \to B$; pro náš případ dostáváme:



■

Monotónní funkce. Říkáme, že n-ární funkce f z množiny $(D_1^+)^n$ do množiny D_2^+ je *monotónní*, jestliže pro všechna $\bar{x}, \bar{y} \in (D_1^+)^n$ ze vztahu $\bar{x} \sqsubseteq \bar{y}$ vyplývá vztah $f(\bar{x}) \sqsubseteq f(\bar{y})$. Intuitivně řečeno, funkce f je monotónní, kdykoliv z předpokladu, že vstupní hodnota $\bar{x}$ je méně nebo stejně definována jako vstupní hodnota $\bar{y}$, vyplývá, že i výstupní hodnota $f(\bar{x})$ je méně nebo stejně definována jako výstupní hodnota $f(\bar{y})$. Třídu všech monotónních funkcí zobrazujících $(D_1^+)^n$ do D_2^+ označíme $[(D_1^+)^n \to D_2^+]$.

Příklad 5.2

1. Identická n-ární funkce I přiřazující každému $\bar{x} \in (D_1^+)^n$ opět $\bar{x}$ je monotónní.

2. Totálně nedefinovaná n-ární funkce Ω přiřazující každému $\bar{x} \in (D_1^+)^n$ hodnotu ω, je monotónní.

3. Každá 0-ární funkce je monotónní.

4. Každá n-ární konstantní funkce, přiřazující každému $\bar{x} \in (D_1^+)^n$ pevný prvek $c \in D_2$, je monotónní.

■

Vlastnosti monotónních funkcí. Popíšeme nyní některé vlastnosti monotónních funkcí, které lze využít jako kritéria pro zjišťování, zda daná funkce zobrazující $(D_1^+)^n$ do D_2 je nebo není monotónní. (Důkazy všech těchto vlastností jsou jednoduché a jsou ponechány čtenáři.)

1. Je-li f unární funkce zobrazující D_1^+ do D_2^+ (tj. má-li f pouze jeden argument), pak f je monotónní, právě když

buď $f(\omega)$ je ω,

nebo $f(a)$ je c pro všechna $a \in D_1^+$.

Každou unární konstantní funkci f, kde $f(a)$ je c pro všechna $a \in D_1$, lze tedy rozšířit na monotónní funkci dvěma způsoby; tím, že položíme $f(\omega)$ rovno buď ω, nebo c.

2. Je-li n-ární funkce f ($n \geqq 2$) zobrazující $(D_1^+)^n$ do D_2^+ monotónní, pak

buď $f(\omega, \ldots, \omega)$ je ω,

nebo $f(a_1, \ldots, a_n)$ je c pro všechna $\langle a_1, \ldots, a_n \rangle \in (D_1^+)^n$.

Každá funkce splňující druhou z uvedených podmínek je monotónní; existují však funkce splňující první podmínku, které monotónní nejsou. Příkladem tu může být funkce podíl x/y nad množinou reálných čísel, rozšířená takto: x/ω je 0 pro všechna $x \in R$ a ω/y je ω pro všechna $y \in R^+$; je to zřejmé z toho, že $1/\omega \not\sqsubseteq 1/1$, ačkoliv $\langle 1, \omega \rangle \sqsubseteq \langle 1, 1 \rangle$.

Regulární rozšíření. Říkáme, že n-ární funkce f ($n \geqq 1$) zobrazující $(D_1^+)^n$ do D_2^+ je *regulárně rozšířená*, jestliže $f(a_1, \ldots, a_n)$ je ω, kdykoliv alespoň jedna z hodnot a_i je ω. Význam regulárního rozšíření je patrný z tohoto tvrzení:

Lemma (o regulárním rozšíření). *Každá regulárně rozšířená funkce je monotónní.*

Důkaz. Důkaz je veden sporem. Předpokládejme, že funkce $f(x_1, \ldots, x_n)$ je regulárně rozšířená, není však monotónní. Poněvadž není monotónní, existují n-tice $\langle a_1, \ldots, a_n \rangle$ a $\langle b_1, \ldots, b_n \rangle$ tak, že $\langle a_1, \ldots, a_n \rangle \sqsubseteq \langle b_1, \ldots, b_n \rangle$ (tj. $a_i \sqsubseteq b_i$ pro $i = 1, \ldots, n$), avšak $f(a_1, \ldots, a_n) \not\sqsubseteq f(b_1, \ldots, b_n)$. Z toho vyplývá, že $\langle a_1, \ldots, a_n \rangle \sqsubset \langle b_1, \ldots, b_n \rangle$, a proto pro jisté i_0 ($1 \leqq i_0 \leqq n$) $a_{i_0} \sqsubset b_{i_0}$: to nelze splnit jinak, než a_{i_0} je ω. Poněvadž f je regulárně rozšířená, také $f(a_1, \ldots, a_n)$ je ω. Z toho ale plyne, že $f(a_1, \ldots, a_n) \sqsubseteq f(b_1, \ldots, b_n)$ bez ohledu na hodnotu $f(b_1, \ldots, b_n)$, což je spor s předpokladem $f(a_1, \ldots, a_n) \not\sqsubseteq f(b_1, \ldots, b_n)$.

■

Příklad 5.3

1. *Funkce podíl* přiřazující uspořádané dvojici $\langle x, y \rangle \in R \times R$ číslo $x/y \in R$, rozšířená na totální funkci tak, že $x/0$ je ω pro všechna $x \in R$, je při regulárním rozšíření monotónní. (x/ω i ω/y je ω pro všechna $x, y \in R^+$.)

2. *Predikát rovnosti* zobrazující množinu $D \times D$ do množiny $\{pravda, nepravda\}$ lze rozšířit zejména těmito zajímavými způsoby:

(a) *Slabou rovností* = zobrazující $D^+ \times D^+$ do $\{pravda, nepravda\}^+$ dostaneme tak, že položíme $\omega = \omega$ a $\omega = d$ i $d = \omega$ rovno ω pro všechna $d \in D^+$. Jde o regulární rozšíření, takže predikát slabé rovnosti je monotónní.

(b) *Silnou rovností* $\equiv$ zobrazující $D^+ \times D^+$ do $\{pravda, nepravda\}^+$ dostaneme tak, že položíme $\omega \equiv \omega$ rovno *pravda* a $\omega \equiv d$ i $d \equiv \omega$ rovno *nepravda* pro všechna $d \in D$. Predikát silné rovnosti není monotónní funkce, jak je zřejmé např. z toho, že pro libovolné $d \in D$ platí $\langle \omega, d \rangle \sqsubseteq \langle d, d \rangle$, zatímco $(\omega \equiv d) \not\sqsubseteq (d \equiv d)$, tj. *nepravda* $\not\sqsubseteq$ *pravda*.

3. *Funkce def* zobrazující D^+ do $\{pravda, nepravda\}^+$, určená vztahy

def(*d*) je *pravda* pro $d \in D$,

def(ω) je *nepravda*,

není monotónní, neboť $\omega \sqsubseteq d$, avšak *def*(ω) $\not\sqsubseteq$ *def*(*d*), tj. *nepravda* $\not\sqsubseteq$ *pravda* ($d \in D$).

4. *Funkce if-then-else* zobrazující $\{pravda, nepravda\} \times D \times D$ do D je definována pro všechna $a, b \in D$ těmito vztahy:

(*if pravda then a else b*) je *a*,

(*if nepravda then a else b*) je *b*.

Tuto funkci lze rozšířit na monotónní funkci zobrazující $\{pravda, nepravda\}^+ \times D^+ \times D^+$ do D^+ např. tak, že položíme pro všechna $a, b \in D^+$

(*if pravda then a else* ω) je *a*,

(*if nepravda then* ω *else b*) je *b*,

(*if* ω *then a else b*) je ω.

Za povšimnutí stojí, že tato funkce je monotónní, ačkoliv to není regulární rozšíření funkce *if-then-else*. Dokázat, že uvažované rozšíření je monotónní, znamená ukázat, že (*if v then a else b*) $\not\sqsubseteq$ (*if v' then a' else b'*), kdykoliv $\langle v, a, b \rangle \sqsubseteq \langle v', a', b' \rangle$, to lze nejlépe určit rozlišením tří případů podle hodnoty v (*pravda*, *nepravda*, ω).

■

V dalším výkladu – pokud nebude výslovně uvedeno jinak – předpokládáme, že všechny uvažované základní funkce (jiné než konstanty a *if-then-else* z př. 5.3) jsou regulárně rozšířeny.

Kompozice monotónních funkcí. Důležitou operací nad funkcemi je jejich *kompozice* (*skládání*), která dovoluje, aby některé funkce byly definovány pomocí jiných, jednodušších. Je-li f funkce z množiny $(D_1^+)^n$ do množiny D_2^+ a g funkce z D_2^+ do D_3^+, pak *funkcí složenou* z f a g[1]) rozumíme funkci z $(D_1^+)^n$ do D_3^+ s hodnotou definovanou výrazem $g(f(\bar{x}))$ pro všechna $\bar{x} \in (D_1^+)^n$. Jsou-li f a g monotónní, je i tato složená funkce monotónní, neboť z $\bar{a} \sqsubseteq \bar{b}$ vyplývá $f(\bar{a}) \sqsubseteq f(\bar{b})$ (díky tomu, že f je monotónní) a odtud zase $g(f(\bar{a})) \sqsubseteq g(f(\bar{b}))$ (díky tomu, že g je monotónní).

[1]) V tomto pořadí. Pozn. překl.

Příklad 5.4

1. Funkce f zobrazující N^* do N^* určená výrazem

$f(x)$: *if* $x = 0$ *then* 1 *else* x

vznikne složením 0-árních funkcí 0, 1, identické funkce I, predikátu slabé rovnosti $=$ a funkce *if-then-else*. Poněvadž všechny tyto funkce jsou monotónní, je monotónní i funkce f.

2. Funkce f zobrazující N^* do N^* určená výrazem

$f(x)$: *if* $x \equiv \omega$ *then* 0 *else* 1

není monotónní, neboť $\omega \sqsubseteq 0$, avšak $f(\omega) \not\sqsubseteq f(0)$, tj. $0 \not\sqsubseteq 1$; všimněme si, že v definici funkce f byl použit nemonotónní predikát silné rovnosti.

3. Funkce f zobrazující N^* do N^* určená výrazem

$f(x)$: *if* $x = \omega$ *then* 0 *else* 1

vznikne složením pouze monotónních funkcí, a je proto sama také monotónní. Poněvadž, na rozdíl proti předcházejícímu případu, je užita slabá rovnost $=$, $f(\omega)$ je nyní ω a nikoliv 0 jako dříve: je zřejmé, že dokonce $f(x)$ je ω pro všechna $x \in N^-$.

■

Přenos monotónních funkcí za *if*. Nechť p, g, h_1 a h_2 jsou monotónní funkce.

Uvažujme funkce f_1 a f_2 určené výrazy

$f_1(\bar{x})$: $g(\textit{if } p(\bar{x}) \textit{ then } h_1(\bar{x}) \textit{ else } h_2(\bar{x}))$.
$f_2(\bar{x})$: *if* $p(\bar{x})$ *then* $g(h_1(\bar{x}))$ *else* $g(h_2(\bar{x}))$.

Obě tyto funkce jsou také monotónní, poněvadž jsou definovány jako kompozice monotónních funkcí.

Mezi funkcemi f_1 a f_2 existuje velmi zajímavý vztah[1]):

1. $f_2(\bar{x}) \sqsubseteq f_1(\bar{x})$ pro všechna $\bar{x} \in (D_1^*)^n$.

2. Jestliže $g(\omega)$ je ω, pak $f_2(\bar{x}) \equiv f_1(\bar{x})$ pro všechna $\bar{x} \in (D_1^*)^n$.

Obě tyto vlastnosti lze dokázat uvážením tří případů podle hodnot $p(\bar{x})$. Je-li $p(\bar{x})$ *pravda* nebo *nepravda*, je $f_2(\bar{x}) \equiv f_1(\bar{x})$; ve třetím případě $p(\bar{x})$ je ω a $f_2(\bar{x}) \sqsubseteq f_1(\bar{x})$, neboť $\omega \sqsubseteq g(\omega)$. Obecně je tedy $f_2(\bar{x}) \sqsubseteq f_1(\bar{x})$ pro všechna $\bar{x} \in (D_1^*)^n$: jestliže však ω je $g(\omega)$, je $f_2(\bar{x}) \equiv f_1(\bar{x})$ i v třetím případě a tento vztah pak také platí obecně.

Uvedené vlastnosti lze zobecnit[2]) i na případ n-ární monotónní funkce g ($n \geqq 1$).

[1]) Tvrzení ukazuji, že funkci g lze „přenést" dovnitř podmínky *if-then-else* při zachování úzkého vztahu mezi původní a výslednou funkcí. V originále se hovoří (na základě analogie s distributivními zákony z algebry) o distribuci g vzhledem k *if-then-else*. Pozn. překl.

[2]) Všimněme si však, že se v této formulaci nepožaduje monotónnost funkcí h_i, ani se netvrdí, že funkce f_1, f_2 jsou monotónní. Pozn. překl.

Definujeme-li

$f_1(\bar{x})$: $g(h_1(\bar{x}), \ldots, h_{i-1}(\bar{x}),$
$\textit{if } p(\bar{x}) \textit{ then } h_i'(\bar{x}) \textit{ else } h_i''(\bar{x}), h_{i+1}(\bar{x}), \ldots, h_n(\bar{x}))$

a

$f_2(\bar{x})$: *if* $p(\bar{x})$ *then* $g(h_1(\bar{x}), \ldots, h_{i-1}(\bar{x}), h_i'(\bar{x}), h_{i+1}(\bar{x}), \ldots, h_n(\bar{x}))$
else $g(h_1(\bar{x}), \ldots, h_{i-1}(\bar{x}), h_i''(\bar{x}), h_{i+1}(\bar{x}), \ldots, h_n(\bar{x}))$,

pak platí:

1. $f_2(\bar{x}) \sqsubseteq f_1(\bar{x})$ pro všechna $\bar{x} \in (D_1^*)^n$.

2. Jestliže $g(a_1, \ldots, a_{i-1}, \omega, a_{i+1}, \ldots, a_n)$ je ω pro všechny hodnoty $a_1, \ldots, a_{i-1}, a_{i+1}, \ldots, a_n$ (včetně ω), pak $f_2(\bar{x}) \equiv f_1(\bar{x})$ pro všechna $\bar{x} \in (D_1^*)^n$.

Tuto vlastnost v dalším výkladu často využijeme.

Supremum. Zavedeme nyní částečné uspořádání na množině $[(D_1^*)^n \to D_2^*]$ všech monotónních funkcí zobrazujících $(D_1^*)^n$ do D_2^* a dokážeme některé jeho vlastnosti.

1. Nechť $f, g \in [(D_1^*)^n \to D_2^*]$. Říkáme, že „*$f$ je méně nebo stejně definováno jako g*" a píšeme $f \sqsubseteq g$, jestliže $f(\bar{x}) \sqsubseteq g(\bar{x})$ pro všechna $\bar{x} \in (D_1^*)^n$. Speciálně je $\Omega \sqsubseteq f$ pro všechna $f \in [(D_1^*)^n \to D_2^*]$. Relace $\sqsubseteq$ je částečné uspořádání na množině $[(D_1^*)^n \to D_2^*]$.

2. Nechť $f, g \in [(D_1^*) \to D_2^*]$. Řekneme, že „*$f$ je rovno g*" a píšeme $f \equiv g$, jestliže $f(\bar{x}) \equiv g(\bar{x})$ pro všechna $\bar{x} \in (D_1^*)^n$; jinak řečeno $f \equiv g$, právě když $f \sqsubseteq g$ a $g \sqsubseteq f$.

3. Nechť $f_0, f_1, f_2, \ldots$ je posloupnost funkcí z $[(D_1^*)^n \to D_2^*]$; stručně ji zapíšeme $\{f_i\}$. Posloupnost $\{f_i\}$ se nazývá *řetězec*, jestliže $f_0 \sqsubseteq f_1 \sqsubseteq f_2 \equiv f_3 \sqsubseteq \ldots$.

4. Buď $\{f_i\}$ posloupnost funkcí z $[(D_1^*)^n \to D_2^*]$ a $f \in [(D_1^*)^n \to D_2^*]$. Říkáme, že f je *horní závora posloupnosti* $\{f_i\}$, jestliže $f_i \sqsubseteq f$ pro všechna $i \geqq 0$. Má-li navíc f tu vlastnost, že pro každou horní závoru g posloupnosti $\{f_i\}$ je $f \sqsubseteq g$, pak se f nazývá *nejmenší horní závora* čili *supremum posloupnosti* $\{f_i\}$ a označuje se $\sup\{f_i\}$. Z definice suprema vyplývá, že pokud existuje, je určeno jednoznačně: jestliže f a g jsou suprema posloupnosti $\{f_i\}$, musí být jak $f \sqsubseteq g$, tak $g \sqsubseteq f$, a tedy $f \equiv g$.

Lemma (o supremu). *Každý řetězec $\{f_i\}$ má supremum.*

Důkaz. Nechť $\bar{a}$ je libovolný prvek z $(D_1^*)^n$; uvažujme posloupnost $f_0(\bar{a})$, $f_1(\bar{a})$, $f_2(\bar{a})$, ... Poněvadž $\{f_i\}$ je řetězec, jsou pouze dvě možnosti: buď $f_i(\bar{a})$ je ω pro všechna $i \geqq 0$, nebo existuje $i_0 > 0$ a $b \in D_2$ tak, že $f_i(\bar{a})$ je ω pro všechna $i < i_0$ a $f_i(\bar{a})$ je b pro všechna $i \geqq i_0$. Je proto možné jednoznačně definovat funkci f tímto předpisem: pro každé $\bar{a} \in (D_1^*)^n$ nechť $f(\bar{a})$ je ω v prvním případě a $f(\bar{a})$ je b v druhém případě.

Snadno se ukáže, že f je monotónní, tj. že náleží do třídy $[(D_1^*)^n \to D_2^*]$. Ukážeme nyní, že f je $\sup\{f_i\}$.

1. f je horní mez posloupnosti $\{f_i\}$. Zvolme libovolné $\bar{a} \in (D_1^*)^n$ a $i \geqq 0$ a rozlišujme dva případy: (a) $f_i(\bar{a})$ je ω: v tomto případě evidentně $f_i(\bar{a}) \sqsubseteq f(\bar{a})$;

(b) $f_i(\bar{a})$ je b pro nějaké $b \in D_2$; v tomto případě je $f_i(\bar{a}) \equiv f(\bar{a})$, a tedy i $f_i(\bar{a}) \sqsubseteq f(\bar{a})$. Platí tedy $f_i \sqsubseteq f$ pro všechna $i \geqq 0$.

2. f je supremum posloupnosti $\{f_i\}$. Nechť g je libovolná horní závora posloupnosti $\{f_i\}$, tj. $f_i \sqsubseteq g$ pro všechna $i \geqq 0$. Zvolme nyní libovolné $\bar{a} \in (D_1^+)^n$ a rozlišujme dva případy:

(a) $f(\bar{a})$ je ω; v tomto případě evidentně $f(\bar{a}) \sqsubseteq g(\bar{a})$;

(b) $f(\bar{a})$ není rovno ω, a tedy $f(\bar{a}) \equiv f_{i_0}(\bar{a})$[1]: poněvadž $f_{i_0} \sqsubseteq g$, je i v tomto případě $f(\bar{a}) \sqsubseteq g(\bar{a})$. Je tedy $f \sqsubseteq g$.

■

Příklad 5.5

Posloupnost funkcí $f_0, f_1, f_2, \ldots$ z $[N^* \to N^*]$[2]) definovaná výrazem

$f_i(x)$: *if* $x < i$ *then* $x!$ *else* ω

tvoří řetězec, neboť $f_i \sqsubseteq f_{i+1}$ pro všechna i (přitom f_0 je Ω). Podle lemmatu o supremu má tedy posloupnost $\{f_i\}$ supremum: je jím funkce $x!$.

5.1.2. Spojité funkcionály

Funkcionál τ *nad* $[(D_1^+)^n \to D_2^+]$ je zobrazení množiny funkcí $[(D_1^+)^n \to D_2^+]$ do sebe: každé monotónní funkci f [zobrazující $(D_1^+)^n$ do D_2^+] jakožto argumentu přiřazuje tedy τ jistou monotónní funkci $\tau[f]$ [zobrazující opět $(D_1^+)^n$ do D_2^+].

U funkcionálů nás bude zajímat především jejich monotónnost a spojitost:

1. *Funkcionál* τ nad $[(D_1^+)^n \to D_2^+]$ se nazývá *monotónní*, jestliže pro všechna $f, g \in [(D_1^+)^n \to D_2^+]$ ze vztahu $f \sqsubseteq g$ vyplývá $\tau[f] \sqsubseteq \tau[g]$.

2. Monotónní funkcionál τ nad $[(D_1^+)^n \to D_2^+]$ se nazývá *spojitý*, jestliže pro každý řetězec funkcí $\{f_i\}$ je

$$\tau[\sup\{f_i\}] \equiv \sup\{\tau[f_i]\}.$$

Tato definice je korektní, poněvadž obě suprema existují: $\sup\{f_i\}$ existuje z toho důvodu, že $\{f_i\}$ je řetězec, tj. $f_0 \sqsubseteq f_1 \sqsubseteq f_2 \sqsubseteq \ldots$; poněvadž τ je monotonní, platí, že $\tau[f_0] \sqsubseteq \tau[f_1] \sqsubseteq \tau[f_2] \sqsubseteq \ldots$, tj. i $\{\tau[f_i]\}$ je řetězec. Lemma o supremu zaručuje tedy i existenci $\sup\{\tau[f_i]\}$.

Příklad 5.6

1. Identický funkcionál

$\tau[F]$: F

přiřazující každé funkci $f \in [(D_1^+)^n \to D_2^+]$ ji samu je zřejmě monotónní, neboť je-li $f \sqsubseteq g$, je i $\tau[f] \equiv f \sqsubseteq g \equiv \tau[g]$. Je také spojitý, protože $\tau[\sup\{f_i\}] \equiv \sup\{f_i\} \equiv \sup\{\tau[f_i]\}$.

---

[1]) Pro vhodné $i_0 \geqq 0$, podle definice funkce f. Pozn. překl.

[2]) Připomeňme, že N označuje množinu přirozených čísel. Pozn. překl.

2. Konstantní funkcionál

$\tau[F]$: h

přiřazující každé funkci $f \in [(D_1^+)^n \to D_2^+]$ jistou pevně zvolenou funkci h je monotónní, poněvadž z $f \sqsubseteq g$ vyplývá $\tau[f] \equiv h \equiv \tau[g]$. Je také spojitý, protože $\tau[\sup\{f_i\}] \equiv h \equiv \sup\{h\} \equiv \sup\{\tau[f]\}$.

■

Funkcionály obvykle definujeme pomocí kompozice známých monotónních funkcí a funkční proměnné F[1]): Jakmile za proměnnou F dosadíme zvolenou známou monotónní funkci f, $\tau[f]$ se určí pomocí pravidel kompozice funkcí; $\tau[f]$ je pak také monotónní[2]). Dokážeme nyní podstatně silnější tvrzení, které využijeme v dalších příkladech.

Věta 5.1 (Spojité funkcionály). *Každý funkcionál* τ, *složený z monotónních funkcí a funkční proměnné* F, *je spojitý.*

Důkaz provedeme indukcí podle struktury τ[3]). Jestliže funkcionál τ je tvořen jen symbolem F či h, kde h je jméno nějaké monotónní funkce, pak τ je spojitý (jak bylo ukázáno v př. 5.6). Při důkazu indukčního kroku rozlišíme dva případy:

1. $\tau[F]$ je $f(\tau_1[F], \ldots, \tau_n[F])$, kde f je nějaká monotónní funkce. Je třeba ukázat, že jsou-li funkcionály $\tau_1, \ldots, \tau_n$ spojité, je spojitý i funkcionál τ. Předně je vidět, že z $g \sqsubseteq h$ vyplývá vzhledem k monotónnosti funkcionálů $\tau_1, \ldots, \tau_n$ i $\tau_j[g] \sqsubseteq \tau_j[h]$, $1 \leqq j \leqq n$. Z monotónnosti funkce f pak plyne $f(\tau_1[g], \ldots, \tau_n[g] \sqsubseteq \sqsubseteq f(\tau_1[h], \ldots, \tau_n[h])$, takže opravdu $\tau[g] \sqsubseteq \tau[h]$, tj. τ je monotónní.

Dále musíme dokázat, že $\tau[\sup\{f_i\}] \equiv \sup\{\tau[f_i]\}$ pro libovolný řetězec $\{f_i\}$. Poněvadž pro libovolné $i \geqq 0$ je $f_i \sqsubseteq \sup\{f_i\}$, z monotónnosti funkcionálů $\tau_1, \ldots, \tau_n$ a funkce f plyne, že $\tau[f_i] \sqsubseteq \tau[\sup\{f_i\}]$. Platí tedy vztah $\sup\{\tau[f_i]\} \sqsubseteq \tau[\sup\{f_i\}]$.

Abychom dokázali opačnou nerovnost, zvolme $\bar{x}_0 \in (D_1^-)^n$. Podle definice τ a indukčního předpokladu máme

$$\begin{aligned}&\tau[\sup\{f_i\}](\bar{x}_0) \equiv \\ &\equiv f(\tau_1[\sup\{f_i\}](\bar{x}_0), \ldots, \tau_n[\sup\{f_i\}](\bar{x}_0)) \equiv \\ &\equiv f(\sup\{\tau_1[f_i]\}(\bar{x}_0), \ldots, \sup\{\tau_n[f_i]\}(\bar{x}_0)).\end{aligned}$$

Z konstrukce suprema podané v důkazu lemmatu o supremu (odst. 5.1.1) vyplývá, že ke každému j, $1 \leqq j \leqq n$, existuje přirozené číslo i_j takové, že pro všechna $k \geqq i_j$ je

$$\sup\{\tau_j[f_i]\}(\bar{x}_0) \equiv \tau_j[f_k](\bar{x}_0).$$

---

[1]) Zde a na řadě dalších míst autor směšuje rovinu syntaxe s rovinou interpretace a hovoří o funkcích tam, kde jde o jejich jména. Funkcionál je určen termem, který vznikne vzájemnou formální substitucí z funkčního symbolu F, jmen daných funkcí a predikátů a jejich argumentů. Pozn. rec.

[2]) Podle tvrzení o kompozici monotónních funkcí z odst. 5.1.1. Pozn. překl.

[3]) To znamená, že funkcionál je zadán pomocí jednodušších objektů, pro něž bude možné využít platnost indukčního předpokladu. Pozn. překl.

Buď i_0 největší z čísel $i_1, i_2, \ldots, i_n$. Pro všechna j, $1 \leqq j \leqq n$, pak platí

$$\sup \{\tau_j[f_i]\}(\bar{x}_0) \equiv \tau_j[f_{i_0}](\bar{x}_0),$$

a proto je

$$\begin{aligned} &f(\sup \{\tau_1[f_i]\}(\bar{x}_0), \ldots, \sup \{\tau_n[f_i]\}(\bar{x}_0)) \equiv \\ &\equiv f(\tau_1[f_{i_0}](\bar{x}_0), \ldots, \tau_n[f_{i_0}](\bar{x}_0)) \equiv \\ &\equiv f(\tau_1[f_{i_0}], \ldots, \tau_n[f_{i_0}])(\bar{x}_0). \end{aligned}$$

Poněvadž $\bar{x}_0$ byl zcela libovolný prvek z $(D_1^+)^n$, platí i obecně

$$\tau[\sup \{f_i\}] \sqsubseteq \sup \{\tau[f_i]\}.$$

2. $\tau[F]$ je $F(\tau_1[F], \ldots, \tau_n[F])$. Opět je třeba ukázat, že jsou-li τ_1, τ_n spojité, má tuto vlastnost i τ. Důkaz této části probíhá zcela obdobně jako v bodě 1: pouze místo monotónnosti funkce f se využívá monotónnost funkcí f_i a sup $\{f_i\}$. ∎

Příklad 5.7

1. Funkcionál τ nad $[N^- \to N^-]$ určený výrazem

$$\tau[F](x): \quad \textit{if } x = 0 \textit{ then } 1 \textit{ else } F(x+1)$$

je složen z monotónních funkcí (*if-then-else*, identická funkce, sčítání, slabá rovnost, nulární funkce 0 a 1) a funkční proměnné F; jde tedy o funkcionál spojitý.

2. Funkcionál τ nad $[N^- \to N^-]$ určený výrazem

$$\tau[F](x): \quad \textit{if } (\forall y \in N)[F(y) = y] \textit{ then } F(x) \textit{ else } \omega$$

je monotónní, nikoliv však spojitý: pro každou funkci z řetězce $f_0 \sqsubseteq f_1 \sqsubseteq f_2 \sqsubseteq \ldots$, kde

$$f_i(x): \quad \textit{if } x < i \textit{ then } x \textit{ else } \omega,$$

platí, že $\tau[F_i]$ je Ω pro všechna i, takže i sup $\{\tau[F_i]\}$ je Ω, zatímco sup $\{f_i\}$, a tedy i $\tau[\sup \{f_i\}]$ jsou identické funkce.

3. Funkcionál τ nad $[N^+ \to N^+]$ určený výrazem

$$\tau[F](x): \quad \textit{if } F(x) \equiv \omega \textit{ then } 0 \textit{ else } \omega$$

není ani monotónní. Abychom to nahlédli, uvažujme nulovou funkci Z, tj. funkci, pro kterou platí $Z(x) \equiv 0$ pro všechna $x \in N^+$: $\tau[Z]$ je Ω a $\tau[\Omega]$ je Z. Je tedy $\tau[\Omega] \not\sqsubseteq \tau[Z]$ (poněvadž $Z \not\sqsubseteq \Omega$), přestože $\Omega \sqsubseteq Z$. Tento výsledek je zajímavé srovnat s vlastnostmi funkcionálu

$$\tau'[F](x): \quad \textit{if } F(x) = \omega \textit{ then } 0 \textit{ else } \omega,$$

který je spojitý (i monotónní), poněvadž vznikne kompozicí z monotónních funkcí a funkční proměnné F. ∎

Uvažujme nyní nějaký funkcionál τ definovaný pomocí monotónních funkcí a funkční proměnné F tak, že v příslušném termu figurují dva různé výskyty symbolu F; funkcionál τ pak píšeme ve tvaru $\tau[F, F]$, který umožní tyto dva výskyty od sebe odlišit. Lze ukázat [viz cv. 5.2(c)], že τ je monotónní vzhledem ke kterémukoliv výskytu proměnné F, tj. že z $f \sqsubseteq g$ plyne pro každé h jednak $\tau[f, h] \sqsubseteq \tau[g, h]$, jednak $\tau[h, f] \sqsubseteq \tau[h, g]$ (standardní podmínku monotónnosti lze při tomto označení zapsat vztahem $\tau[f, f] \sqsubseteq \tau[g, g]$). Analogický výsledek platí i v obecném případě libovolného počtu výskytů funkční proměnné F.

5.1.3. Pevné body funkcionálů

Buď τ funkcionál nad $[(D_1^+)^n \to D_2^+]$. Říkáme, že funkce $f \in [(D_1^-)^n \to D_2^-]$ je *pevný bod funkcionálu* τ, jestliže $\tau[f] \equiv f$, tj. jestliže τ přiřazuje funkci f ji samu. Je-li f pevný bod funkcionálu τ a platí $f \sqsubseteq g$ pro všechny možné (další) pevné body g téhož funkcionálu τ, pak se f nazývá *nejmenší pevný bod funkcionálu* τ. Funkcionál nemusí mít žádné pevné body, může jich mít konečně nebo nekonečně mnoho, má však vždy nejvýše jeden nejmenší pevný bod: jestliže totiž f_1 i f_2 jsou nejmenší pevné body funkcionálu τ, pak $f_1 \sqsubseteq f_2$ a $f_2 \sqsubseteq f_1$, takže $f_1 \equiv f_2$.

Nechť τ je nějaký monotónní funkcionál nad $[(D_1^+)^n \to D_2^-]$. Uvažujme posloupnost funkcí $\tau^0[\Omega], \tau^1[\Omega], \tau^2[\Omega], \ldots$, kde $\tau^0[\Omega]$ je Ω (totálně nedefinovaná funkce) a pro všechna $i \geqq 0$ $\tau^{i+1}[\Omega]$ je $\tau[\tau^i[\Omega]]$. Každá funkce $\tau^i[\Omega]$ patří do $[(D_1^+)^n \to D_2^+]$; poněvadž $\Omega \sqsubseteq \tau[\Omega]$ a poněvadž τ je monotónní, tvoří posloupnost $\{\tau^i[\Omega]\}$ řetězec, tj.

$$\Omega \sqsubseteq \tau[\Omega] \sqsubseteq \tau^2[\Omega] \sqsubseteq \ldots$$

a podle lemmatu o supremu existuje sup $\{\tau^i[\Omega]\}$.

Příklad 5.8

1. Pro funkcionál τ nad $[N^+ \to N^+]$ určený výrazem

$$\tau[F](x): \quad \textit{if } x = 0 \textit{ then } 1 \textit{ else } x\,F(x-1)$$

je pro každé $i \geqq 0$

$$\tau^i[\Omega](x): \quad \textit{if } x < i \textit{ then } x! \textit{ else } \omega$$

a

$$\sup \{\tau^i[\Omega]\}(x): \quad x!.$$

Abychom to nahlédli, musíme ukázat, že pro všechna $i \geqq 0$ je $\tau[\tau^i[\Omega]] \equiv \tau^{i+1}[\Omega]$ (všimněme si, že $\tau^0[\Omega] \equiv \Omega$):

$$\begin{aligned} &\tau[\tau^i[\Omega]](x) \equiv \textit{if } x = 0 \textit{ then } 1 \textit{ else } x\,.\,(\textit{if } x - 1 < i \textit{ then } (x-1)! \textit{ else } \omega) \equiv \\ &\equiv \textit{if } x = 0 \textit{ then } 1 \textit{ else } (\textit{if } x < i+1 \textit{ then } x! \textit{ else } \omega) \equiv \\ &\equiv \textit{if } x < i+1 \textit{ then } x! \textit{ else } \omega \equiv \\ &\equiv \tau^{i+1}[\Omega](x). \end{aligned}$$

2. Definujme funkcionál τ nad $[N^- \to N^-]$ tímto výrazem:

$\tau[F](x)$: *if* $x > 100$ *then* $x - 10$ *else* $F(F(x + 11))$.

Pro $1 \leqq i \leqq 11$ je

$\tau^i[\Omega](x)$: *if* $x > 100$ *then* $x - 10$ *else if* $x > 101 - i$ *then* 91 *else* ω,

zatímco pro $x \geqq 11$ je

$\tau^i[\Omega](x)$: *if* $x > 100$ *then* $x - 10$ *else if* $x > 90 - 11(i - 11)$ *then* 91 *else* ω.

Všimněme si, že pro $i = 11$ jsou obě definice $\tau^i[\Omega]$ totožné. Dá se ukázat, že $\tau^{i+1}[\Omega] \equiv \tau[\tau^i[\Omega]]$ pro všechna $i \geqq 0$. Zřejmý je též tvar suprema uvažované posloupnosti[1]):

$\sup\{\tau^i[\Omega]\}(x)$: *if* $x > 100$ *then* $x - 10$ *else* 91

∎

V obou příkladech byla ilustrována nejen existence suprema, ale i jeho základní vlastnost ve smyslu tzv. Kleeneovy věty:

Věta 5.2 (První věta o rekurzi, Kleene). *Každý spojitý funkcionál* τ *má (jediný) nejmenší pevný bod* f_τ; *přitom platí, že* f_τ *je* $\sup\{\tau^i[\Omega]\}$.

Důkaz. Poněvadž τ je spojitý, je i monotónní, a proto posloupnost $\{\tau^i[\Omega]\}$ je řetězec; tedy má supremum, které označíme f_τ. Důkaz toho, že f_τ je nejmenší pevný bod, rozdělíme do dvou kroků.

1. f_τ je pevný bod funkcionálu τ. Poněvadž τ je spojitý, je

$$\tau[f_\tau] \equiv \tau[\sup\{\tau^i[\Omega]\}] \equiv \sup\{\tau^{i+1}[\Omega]\} \equiv \sup\{\tau^i[\Omega]\} \equiv f_\tau,$$

a tedy $\tau[f_\tau] \equiv f_\tau$.

2. Pro každý pevný bod g funkcionálu τ je $f_\tau \sqsubseteq g$. Nejprve dokážeme pomocí matematické indukce, že pro všechna $i \geqq 0$ je $\tau^i[\Omega] \sqsubseteq g$. Pro $i = 0$ je to zřejmé: $\tau^0[\Omega] \equiv \Omega \sqsubseteq g$. Jestliže $\tau^{i-1}[\Omega] \sqsubseteq g$ pro nějaké $i \geqq 1$, pak z předpokladu monotónnosti τ a z předpokladu, že g je pevný bod, plyne

$$\tau^i[\Omega] \equiv \tau[\tau^{i-1}[\Omega]] \sqsubseteq \tau(g) \equiv g.$$

Je tedy $\tau^i[\Omega] \sqsubseteq g$ pro všechna $i \geqq 0$ a g je horní závora posloupnosti $\{\tau^i[\Omega]\}$. Poněvadž f_τ je supremum této posloupnosti, je $f_\tau \sqsubseteq g$.

∎

Lze ukázat, že tvrzení věty 5.2 platí i pro libovolný monotónní (nikoliv nutně spojitý) funkcionál. Avšak, jak ukazuje příklad, monotónnost je pro platnost vět podstatná.

[1]) Srov. cv. 3.5. Pozn. překl.

Příklad 5.9

1. Funkcionál τ nad $[I^+ \to I^+]$ určený výrazem

$\tau[F](x)$: *if* $F(x) \equiv 0$ *then* 1 *else* 0

není monotonní a nemá žádné pevné body. Kdyby totiž f byl pevný bod funkcionálu τ, pak v případě $f(0) \equiv 0$ by bylo $\tau[f](0) \equiv 1$ a v případě $f(0) \not\equiv 0$ by bylo $\tau[f](0) \equiv 0$.

2. Funkcionál τ nad $[I^+ \to I^+]$ určený výrazem

$\tau[F](x)$: *if* $F(x) \equiv 0$ *then* 0 *else* 1

má dva pevné body, totiž konstantní funkce 0 a 1, nemá však žádný nejmenší pevný bod. Funkcionál τ není monotónní.

∎

V příkladech 5.10 až 5.12 uvedeme řadu funkcionálů, složených z monotónních funkcí a funkční proměnné F — všechny jsou proto spojité. Pro každý z nich popíšeme jeho nejmenší pevný bod: obecná metoda, jejíž pomocí lze ukázat, že jistá funkce f je nejmenším pevným bodem daného spojitého funkcionálu spočívá v sestrojení funkcí $\tau^i[\Omega]$ pro všechna $i \geqq 0$ a v důkazu vztahu $f \equiv \sup\{\tau^i[\Omega]\}$. Další metody budou popsány v čl. 5.3.

Příklad 5.10

1. Funkcionál τ nad $[(D_1^-)^n \to D_2^-]$ určený výrazem

$\tau[F](\bar{x})$: $F(\bar{x})$

má za svůj pevný bod každou funkci. Nejmenší pevný bod tedy tvoří funkce Ω.

2. Funkcionál τ nad $[(D_1^+)^n \to D_2^+]$ určený výrazem

$\tau[F](x)$: *if* $p(x)$ *then* $F(x)$ *else* $h(x)$,

kde p a h jsou libovolné pevné (regulárně rozšířené) funkce, má pevné body

if $p(x)$ *then* $g(x)$ *else* $h(x)$,

kde g je libovolná funkce z $[(D_1^+)^n \to D_2^+]$. Nejmenším pevným bodem funkcionálu τ je funkce

if $p(x)$ *then* $\Omega(x)$ *else* $h(x)$.

∎

Příklad 5.11

1. Funkcionál τ nad $[N^- \to N^-]$ určený výrazem

$\tau[F](x)$: *if* $x = 0$ *then* 1 *else* $F(x + 1)$

má tyto pevné body:

$f_0(x)$: *if* $x = 0$ *then* 1 *else* 0,
$f_1(x)$: *if* $x = 0$ *then* 1 *else* 1,
$f_2(x)$: *if* $x = 0$ *then* 1 *else* 2,
⋮
$f_n(x)$: *if* $x = 0$ *then* 1 *else* n,
⋮
$f(x)$: *if* $x = 0$ *then* 1 *else* ω

neboť pro všechna $n \in N^+$ (tedy i pro $n \equiv \omega$!) a $x \in N^+$ je

$$\begin{aligned}\tau[f_n](x) &\equiv \textit{if } x = 0 \textit{ then } 1 \textit{ else } (\textit{if } x + 1 = 0 \textit{ then } 1 \textit{ else } n) \equiv \\ &\equiv \textit{if } x = 0 \textit{ then } 1 \textit{ else } n \equiv \\ &\equiv f_n(x).\end{aligned}$$

Snadno lze ukázat, že žádné jiné pevné body neexistují. Nejmenší pevný bod tedy je

$f_\tau(x)$: *if* $x = 0$ *then* 1 *else* ω.

2. Funkcionál τ nad $[N^- \to N^-]$ určený výrazem

$\tau[F](x)$: *if* $x > 100$ *then* $x - 10$ *else* $F(F(x + 11))$

má jediný pevný bod, totiž

if $x > 100$ *then* $x - 10$ *else* 91,

který je tedy současně nejmenším pevným bodem f_τ; jde o tzv. „*funkci* 91" (viz př. 5.8 a cv. 3.5).

■

Příklad 5.12

Některé z pevných bodů funkcionálu τ nad $[I^+ \to I^+]$ určeného výrazem

$\tau[F](x, y)$: *if* $x = y$ *then* $y + 1$ *else* $F(x, F(x - 1, y + 1))$

jsou:

$f(x, y)$: *if* $x = y$ *then* $y + 1$ *else* $x + 1$,
$g(x, y)$: *if* $x \geqq y$ *then* $x + 1$ *else* $y - 1$,
$h(x, y)$: *if* $(x \geqq y) \wedge (x - y$ *sudé*$)$ *then* $x + 1$ *else* ω.

Funkce $f(x, y)$ je pevný bod funkcionálu τ, protože

$$\begin{aligned}\tau[f](x, y) &\equiv \textit{if } x = y \textit{ then } y + 1 \textit{ else } f(x, f(x - 1, y + 1)) \equiv \\ &\equiv \textit{if } x = y \textit{ then } y + 1 \textit{ else} \\ &\quad (\textit{if } x = [\textit{if } x - 1 = y + 1 \textit{ then } y + 2 \textit{ else } x] \\ &\quad \textit{then } [\textit{if } x - 1 = y + 1 \textit{ then } y + 2 \textit{ else } x] + 1 \textit{ else } x + 1) \equiv\end{aligned}$$

$$\begin{aligned}&\equiv \textit{if } x = y \textit{ then } y + 1 \textit{ else } (\textit{if } x = y + 2 \textit{ then } y + 3 \textit{ else } x + 1) \equiv \\ &\equiv \textit{if } x = y \textit{ then } y + 1 \textit{ else } x + 1 \equiv \\ &\equiv f(x, y).\end{aligned}$$

Při provádění naznačených úprav je třeba vždy pečlivě vyšetřit případ, kdy x či y je ω. Čtenáře mohlo např. napadnout, proč jsme volili $f(x, y)$ v uvedeném zdánlivě zbytečně složitém tvaru a neuvažovali místo toho $f'(x, y)$: $x + 1$. Důvod je právě v tom, že $f'(x, \omega)$ je $x + 1$, zatímco $\tau[f'(x, \omega)]$ je ω; f' tedy není pevný bod funkcionálu τ.

Funkce $g(x, y)$ je pevný bod funkcionálu τ, protože

$$\begin{aligned}\tau[g](x, y) &\equiv \textit{if } x = y \textit{ then } y + 1 \\ &\quad \textit{else } g(x, \textit{if } x - 1 \geqq y + 1 \textit{ then } (x - 1) + 1 \textit{ else } (y + 1) - 1) \equiv \\ &\equiv \textit{if } x = y \textit{ then } y + 1 \\ &\quad \textit{else } (\textit{if } x - 1 \geqq y + 1 \textit{ then } g(x, x) \textit{ else } g(x, y)) \equiv \\ &\equiv \textit{if } x = y \textit{ then } y + 1 \\ &\quad \textit{else } (\textit{if } x - 1 \geqq y + 1 \textit{ then } x + 1 \textit{ else } g(x, y)) \equiv \\ &\equiv \textit{if } x = y \vee x \geqq y + 2 \textit{ then } x + 1 \textit{ else } g(x, y) \equiv \\ &\equiv g(x, y).\end{aligned}$$

Podobně lze ukázat, že i $h(x, y)$ je pevný bod funkcionálu τ; je to dokonce nejmenší pevný bod funkcionálu τ.

■

5.2. REKURZÍVNÍ PROGRAMY

Po přípravných úvahách čl. 5.1 můžeme nyní zavést ústřední pojem této kapitoly. Budeme vyšetřovat třídu *rekurzivních programů*, z nichž každý je určen *rekurzivní definicí* tvaru[1])

$$F(\bar{x}) \Leftarrow \tau[F](\bar{x}),$$

kde $\tau[F](\bar{x})$ je funkcionál nad $[(D^-)^n \to D^-]$, vyjádřený pomocí kompozice známých monotónních funkcí a predikátů (nazývají se *základní funkce*, resp. *predikáty*) a funkční proměnné F, aplikovaných na individuové proměnné $\bar{x} = \langle x_1, x_2, \ldots, x_n \rangle$.

Význam daného programu, tj. funkci, která je jím určena, je ovšem třeba teprve stanovit; teprve pak bude zřejmá sémantika použitého jazyka. Dohodněme se proto, že *rekurzivním programem P*

$$P: \quad F(\bar{x}) \Leftarrow \tau[F](\bar{x})$$

je určena funkce f_τ (tj. nejmenší pevný bod funkcionálu τ)[2]) a označme ji v tomto kontextu symbolem f_P.

---

[1]) Obecnější rekurzívní programy, určené systémem rekurzívních definic, budou zkoumány v odst. 5.2.3.

[2]) Existence f_τ je zaručena větami 5.1 a 5.2.

Tak např. rekurzivní program P nad přirozenými čísly

$$P: \quad F(x) \Leftarrow \textit{if } x = 0 \textit{ then } 1 \textit{ else } x \,.\, F(x-1)$$

určuje funkci faktoriál [tj. $f_P(x)$ je $x!$], neboť $x!$ je nejmenší pevný bod funkcionálu

$$\tau[F](x): \quad \textit{if } x = 0 \textit{ then } 1 \textit{ else } x \,.\, F(x-1).$$

Aby mohl mít náš „programovací jazyk" praktický význam, je ovšem třeba popsat způsob, jak vyčíslit hodnoty funkce, definované daným rekurzívním programem. Za tím účelem zavedeme jistá *výpočetní pravidla* pro provádění rekurzívních programů. Je však třeba mít na paměti, že výpočetní pravidla by měla být formulována tak, aby funkce vyčíslovaná ve shodě s nimi daným rekurzívním programem P byla právě f_P. Výpočetní pravidla s touto vlastností nazveme *korektní*[1]).

5.2.1. Výpočetní pravidla

Buď $F(\bar{x}) \Leftarrow \tau[F](\bar{x})$ rekurzívní program nad nějakým oborem D. Pro libovolnou danou vstupní hodnotu $\bar{d} \in (D^+)^n$ proměnné $\bar{x}$ definujeme běh programu pomocí tzv. *výpočetní posloupnosti pro* $\bar{d}$ čili *výpočtu hodnoty* $F(\bar{d})$, tj. pomocí posloupnosti termů

$$t_0, t_1, t_2, \ldots$$

sestrojené takto:

1. První term t_0 je $F(\bar{d})$.

2. Pro každé i ($i \geqq 0$) se t_{i+1} získá z t_i ve dvou krocích:

(a) *Substituce:* některé výskyty (otázka, o které výskyty jde, bude rozvedena dále) symbolu F v termu t_i současně nahradíme výrazem $\tau[F]$:

(b) *Zjednodušení*[2]): základní funkce a predikáty nahradíme jejich hodnotami, kdykoliv je to možné (tento krok je opakován, až už nelze dále zjednodušovat).

Výpočetní posloupnost je konečná a jejím posledním členem je výsledný term t_k právě tehdy, když t_k již neobsahuje žádné výskyty symbolu F, tj. t_k je prvek z D^+.

Dané *výpočetní pravidlo* C specifikuje, které výskyty výrazu $F(\bar{e})$ mají být při provádění substituce nahrazeny výrazem $\tau[F](\bar{e})$. Zajímají nás při tom pouze „rozumná" výpočetní pravidla, která ke každému termu t_i (nějaké výpočetní posloupnosti) obsahujícímu ještě výskyty F, určí neprázdnou podmnožinu těchto výskytů, které budou při substituci nahrazeny. Je-li dáno výpočetní pravidlo C, pak uvažovaný rekurzívní program definuje parciální funkci C_P, zobrazující $(D^+)^n$ do D^+ takto: Je-li pro vstupní hodnotu $\bar{d} \in (D^+)^n$ výpočetní posloupnost pro $\bar{d}$ konečná, je $C_P(\bar{d})$ rovno t_k; je-li tato výpočetní posloupnost nekonečná, řekneme, že $C_P(\bar{d})$ je ω.

---

[1]) V originále „fixpoint rules", pravidla „pevného bodu". Ne všechna smysluplná výpočetní pravidla jsou korektní – viz dál. Pozn. překl.

[2]) Jindy se v této souvislosti mluví o „vyhodnocení". Pozn. překl.

Nejpřirozenější a dále uvažovaná jsou tato výpočetní pravidla[1]):

1. „*Nejlevější uvnitř*" (také: „*volání hodnotou*")[2]): Nahraď (pouze) ten nejlevější výskyt proměnné F, který ve svých argumentech již nemá žádné další výskyty F (je „nejvíce uvnitř" celého výrazu). Označme funkci definovanou pomocí tohoto výpočetního pravidla H_P.

2. „*Všechny uvnitř*": Nahraď současně všechny ty výskyty symbolu F, které ve svých argumentech nemají již žádné další výskyty F. Označme funkci definovanou pomocí tohoto výpočetního pravidla U_P.

3. „*Nejlevější*" (také: „*volání jménem*"): Nahraď (pouze) nejlevější výskyt symbolu F. Označme funkci definovanou pomocí tohoto výpočetního pravidla J_P.

4. „*Všechny vně*": Nahraď současně všechny ty výskyty symbolu F, které nejsou obsaženy v argumentech žádných dalších výskytů F (které jsou „vně dosahu" ostatních výskytů F). Označme funkci definovanou pomocí tohoto pravidla V_P.

5. „*Jednoduchý argument*": Nahraď současně všechny ty výskyty symbolu F, které mají aspoň jeden argument jednoduchý v tom smyslu, že neobsahuje žádný výskyt F. Označme funkci definovanou pomocí tohoto pravidla A_P.

6. „*Všechny*": Nahraď současně všechny výskyty proměnné F. Označme funkci definovanou pomocí tohoto pravidla X_P.[3])[4])

Příklad 5.13
V termu

$$F(0, F(1, 1)) + F(F(2, 2), F(3, 3))$$

označíme (šipkami) ty výskyty proměnné F, které budou podle jednotlivých výpočetních pravidel nahrazovány:

---

[1]) Doporučujeme čtenáři, aby současně s definicemi sledoval př. 5.13. Pozn. překl.

[2]) Termín „volání hodnotou", resp. „jménem" je znám uživatelům některých praktických programovacích jazyků (např. Algolu 60) většinou v souvislosti s náhradou formálních parametrů procedur skutečnými. K volání hodnotou se nejprve skutečný parametr vyhodnotí a teprve pak se hodnota substituuje do této procedury (pracuje se tedy nejprve s objektem, který je z hlediska této procedury „nejvíce uvnitř"), u volání jménem se provede náhrada formálního parametru „jménem", tj. symbolem skutečného parametru a tato substituce tak předchází vyhodnocení. Tyto mechanismy odpovídají vztahu mezi výpočetními pravidly 1 a 3. Pozn. překl.

[3]) Poněvadž některé ze zkratek odvozených z angličtiny se někdy v literatuře používají bez bližšího vysvětlení, uvedeme anglické ekvivalenty zavedených termínů a zkratek: 1. Leftmost – innermost (LI_P). 2. Parallel – innermost (PI_P). 3. Leftmost (L_P). 4. Parallel – outermost (PO_P). 5. Free-argument (FA_P). 6. Full-substitution (FS_P). Pozn. překl.

[4]) Formulace posledních dvou pravidel neurčuje jednoznačně zamýšlený způsob provedení náhrad. [Čtenář se o tom přesvědčí, když bude chtít provést „současně" obě náhrady např. v termu $F(x_1, F(x_1, x_2))$.] Jednoznačnosti se dosáhne induktivní definicí výsledku náhrad. Pozn. rec.

Nejlevější uvnitř: $F(0, F(1, 1)) + F(F(2, 2), F(3, 3))$
↑

Všechny uvnitř: $F(0, F(1, 1)) + F(F(2, 2), F(3, 3))$
↑ ↑ ↑

Nejlevější: $F(0, F(1, 1)) + F(F(2, 2), F(3, 3))$
↑

Všechny vně: $F(0, F(1, 1)) + F(F(2, 2), F(3, 3))$
↑ ↑

Jednoduchý argument: $F(0, F(1, 1)) + F(F(2, 2), F(3, 3))$
↑ ↑ ↑ ↑

Všechny: $F(0, F(1, 1)) + F(F(2, 2), F(3, 3))$
↑ ↑ ↑ ↑ ↑

Příklad 5.14
Rekurzivní program P nad oborem celých čísel[1])

$$F(x) \Leftarrow \textit{if } x > 100 \textit{ then } x - 10 \textit{ else } F(F(x + 11))$$

definuje (jako svůj nejmenší pevný bod) funkci

$$f_P(x): \quad \textit{if } x > 100 \textit{ then } x - 10 \textit{ else } 91.$$

Sestrojíme tři výpočetní posloupnosti pro $x = 99$, a to při použití tří různých výpočetních pravidel; výskyty symbolu F, za které bude prováděna náhrada jsou opět vyznačeny šipkami.

1. Při použití pravidla „všechny" dostáváme:

t_0 je $F(99)$
↑
$\textit{if } 99 > 100 \textit{ then } 99 - 10 \textit{ else } F(F(99 + 11))$

t_1 je $F(F(110))$
↑ ↑
$\textit{if}\,[\textit{if } 110 > 100 \textit{ then } 110 - 10 \textit{ else } F(F(110 + 11))] > 100$
$\textit{then}\,[\textit{if } 110 > 100 \textit{ then } 110 - 10 \textit{ else } F(F(110 + 11))] - 10$
$\textit{else } F(F([\textit{if } 110 > 100 \textit{ then } 110 - 10 \textit{ else } F(F(110 + 11))] + 11))$

t_2 je $F(F(111))$
↑ ↑
$\textit{if}\,[\textit{if } 111 > 100 \textit{ then } 111 - 10 \textit{ else } F(F(111 + 11))] > 100$
$\textit{then}\,[\textit{if } 111 > 100 \textit{ then } 111 - 10 \textit{ else } F(F(111 + 11))] - 10$
$\textit{else } F(F([\textit{if } 111 > 100 \textit{ then } 111 - 10 \textit{ else } F(F(111 + 11))] + 11))$

t_3 je 91.

[1]) Viz př. 5.11(2). Pozn. překl.

Výpočetní posloupnost tedy je

$$F(99) \to F(F(110)) \to F(F(111)) \to 91,$$
↑ ↑ ↑ ↑ ↑ ↑

tedy $X_P(99)$ je 91.

2. Při použití pravidla „nejlevější uvnitř" dostáváme:

t_0 je $F(99)$
↑
$\textit{if } 99 > 100 \textit{ then } 99 - 10 \textit{ else } F(F(99 + 11))$

t_1 je $F(F(110))$
↑
$F(\textit{if } 110 > 100 \textit{ then } 110 - 10 \textit{ else } F(F(110 + 11)))$

t_2 je $F(100)$
↑
$\textit{if } 100 > 100 \textit{ then } 100 - 10 \textit{ else } F(F(100 + 11))$

t_3 je $F(F(111))$
↑
$F(\textit{if } 111 > 100 \textit{ then } 111 - 10 \textit{ else } F(F(111 + 11)))$

t_4 je $F(101)$
↑
$\textit{if } 101 > 100 \textit{ then } 101 - 10 \textit{ else } F(F(101 + 11))$

t_5 je 91

čili stručně

$$F(99) \to F(F(110)) \to F(100) \to F(F(111)) \to F(101) \to 91.$$
↑ ↑ ↑ ↑ ↑

Tedy $H_P(99)$ je 91. Jak se snadno ukáže, v tomto případě vedou ke stejné výpočetní posloupnosti i pravidla „všechny uvnitř" a „jednoduchý argument", takže i $U_P(99)$ i $A_P(99)$ jsou 91.

3. Při použití pravidla „nejlevější" dostáváme:

t_0 je $F(99)$
↑

t_1 je $F(F(110))$
↑

t_2 je $\textit{if } F(110) > 100 \textit{ then } F(110) - 10 \textit{ else } F(F(F(110) + 11))$
↑

t_3 je $F(F(F(110) + 11))$
↑

t_4 je $\textit{if } F(F(110) + 11) > 100 \textit{ then } F(F(110) + 11) - 10$
↑
$\textit{else } F(F(F(F(110) + 11) + 11))$

t_5 je *if* $[$*if* $F(110) > 89$ *then* $F(110) + 1$ *else* $F(F(F(110) + 22))] > 100$
↑
then $F(F(110) + 11) - 10$ *else* $F(F(F(F(110) + 11) + 11))$

t_6 je *if* $F(110) > 99$ *then* $F(F(110) + 11) - 10$ *else* $F(F(F(F(110) + 11) + 11))$
↑

t_7 je $F(F(110) + 11) - 10$
↑

t_8 je $[$*if* $F(110) > 89$ *then* $F(110) + 1$ *else* $F(F(F(110) + 22))] - 10$
↑

t_9 je $F(110) - 9$
↑

t_{10} je 91.

Tedy i v tomto případě $J_P(99)$ je 91.

■

Výpočetní diagram. Než přejdeme k vyšetřování vztahu mezi C_P (tj. funkcí definovanou pomocí programu P a jistého výpočetního pravidla C) a f_P (tj. nejmenším pevným bodem programu P), zavedeme několik pojmů, které budou užitečné při dokazování vět odst. 5.2.2.

Nechť je dán rekurzivní program P tvaru

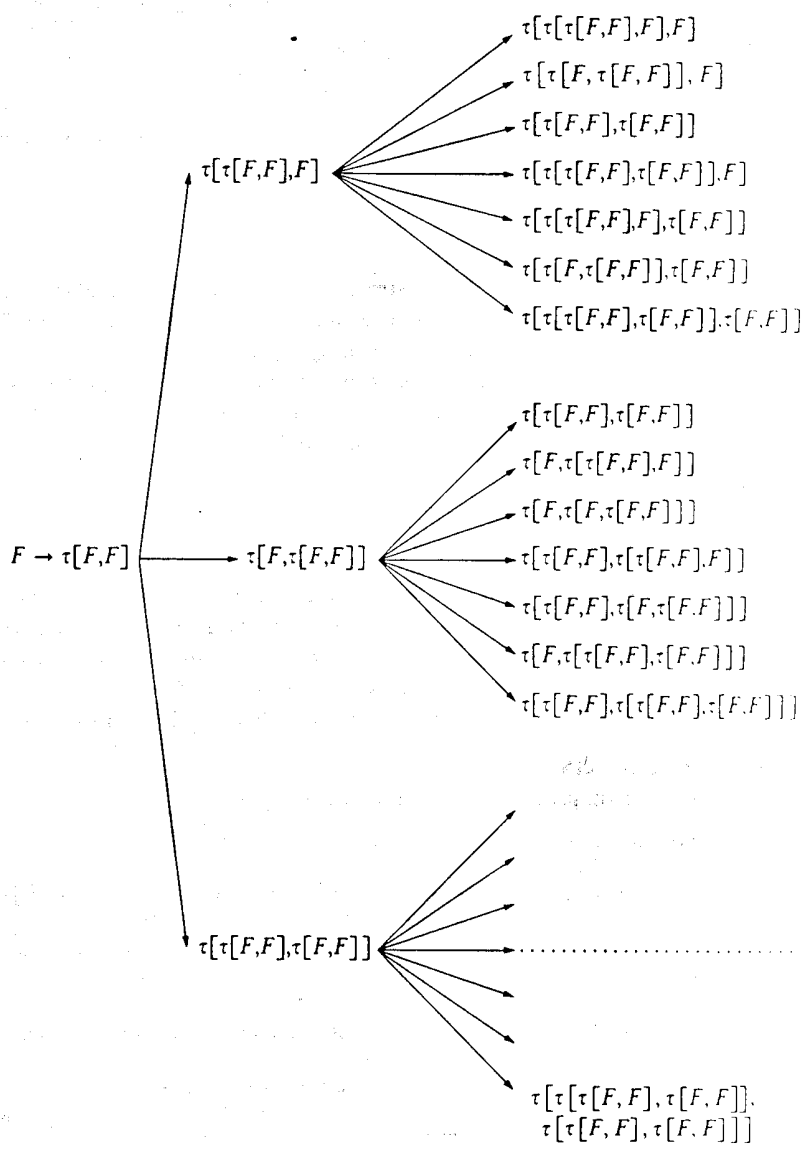
$$F(\bar{x}) \Leftarrow \tau[F](\bar{x}).$$

Předpokládejme pro jednoduchost, že ve funkcionálu τ jsou pouze dva výskyty symbolu F, které od sebe odlišíme formálním zápisem $\tau[F, F]$. Poněvadž τ je určen pomocí kompozice monotónních základních funkcí a predikátů a funkční proměnné F, musí být monotónní i vzhledem k oběma výskytům proměnné F, tzn. že z předpokladu $f \sqsubseteq g$ plyne $\tau[f, h] \sqsubseteq \tau[g, h]$ i $\tau[h, f] \sqsubseteq \tau[h, g]$ pro libovolnou funkci h a samozřejmě i $\tau[f, f] \sqsubseteq \tau[g, g]$[1]).

Podívejme se na diagram na str. 345, tzv. *výpočetní diagram funkcionálu* τ. Jde tu o *nekonečný strom*, jehož každý prvek (uzel) je term, který získáme z jeho předchůdce náhradou jednoho nebo více výskytů symbolu F výrazem $\tau[F, F]$. Z každého termu výpočetního diagramu lze získat jistou funkci tak, že nahradíme všechny výskyty F symbolem totálně nedefinované funkce Ω. Cesta diagramem taková, že všechny výskyty F ve všech termech na cestě jsou nahrazeny symbolem Ω, se nazývá Ω-*cesta*. Poněvadž τ je monotónní vzhledem k libovolnému výskytu F a poněvadž $\Omega \sqsubseteq \tau[\Omega, \Omega]$, platí vztahy

$$\tau[\Omega, \Omega] \sqsubseteq \tau[\tau[\Omega, \Omega], \Omega],$$
$$\tau[\Omega, \Omega] \sqsubseteq \tau[\Omega, \tau[\Omega, \Omega]],$$
$$\tau[\Omega, \Omega] \sqsubseteq \tau[\tau[\Omega, \Omega], \tau[\Omega, \Omega]]$$

[1]) O této vlastnosti funkcionálů byla zmínka již v závěru odst. 5.1.2. Pozn. překl.



atd.; každá Ω-cesta tedy udává jistý řetězec funkcí, např.

$$\Omega \sqsubseteq \tau[\Omega, \Omega] \sqsubseteq \tau[\Omega, \tau[\Omega, \Omega]] \sqsubseteq \tau[\Omega, \tau[\tau[\Omega, \Omega], \Omega]] \sqsubseteq \ldots .$$

Všimněme si, že pro „dolní" cestu diagramem[1])

$$F \to \tau[F, F] \to \tau[\tau[F, F], \tau[F, F]] \to \ldots,$$

kdy se v každém kroku provádí náhrada všech výskytů symbolů F výrazem $\tau[F, F]$, je odpovídající Ω-cesta dána řetězcem

$$\Omega \sqsubseteq \tau[\Omega] \sqsubseteq \tau^2[\Omega] \sqsubseteq \tau^3[\Omega] \sqsubseteq \ldots,$$

který byl použit při důkazu věty 5.2.

Mezi výpočetními posloupnostmi $t_0, t_1, t_2, \ldots$ daného rekurzivního programu $F(\bar{x}) \Leftarrow \tau[F](\bar{x})$, které odpovídají zvolenému výpočetnímu pravidlu a mezi Ω-cestami výpočetního diagramu funkcionálu τ existuje významný vztah:

Ke každému rekurzivnímu programu P: $F(\bar{x}) \Leftarrow \tau[F](\bar{x})$, prvku $\bar{d}$ a výpočetnímu pravidlu C existuje (konečná či nekonečná) cesta $C_P^0[F]$, $C_P^1[F]$, $C_P^2[F]$, ... výpočetním diagramem funkcionálu τ taková, že

$$C_P^i[F](\bar{d}) \equiv t_i$$

a[2])

$$\sup\{C_P^i[\Omega]\}(\bar{d}) \equiv C_P(\bar{d}).^3)$$

Při konstrukci cesty $C_P^i[F]$ vyjdeme z uzlu F a postupně nahrazujeme některé výskyty F výrazem $\tau[F, F]$. Které výskyty F mají být takto nahrazeny, je dáno výpočetní posloupností pro $\bar{d}$[4]): v každém kroku získáme $C_P^{i+1}[F]$ z $C_P^i[F]$ náhradou právě těch výskytů F, které byly pravidlem C vybrány pro konstrukci t_{i+1} z t_i. Také délka cesty je dána délkou výpočetní posloupnosti pro $\bar{d}$.

Příklad 5.15

Rekurzivní program nad oborem přirozených čísel

$$F(x) \Leftarrow \textit{if } x < 2 \textit{ then } 1 \textit{ else } [F(x-1) + F(x-2)]$$

(tzv. *Fibonacciův program*) zapíšeme v obecném tvaru $F(x) \Leftarrow \tau[F, F](x)$. Levá strana následující tabulky popisuje výpočet hodnoty $F(3)$ odpovídající výpočetnímu pravidlu „nejlevější"; v pravém sloupci je uvedena příslušná cesta výpočetním diagramem (jako dříve i nyní šipky označují nahrazované F)

[1]) Při níž v každém uzlu je sledována větev směřující co nejvíce dolů v uvedeném zobrazení diagramem. Pozn. překl.

[2]) Supremum konečného řetězce se vždy rovná poslednímu členu řetězce.

[3]) Důkaz výsledků tohoto druhu vyžaduje mj. podrobné vyšetření všech možných zjednodušení členů výpočetní posloupnosti [krok 2(b)]. Při úvahách o výpočetních posloupnostech a výpočetních diagramech je přitom na místě detailnější rozlišování mezi termy, jejich hodnotami a jejich označením na různých úrovních. Pozn. rec.

[4]) Některé z výskytů F v $C_P^i[F]$ nemusí být v termu t_i, protože byly eliminovány zjednodušováním během výpočtu.

$F(3)$	$C_P^0[F]$: F
$\to F(2) + F(1)$	$C_P^1[F]$: $\tau[F, F]$
$\to [F(1) + F(0)] + F(1)$	$C_P^2[F]$: $\tau[\tau[F, F], F]$
$\to [1 + F(0)] + F(1)$	$C_P^3[F]$: $\tau[\tau[\tau[F, F], F], F]$
$\to 2 + F(1)$	$C_P^4[F]$: $\tau[\tau[\tau[F, F], \tau[F, F]], F]$
$\to 3$	$C_P^5[F]$: $\tau[\tau[\tau[F, F], \tau[F, F]], \tau[F, F]]$

Prvky $C_P^i[F](x)$ této cesty tedy jsou tyto:

$C_P^0[F](x)$: $F(x)$

$C_P^1[F](x)$: $\textit{if } x < 2 \textit{ then } 1 \textit{ else } [F(x-1) + F(x-2)]$

$C_P^2[F](x)$: $\textit{if } x < 2 \textit{ then } 1 \textit{ else } \{\textit{if } x - 1 < 2 \textit{ then } 1 \textit{ else } [F(x-2) + F(x-3)] + F(x-2)\}$

$C_P^3[F](x)$:
$\textit{if } x < 2 \textit{ then } 1$
$\textit{else } \{\textit{if } x - 1 < 2 \textit{ then } 1$
$\textit{else } \{\textit{if } x - 2 < 2 \textit{ then } 1$
$\textit{else } [F(x-3) + F(x-4)] + F(x-3)\}$
$+ F(x-2)\}$

$C_P^4[F](x)$:
$\textit{if } x < 2 \textit{ then } 1$
$\textit{else } \{\textit{if } x - 1 < 2 \textit{ then } 1$
$\textit{else } \{\textit{if } x - 2 < 2 \textit{ then } 1$
$\textit{else } [F(x-3) + F(x-4)]$
$+ \textit{if } x - 3 < 2 \textit{ then } 1$
$\textit{else } [F(x-4) + F(x-5)]\}$
$+ F(x-2)\}$

$C_P^S[F](x)$:
if $x < 2$ *then* 1
else {*if* $x - 1 < 2$ *then* 1
else {*if* $x - 2 < 2$ *then* 1
else $[F(x-3) + F(x-4)]$
$+$ *if* $x - 3 < 2$ *then* 1
else $[F(x-4) + F(x-5)]$}
$+$ *if* $x - 2 < 2$ *then* 1 *else* $[F(x-3) + F(x-4)]$}

Hodnota $C_P^S[\Omega](3)$ je

if $3 < 2$ *then* 1
else {*if* $2 < 2$ *then* 1
else {$[$*if* $1 < 2$ *then* 1 *else* $\omega]$
$+$ $[$*if* $0 < 2$ *then* 1 *else* $\omega]$}
$+$ $[$*if* $1 < 2$ *then* 1 *else* $\omega]$},

tj. 3, což je ve shodě s dříve stanovenou hodnotou $F(3)$.

■

5.2.2. Korektní výpočetní pravidla

Výpočetní pravidlo C nazveme *korektním*, jestliže pro každý rekurzivní program P nad D a každé $\bar{d} \in (D^+)^n$ je

$$C_P(\bar{d}) \equiv f_P(\bar{d}).$$

První tři výpočetní pravidla z odst. 5.2.1, totiž „nejlevější uvnitř" (volání hodnotou), „všechny uvnitř" a „nejlevější" (volání jménem), korektní nejsou, jak je vidět z následujícího příkladu rekurzivního programu P, pro nějž $f_P \not\equiv H_P$, $f_P \not\equiv U_P$, $f_P \not\equiv J_P$.

Příklad 5.16
V rekurzivním programu P nad oborem celých čísel

$$F(x, y) \Leftarrow \textit{if } x = 0 \textit{ then } 0 \textit{ else } [F(x+1, F(x, y)) * F(x-1, F(x, y))]$$

označuje $*$ funkci násobení rozšířenou takto:

$0 * \omega$ a $\omega * 0$ je 0
$a * \omega$ a $\omega * a$ je ω (pro všechna $a \neq 0$)
$\omega * \omega$ je ω

Takto definované násobení představuje monotónní funkci a může proto být skutečně použito jako základní funkce v rekurzivním programu.

Nejmenším pevným bodem $f_P(x, y)$ je funkce $Z(x, y)$ taková, že $Z(x, y)$ je 0 pro všechna $x \in I$, $y \in I^+$ a $Z(\omega, y)$ je ω pro všechna $y \in I^+$; je tedy i $f_P(1, 0) = 0$.

Na druhé straně však výpočet hodnoty $F(1, 0)$ podle pravidla „všechny uvnitř" vede k nekonečné posloupnosti

$$\begin{aligned} F(1,0) &\to F(2, F(1,0)) * F(0, F(1,0)) \\ &\to F(2, F(2, F(1,0)) * F(0, F(1,0))) * F(0, F(2, F(1,0)) * F(0, F(1,0))) \\ &\to \ldots \end{aligned}$$

Také při použití pravidla „nejlevější uvnitř" a „nejlevější" jsou výpočetní posloupnosti při výpočtu hodnoty $F(1, 0)$ nekonečné. Hodnoty $H_P(1, 0)$, $U_P(1, 0)$, $J_P(1, 0)$ jsou proto všechny ω.

Jiná je situace při použití pravidla „všechny vně", kde výpočetní posloupnost má tvar

$$\begin{aligned} F(1,0) &\to F(2, F(1,0)) * F(0, F(1,0)) \\ &\to [F(3, F(2, F(1,0))) * F(1, F(2, F(1,0)))] * 0 \\ &= 0 \end{aligned}$$

I pravidla „jednoduchý argument" a „všechny" vedou k stejnému výsledku, totiž k hodnotě 0 (nikoliv ω). Podstatná tu ovšem je přijatá zásada, podle které jsme povinni zjednodušovat během procesu výpočtu všechny výrazy, kdykoliv je to jen možné. Můžeme-li stanovit hodnotu funkce na základě znalosti některých (nikoli nutně všech) jejích argumentů, musíme funkci touto hodnotou nahradit, aniž bychom čekali na vyhodnocení zbývajících argumentů. Term

$$[F(3, F(2, F(1,0))) * F(1, F(2, F(1,0)))] * 0$$

proto nahrazujeme povinně hodnotou 0[1]).

■

Výpočetní pravidlo C nemusí být korektní, přesto však obecně existuje důležitý vztah mezi C_P a f_P:

Věta 5.3 (Cadiou). *Pro libovolné výpočetní pravidlo* C_P *je funkce* C_P *méně nebo stejně definována jako nejmenší pevný bod* f_P, *tj.* $C_P \sqsubseteq f_P$[2]).

Důkaz. Buď P libovolný rekurzivní program $F(\bar{x}) \Leftarrow \tau[F, F](\bar{x})$ nad D, $\bar{d}$ libovolný prvek z množiny $(D^+)^n$, C libovolné výpočetní pravidlo a $C_P^0[F]$, $C_P^1[F]$, $C_P^2[F]$, ... příslušná cesta ve výpočetním diagramu funkcionálu τ. Uvažujme také „dolní" cestu v diagramu: $\tau^0[F]$, $\tau^1[F]$, $\tau^2[F]$, ... Poněvadž pro každé i lze $\tau^i[\Omega]$ získat z $C_P^i[\Omega]$ náhradou některých výskytů symbolu Ω výrazem $\tau[\Omega, \Omega]$, je

$$C_P^i[\Omega] \sqsubseteq \tau^i[\Omega] \quad \text{pro všechna} \quad i \geq 0.$$

[1]) Na základě definice funkce $*$ je totiž vždy $[\,] * 0$ rovno 0. Pozn. překl.
[2]) Připomeňme, že vztah $\sqsubseteq$ má za následek rovnost hodnot všude tam, kde výrazy na obou stranách jsou definovány: všechna výpočetní pravidla i všechny pevné body dávají proto tytéž výsledky, kdekoliv jsou definovány. Pozn. překl.

Poněvadž $\tau^i[\Omega] \sqsubseteq \sup\{\tau^i[\Omega]\}$, je tedy i $C_P^i[\Omega] \sqsubseteq \sup\{\tau^i[\Omega]\}$ pro všechna $i \geqq 0$; odtud vyplývá

$$\sup\{C_P^i[\Omega]\} \sqsubseteq \sup\{\tau^i[\Omega]\}.$$

Dokazovaný vztah $C_P(\bar{d}) \sqsubseteq f_P(\bar{d})$ je tedy důsledkem toho, že $C_P(\bar{d}) \equiv \sup\{C_P^i[\Omega]\}(\bar{d})$ a $f_P \equiv \sup\{\tau^i[\Omega]\}$ (viz věta 5.2).

■

Uvedeme nyní jistou postačující podmínku pro to, aby dané výpočetní pravidlo bylo korektní; poněvadž pravidla „všechny vně", „jednoduchý argument" a „všechny" splňují tuto podmínku, jsou korektní.

K formulaci hlavního výsledku budeme potřebovat ještě některé pomocné pojmy: nechť $\alpha[F^1, \ldots, F^i, F^{i+1}, \ldots, F^k](\bar{d})$ označuje nějaký term výpočetní posloupnosti pro $\bar{d}$; horní indexy tu rozlišují jednotlivé výskyty F v α (nemusí však odpovídat jejich pořadí). Předpokládejme, že k náhradě byly vybrány výskyty $F^1, \ldots, F^i$. Řekneme, že prováděná *substituce je bezpečná*, jestliže

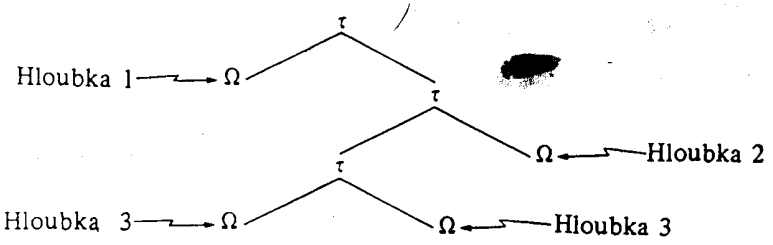
$$\alpha[\Omega, \ldots, \Omega, f_P, \ldots, f_P](\bar{d}) \equiv \omega.$$

Výpočetní pravidlo je bezpečné, jestliže všechny substituce, které jsou podle něj prováděny, jsou bezpečné.

Věta 5.4 (Vuillemin). *Každé bezpečné výpočetní pravidlo je korektní.*

Důkaz. Předpokládejme pro důkaz sporem existenci nějakého výpočetního pravidla C, které je bezpečné, není však korektní. Buď P rekurzivní program $F(\bar{x}) \Leftarrow \tau[F, F](\bar{x})$ nad D, pro nějž $C_P \not\equiv f_P$. Podle věty 5.3 je $C_P \sqsubseteq f_P$, takže není-li C_P korektní pravidlo, musí pro jisté $\bar{d} \in (D^-)^n$ být $C_P(\bar{d}) \equiv \omega$ a $f_P(\bar{d}) \not\equiv \omega$. Poněvadž podle věty 5.2 je $f_P(\bar{d}) \equiv \sup\{\tau^i[\Omega]\}(\bar{d})$, z $f_P(\bar{d}) \not\equiv \omega$ vyplývá $\tau^m[\Omega](\bar{d}) \not\equiv \omega$ pro vhodné $m \geqq 0$.

Pro potřeby tohoto důkazu vyjádříme prvky výpočetního diagramu funkcionálu τ ve tvaru binárního stromu, v němž každý list je označen symbolem Ω a ostatní uzly symbolem τ. Snadno pak zavedeme pojem hloubky výskytu Ω v daném termu; tak např. term $\tau[\Omega, \tau[\tau[\Omega, \Omega]], \Omega]]$ bude vyjádřen ve tvaru



Buď $C_P^0[F], C_P^1[F], C_P^2[F], \ldots$ cesta ve výpočetním diagramu funkcionálu τ, odpovídající zvolené vstupní hodnotě $\bar{d}$ a uvažovanému výpočetnímu pravidlu C. Poněvadž podle předpokladu $C_P(\bar{d}) \equiv \omega$, jsou dvě možnosti: buď je výpočetní posloupnost pro $\bar{d}$ konečná a její poslední term je ω, nebo je výpočetní posloupnost nekonečná. V prvním případě odvodíme spor snadno: nahradíme-li všechny výskyty F v termech výpočetní posloupnosti výrazem f_P, je každý term ekvivalentní svému předchůdci, přičemž první je $f_P(\bar{d})$ a poslední je ω. Je tedy $f_P(\bar{d}) \equiv \omega$, což odporuje předpokladu $f_P(\bar{d}) \not\equiv \omega$.

Ve druhém případě je cesta $C_P^0[F], C_P^1[F], C_P^2[F], \ldots$ nekonečná. Poněvadž každý term $C_P^i[F]$ obsahuje jen konečně mnoho výskytů symbolů F s hloubkou menší nebo rovnou m[1]), musí existovat N ($N \geqq 0$) takové, že $C_P^{N-1}[F]$ je získáno z $C_P^N[F]$ náhradou pouze symbolů F, jejichž hloubka je větší než m. Označme C_P^N výrazem $\alpha[F^1, \ldots, F^i, F^{i+1}, \ldots, F^k]$, kde $F^1, \ldots, F^i$ jsou právě ta F, která jsou nahrazována při přechodu k $C_P^{N-1}[F]$. Podívejme se nyní na term

$$\alpha[\Omega, \ldots, \Omega, \tau^m[\Omega], \ldots, \tau^m[\Omega]].$$

Všech prvních i výskytů F má hloubku větší než m a jsou nahrazeny symbolem Ω, zbývající výskyty F jsou nahrazeny výrazem $\tau^m[\Omega]$; i tyto výskyty symbolu Ω mají hloubku $\geqq m$. Je proto[2])

$$\tau^m[\Omega] \sqsubseteq \alpha[\Omega, \ldots, \Omega, \tau^m[\Omega], \ldots, \tau^m[\Omega]].$$

Dále, poněvadž $\tau^m[\Omega] \sqsubseteq f_P$ a α je monotónní vzhledem ke každému výskytu symbolu F, je

$$\alpha[\Omega, \ldots, \Omega, \tau^m[\Omega], \ldots, \tau^m[\Omega]] \sqsubseteq \alpha[\Omega, \ldots, \Omega, f_P, \ldots, f_P].$$

Máme tedy $\tau^m[\Omega] \sqsubseteq \alpha[\Omega, \ldots, \Omega, f_P, \ldots, f_P]$. Poněvadž $\tau^m[\Omega](\bar{d}) \not\equiv \omega$, je $\alpha[\Omega, \ldots, \Omega, f_P, \ldots, f_P](\bar{d}) \not\equiv \omega$, což odporuje našemu předpokladu, že C je bezpečné výpočetní pravidlo.

■

Důsledek. *Výpočetní pravidla „všechny vně", „jednoduchý argument"*[3]) *a „všechny" jsou bezpečná, a tedy i korektní.*

Důkaz. Místo formálního důkazu se spokojíme rozborem několika typických případů.

Abychom ukázali, že pravidlo „všechny vně" je bezpečné, všimněme si termu tvaru

$$g(a_1, \ldots, a_k, F(t_1), \ldots, F(t_n)),$$
$$\uparrow \qquad \uparrow$$

[1]) m bylo zavedeno v první části důkazu. Pozn. překl.

[2]) Připomeňme, že každá Ω-cesta výpočetním diagramem udává řetězec a že existuje cesta procházející uvažovanými termy. Pozn. překl.

[3]) K důkazu bezpečnosti pravidla „jednoduchý argument" je třeba předpokládat, že t_i není konstantní funkce.

kde g je základní monotónní funkce, $a_i \in D^+$ a termy t_i mohou obsahovat další výskyty F. Jestliže se zapsaný term objeví ve výpočetní posloupnosti, budou k další substituci vybrány právě ty výskyty F, které jsou uvedeny v zápisu. Z povinnosti zjednodušovat výrazy výpočetní posloupnosti kdykoli je to možné vyplývá, že

$$g(a_1, \ldots, a_k, \omega, \ldots, \omega) \equiv \omega.$$

Kdyby totiž bylo $g(a_1, \ldots, a_k, \omega, \ldots, \omega) \not\equiv \omega$, bylo by $g(a_1, \ldots, a_k, \omega, \ldots, \omega) \equiv d$ pro nějaké $d \in D$ a z monotónnosti g by plynulo, že $g(a_1, \ldots, a_k, b_1, \ldots, b_n) \equiv d$ pro libovolná $b_i \in D^+$. V tom případě jsme však byli povinni nahradit term

$$g(a_1, \ldots, a_k, F(t_1), \ldots, F(t_n))$$

hodnotou d již v některém dřívějším kroku výpočtu a tento term tedy nemůže být prvkem výpočetní posloupnosti. Je proto skutečně $g(a_1, \ldots, a_k, \omega, \ldots, \omega) \equiv \omega$, což je právě podmínka potřebná pro bezpečnost substituce. Poznamenejme ještě, že pro termy tvaru $F(\ldots)$ je substituce evidentně bezpečná.

Z dokázané bezpečnosti pravidla „všechny vně" vyplývá bezpečnost pravidla „všechny", neboť pochopitelně všechny výskyty F nahrazované podle prvního z nich jsou vždy obsaženy v množině výskytů F nahrazovaných podle druhého.

K důkazu bezpečnosti pravidla „jednoduchý argument" uvažujme term

$$F(F(t_1), F(t_2), \ldots, F(t_n)),$$
$$\uparrow \quad \uparrow \quad \uparrow$$

kde v žádném t_i se již F nevyskytuje. Nahrazovány budou právě vyznačené výskyty F, takže je třeba zkontrolovat hodnotu

$$f_P(\Omega(t_1), \Omega(t_2), \ldots, \Omega(t_n)).$$

Poněvadž f_P je monotónní a není to konstantní funkce (podle předpokladu uvedeného v pozn. pod čarou k formulaci důsledku) je $f_P(\omega, \omega, \ldots, \omega) \equiv \omega$, což je opět podmínka bezpečnosti substituce. [Viz ještě cv. 5.5(b).]

■

V př. 5.16 jsme viděli, že pravidla „nejlevější uvnitř", „všechny uvnitř" i „nejlevější" nejsou korektní. Protipříklad využíval základní funkci (násobení), která není regulárně rozšířená. Poněvadž většina používaných základních funkcí regulárně rozšířená je, zajímá nás přirozeně, jak vypadá situace v takovém případě. Platí:

Pro třídu rekurzivních programů používajících pouze regulárně rozšířené základní funkce (a funkci if-then-else) zůstávají pravidla „nejlevější uvnitř" (volání hodnotou) a „všechny uvnitř" nekorektní, zatímco pravidlo „nejlevější" (volání jménem) se za tohoto předpokladu stává bezpečným, a tedy korektním.

Abychom to nahlédli, uvažujme libovolný term výpočetní posloupnosti tvaru

$$g(F(t_1), F(t_2)),$$
$$\uparrow$$

kde g je regulárně rozšířená základní funkce a termy t_1, t_2 mohou obsahovat další výskyty symbolu F. Pravidlo „nejlevější" nahradí nejlevější výskyt F. Poněvadž g je regulárně rozšířená, je $g(\omega, f_P(t_2)) \equiv \omega$, a tato náhrada je tedy bezpečná. Současně je vidět, že bez dodatečného omezení na regulárně rozšířené funkce může být $g(\omega, f_P(t_2)) \not\equiv \omega$ (jak jsme viděli dříve) a substituce tedy nemusí být bezpečná. Jediná povolená funkce, u níž nevyžadujeme regulární rozšíření, je *if-then-else*; všimněme si však, že *if* ω *then* a *else* b je ω (pro všechna $a, b \in D^+$) a že pravidlo „nejlevější" nahrazuje ve výraze *if* x *then* y *else* z výskyt x, aniž by došlo k pokusu o vyhodnocení y či z. To je příčina toho, že pravidlo „nejlevější" je jediné, které je za uvedených podmínek bezpečné.

Že ani užití regulárně rozšířených základních funkcí nezaručí bezpečnost zbylých pravidel, je vidět z příkladu tohoto rekurzivního programu:

Příklad 5.17 (Morris).
Nejmenší pevný bod rekurzivního programu P nad oborem celých čísel

$$F(x, y) \Leftarrow \textit{if } x = 0 \textit{ then } 1 \textit{ else } F(x - 1, F(x, y))$$

je

$$f_P(x, y): \quad \textit{if } x \geqq 0 \textit{ then } 1 \textit{ else } \omega.$$

Výpočetní posloupnost pro $F(1, 0)$ podle pravidla „nejlevější uvnitř" je

$$F(1, 0) \to F(0, F(1, 0)) \to F(0, F(0, F(1, 0))) \to \ldots,$$
$$\uparrow \quad \uparrow \quad \uparrow$$

jde tedy o posloupnost nekonečnou a $H_P(1, 0)$ je ω. Obecně je

$$H_P(x, y): \quad \textit{if } x = 0 \textit{ then } 1 \textit{ else } \omega,$$

což je funkce méně definovaná než f_P. Také funkce U_P je identická s H_P, takže platí $H_P \not\equiv f_P$ i $U_P \not\equiv f_P$.

■

5.2.3. Systémy rekurzivních definic

Dosud uvažovaný „programovací jazyk" je značně omezený, zejména proto, že připouští v každém rekurzivním programu pouze jedinou funkční proměnnou F a jedinou rekurzivní definici. Rozšíříme proto možnosti tohoto jazyka a budeme uvažovat rekurzivní programy nad D určené systémem rekurzivních rovnic tvaru

$$F_1(\bar{x}) \Leftarrow \tau_1[F_1, \ldots, F_m](\bar{x}),$$
$$F_2(\bar{x}) \Leftarrow \tau_2[F_1, \ldots, F_m](\bar{x}),$$
$$\ldots\ldots\ldots\ldots\ldots\ldots$$
$$F_m(\bar{x}) \Leftarrow \tau_m[F_1, \ldots, F_m](\bar{x}),$$

kde každé τ_i je funkcionál[1]) nad $[(D^+)^n \to D^+]$ složený ze známých monotónních

[1]) Jde o zobecnění pojmu funkcionálu z odst. 5.1.2 na případ m funkčních argumentů. Na tyto funkcionály se pojem monotónnosti a spojitosti rozšíří (po souřadnicích). Pozn. rec.

základních funkcí a predikátů a funkčních proměnných $F_1, F_2, \ldots, F_m$ a individuových proměnných $\bar{x} = \langle x_1, x_2, \ldots, x_n \rangle$. Abychom se mohli zabývat takovými systémy rekurzivních definic, rozšíříme některé ze zavedených pojmů.

Řekneme, že *m-tice funkcí* $f = \langle f_1, \ldots, f_m \rangle$ *je monotónní*, je-li každá z funkcí f_i monotónní. Je-li $f = \langle f_1, \ldots, f_m \rangle$ a $g = \langle g_1, \ldots, g_m \rangle$, píšeme $f \sqsubseteq g$, právě když $f_i \sqsubseteq g_i$ pro všechna i, $1 \leqq i \leqq m$. Funkcionál $\tau[F_1, \ldots, F_m]$ nad $[(D_1^+)^n \to D_2^+]^m$, tj. zobrazení přiřazující m-tici funkcí $\langle f_1, \ldots, f_m \rangle$ m-tici funkcí $\langle g_1, \ldots, g_m \rangle$, kde všechna f_i, g_i jsou z $[(D_1^+)^n \to D_2^+]$, vyjádříme pomocí jednotlivých složek – funkcionálů $\tau_1, \ldots, \tau_m$; $\tau[F_1, \ldots, F_m]$ píšeme ve tvaru

$$\langle \tau_1[F_1, \ldots, F_m], \tau_2[F_1, \ldots, F_m], \ldots, \tau_m[F_1, \ldots, F_m] \rangle.$$

Řekneme, že *m-tice funkcí* $f = \langle f_1, \ldots, f_m \rangle$ *je pevným bodem funkcionálu* τ, jestliže $f_i = \tau_i[f_1, \ldots, f_m]$ pro všechna i, $1 \leqq i \leqq m$. Je-li f pevný bod funkcionálu τ a $f \sqsubseteq g$ pro libovolný jiný pevný bod téhož funkcionálu, řekneme, že f je jeho *nejmenší pevný bod*.

Funkcionál $\tau = \langle \tau_1, \ldots, \tau_m \rangle$ nad $[(D_1^-)^n \to D_2^-]^m$ se nazývá *spojitý*, je-li každý funkcionál τ_i spojitý. Pro takto zobecněné pojmy platí výsledky vět 5.1 a 5.2:

(1) *Každý funkcionál* τ *složený z monotónních funkcí a funkčních proměnných* $F_1, \ldots, F_m$ *je spojitý.*

(2) *Každý spojitý funkcionál* τ *má (jediný) nejmenší pevný bod; označujeme ho* $f_\tau = \langle f_{\tau_1}, \ldots, f_{\tau_m} \rangle$.

Příklad 5.18

Uvažujme funkcionál

$$\tau[F_1, F_2] = \langle \tau_1[F_1, F_2], \tau_2[F_1, F_2] \rangle$$

nad $[N^- \to N^-]^2$, kde

$$\tau_1[F_1, F_2](x): \quad \textit{if } x = 0 \textit{ then } 1 \textit{ else } [F_1(x - 1) + F_2(x - 1)]$$

a

$$\tau_2[F_1, F_2](x): \quad \textit{if } x = 0 \textit{ then } 0 \textit{ else } F_2(x + 1).$$

Pro libovolné $n \in N$ je pevným bodem funkcionálu τ dvojice $\langle g_n, h_n \rangle$, kde:

$$g_n(x): \quad \textit{if } x = 0 \vee x = 1 \textit{ then } 1 \textit{ else } [n(x - 1) + 1]$$

a

$$h_n(x): \quad \textit{if } x = 0 \textit{ then } 0 \textit{ else } n.$$

Platí totiž $g_n \equiv \tau_1[g_n, h_n]$ i $h_n \equiv \tau_2[g_n, h_n]$.

Nejmenším pevným bodem funkcionálu τ je dvojice

$$\langle \textit{if } x = 0 \vee x = 1 \textit{ then } 1 \textit{ else } \omega, \textit{ if } x = 0 \textit{ then } 0 \textit{ else } \omega \rangle.$$

■

Příklad 5.19

Funkcionál

$$\tau[F_1, F_2] = \langle \tau_1[F_1, F_2], \tau_2[F_1, F_2] \rangle$$

nad $[N^- \to N^-]^2$, kde

$$\tau_1[F_1, F_2](x): \quad \textit{if } x > 100 \textit{ then } x - 10 \textit{ else } F_1(F_2(x + 11))$$

a

$$\tau_2[F_1, F_2](x): \quad \textit{if } x > 100 \textit{ then } x - 10 \textit{ else } F_2(F_1(x + 11)),$$

má za svůj jediný (a tím i nejmenší) pevný bod dvojici[1])

$$\langle \textit{if } x > 100 \textit{ then } x - 10 \textit{ else } 91, \textit{ if } x > 100 \textit{ then } x - 10 \textit{ else } 91 \rangle.$$

■

Na případ systému rekurzivních definic lze bezprostředně přenést i pojem výpočetních pravidel. Všechny výsledky o vztahu mezi nejmenším pevným bodem a funkcemi, které získáme výpočtem podle zvoleného pravidla, zůstávají v platnosti.

Příklad 5.20

Podívejme se znovu[2]) na rekurzivní program nad přirozenými čísly

$$F_1(x) \Leftarrow \textit{if } x = 0 \textit{ then } 1 \textit{ else } [F_1(x - 1) + F_2(x - 1)],$$
$$F_2(x) \Leftarrow \textit{if } x = 0 \textit{ then } 0 \textit{ else } F_2(x + 1),$$

jehož nejmenší pevný bod tvoří dvojice

$$\langle \textit{if } x = 0 \vee x = 1 \textit{ then } 1 \textit{ else } \omega, \textit{ if } x = 0 \textit{ then } 0 \textit{ else } \omega \rangle.$$

Při výpočtu $F_1(2)$ pomocí pravidla „nejlevější" dostáváme

t_0 je $F_1(2)$
↑
$\textit{if } 2 = 0 \textit{ then } 1 \textit{ else } [F_1(2 - 1) + F_2(2 - 1)]$
t_1 je $F_1(1) + F_2(1)$
↑
$[\textit{if } 1 = 0 \textit{ then } 1 \textit{ else } [F_1(1 - 1) + F_2(1 - 1)]] + F_2(1)$
t_2 je $F_1(0) + F_2(0) + F_2(1)$
↑
$[\textit{if } 0 = 0 \textit{ then } 1 \textit{ else } [F_1(0 - 1) + F_2(0 - 1)]] + F_2(0) + F_2(1)$
t_3 je $1 + F_2(0) + F_2(1)$
↑
$1 + [\textit{if } 0 = 0 \textit{ then } 0 \textit{ else } F_2(0 + 1)] + F_2(1)$

[1]) Viz př. 5.14. Pozn. překl.
[2]) Viz př. 5.18. Pozn. překl.

t_4 je $1 + F_2(1)$

$\uparrow$

$1 + [\mathit{if}\ 1 = 0\ \mathit{then}\ 0\ \mathit{else}\ F_2(1 + 1)]$

t_5 je $1 + F_2(2)$

$\uparrow$

atd. Výpočet se nikdy nezastaví, a proto hodnota $F_1(2)$ je ω, což je ve shodě s hodnotou $f_{\tau_1}(2)$.

■

5.3. VERIFIKAČNÍ METODY

V tomto článku popíšeme některé induktivní metody pro dokazování vlastností nejmenších pevných bodů rekurzivních programů. Půjde především o metodu výpočetní indukce, která provádí důkaz indukcí podle úrovně rekurze, a o metodu strukturní indukce, což je v podstatě indukce podle hloubky datových struktur.

Na tyto dvě metody se zaměříme proto, že tvoří přirozený základ pro budoucí automatické systémy verifikace programů; většinu dalších známých metod můžeme získat jejich přímou aplikací.

5.3.1. Postupná výpočetní indukce[1])

Popíšeme nyní metodu, jak dokázat nějakou vlastnost nejmenšího pevného bodu f_P daného rekurzivního programu P pomocí indukce podle úrovně rekurze. Metodu vyložíme nejprve pro případ jednoduchých rekurzivních programů sestávajících z jediné rekurzivní definice; obecnějšími případy se budeme zabývat později.

Buď P rekurzivní program tvaru $F(\bar{x}) \Leftarrow \tau[F](\bar{x})$. Podle věty 5.2 má posloupnost $\{\tau^i[\Omega]\}$ za své supremum nejmenší pevný bod funkcionálu τ, tj.

$$\sup\{\tau^i[\Omega]\} \equiv f_\tau \equiv f_P.$$

Tento vztah naznačuje jistou možnost induktivního důkazu vlastností funkce f_P. Abychom dokázali, že f_P má jistou vlastnost φ [tj. že $\varphi(f_P)$ je *pravda*], stačí ukázat, že pro všechna $i \geqq 0$ je $\varphi(\tau^i[\Omega])$ *pravda* a že φ zůstane zachováno i při přechodu k supremu. Tento poslední dodatek je nutný, neboť obecně bohužel neplatí, že z pravdivosti $\varphi(\tau^i[\Omega])$ pro všechna i lze usoudit na pravdivost $\varphi(\sup\{\tau^i[\Omega]\})$, jak je vidět na tomto příkladu:

Příklad 5.21

Pro rekurzivní program nad přirozenými čísly

P: $F(x) \Leftarrow \mathit{if}\ x = 0\ \mathit{then}\ 1\ \mathit{else}\ x \,.\, F(x - 1)$

[1]) V orig. „stepwise computational induction". Pozn. překl.

je pro všechna $i \geqq 0$ (viz př. 5.8)

$$\tau^i[\Omega](x): \quad \mathit{if}\ x < i\ \mathit{then}\ x!\ \mathit{else}\ \omega$$

a

$$\sup\{\tau^i[\Omega]\}(x): \quad x!\,.$$

Zvolíme-li predikát φ

$$\varphi(F): \quad (\exists x \in N)[F(x) \equiv \omega],$$

pak $\varphi(\tau^i[\Omega])$ je *pravda* pro všechna $i \geqq 0$, avšak $\varphi(\sup\{\tau^i[\Omega]\})$ neplatí, poněvadž pro všechna $x \in N$ je $x! \not\equiv \omega$.

■

Řekneme, že *predikát $\varphi(F)$ je přípustný*, jestliže pro každý spojitý funkcionál τ z toho, že pro všechna $i \geqq 0$ platí $\varphi(\tau^i[\Omega])$, vyplývá, že platí i $\varphi(\sup\{\tau^i[\Omega]\})$. Vyslovíme a dokážeme nyní lemma, které zaručí přípustnost predikátů, které budeme používat v příkladech. (Ve cv. 5.11 je čtenář žádán, aby dokázal přípustnost mnohem širší třídy predikátů[1]).)

Lemma (o přípustnosti predikátů). *Každý predikát tvaru* $\bigwedge_{i=1}^{n} (\alpha_i[F] \sqsubseteq \beta_i[F])$, *tj. tvaru konečné konjunkce nerovností* $\sqsubseteq$, *kde* α_i *a* β_i *jsou spojité funkcionály, je přípustný.*

Důkaz. Předpokládejme $\varphi(F)$ ve tvaru $\alpha[F] \sqsubseteq \beta[F]$ a nechť $\alpha[\tau^i[\Omega]] \sqsubseteq \beta[\tau^i[\Omega]]$ pro všechna $i \geqq 0$. Poněvadž $\tau^i[\Omega] \sqsubseteq \sup\{\tau^i[\Omega]\}$, z monotónnosti funkcionálu β vyplývá, že $\beta[\tau^i[\Omega]] \sqsubseteq \beta[\sup\{\tau^i[\Omega]\}]$ a tím i $\alpha[\tau^i[\Omega]] \sqsubseteq \beta[\sup\{\tau^i[\Omega]\}]$ pro všechna $i \geqq 0$. Je tedy $\beta[\sup\{\tau^i[\Omega]\}]$ horní závora posloupnosti $\{\alpha[\tau^i[\Omega]]\}$, a proto $\sup\{\alpha[\tau^i[\Omega]]\} \sqsubseteq \beta[\sup\{\tau^i[\Omega]\}]$. Poněvadž α je spojitý funkcionál, je $\alpha[\sup\{\tau^i[\Omega]\}] \equiv \sup\{\alpha[\tau^i[\Omega]]\}$, takže

$$\alpha[\sup\{\tau^i[\Omega]\}] \sqsubseteq \beta[\sup\{\tau^i[\Omega]\}].$$

Tento vztah ovšem neříká nic jiného, než že platí $\varphi(\sup\{\tau^i[\Omega]\})$. Zobecnění na případ, kdy φ je konečná konjunkce nerovností, je zřejmé.

■

Na důležitý případ (silné) rovnosti $\alpha[F] \equiv \beta[F]$, kde α a β jsou spojité funkcionály, se tvrzení lemmatu také vztahuje, neboť tento predikát lze vyjádřit ve tvaru $(\alpha[F] \sqsubseteq \beta[F]) \wedge (\beta[F] \sqsubseteq \alpha[F])$.

Přímo z definice přípustnosti predikátu vyplývá důležitá věta:

Věta 5.5 (Postupná výpočetní indukce – de Bakker, Scott). *Je-li P rekurzivní program P: $F(\bar{x}) \Leftarrow \tau[F](\bar{x})$ a $\varphi(F)$ přípustný predikát a platí-li $\varphi(\Omega)$ a $\forall f[\varphi(f) \supset \varphi(\tau[f])]$, pak platí i $\varphi(f_P)$.*

[1]) Tam popsané. Pozn. překl.

Kvantifikátor $\forall f$ ve formulaci věty je vztažen na všechny funkce z $[(D^-)^n \to D^-]$, kde D je obor programu P.

Příklad 5.22

Uvažujme rekurzivní program

P: $F(x) \Leftarrow$ *if* $p(x)$ *then* x *else* $F(F(h(x)))$.

Pro naše účely není nutné specifikovat blíže ani obor D, ani význam symbolů p a h; budeme však předpokládat – a to i v dalších příkladech – že p a h jsou regulárně rozšířeny, tj. že $p(\omega)$ i $h(\omega)$ je ω. Chceme ukázat, že

$$f_P f_P \equiv f_P,$$

tj.[1])

$$\forall x[f_P(f_P(x)) \equiv f_P(x)].$$

Položme

$$\varphi(F): \quad \forall x[f_P(F(x)) \equiv F(x)]$$

a dokažujme $\varphi(f_P)$ pomocí výpočetní indukce[2]). Všimněme si ještě, že $f_P(\omega)$ je ω, neboť:

$f_P(\omega) \equiv$ *if* $p(\omega)$ *then* ω *else* $f_P(f_P(h(\omega))) \equiv$	f_P je pevný bod!
$\equiv$ *if* ω *then* ω *else* $f_P(f_P(\omega)) \equiv$	neboť $p(\omega)$ je ω a $h(\omega)$ je ω
$\equiv \omega$	podle definice hodnoty výrazu (*if* ω *then* a *else* b)

Krok 1. $\varphi(\Omega)$: $\forall x[f_P(\Omega(x)) \equiv \Omega(x)]$. Důkaz plyne z toho, že pro každé x je $f_P(\Omega(x)) \equiv f_P(\omega) \equiv \omega \equiv \Omega(x)$.

Krok 2. $\forall f[\varphi(f) \supset \varphi(\tau[f])]$. Z indukčního předpokladu $\forall x[f_P(f(x)) \equiv f(x)]$ je třeba dokázat $\forall x[f_P(\tau[f](x)) \equiv \tau[f](x)]$. To však plyne z toho, že pro všechna x

$f_P(\tau[f](x)) \equiv$	
$\equiv f_P($*if* $p(x)$ *then* x *else* $f(f(h(x))))) \equiv$	definice τ
$=$ *if* $p(x)$ *then* $f_P(x)$ *else* $f_P(f(f(h(x)))) \equiv$	přenos f_P za *if* (viz odst. 5.1.1)
$\equiv$ *if* $p(x)$ *then* (*if* $p(x)$ *then* x *else* $f_P(f_P(h(x))))$ *else* $f_P(f(f(h(x)))) \equiv$	definice f_P
$\equiv$ *if* $p(x)$ *then* x *else* $f_P(f(f(h(x)))) \equiv$	zjednodušení
$\equiv$ *if* $p(x)$ *then* x *else* $f(f(h(x))) \equiv$	indukční předpoklad
$\equiv \tau[f](x)$	definice τ

∎

Příklad 5.23

Pro rekurzivní program

P: $F(x, y) \Leftarrow$ *if* $p(x)$ *then* y *else* $h(F(k(x), y))$

chceme dokázat platnost vztahu

$$\forall x \forall y[h(f_P(x, y)) \equiv f_P(x, h(y))].$$

Položme

$$\varphi(F): \quad \forall x \forall y[h(F(x, y)) \equiv F(x, h(y))].$$

Poněvadž $h(F(x, y))$ a $F(x, h(y))$ definují spojité funkcionály, je $\varphi(F)$ přípustný predikát.

Krok 1. $\varphi(\Omega)$: $\forall x \forall y[h(\Omega(x, y)) \equiv \Omega(x, h(y))]$. Důkaz je zřejmý, neboť $h(\omega)$ je ω.

Krok 2. $\forall f[\varphi(f) \supset \varphi(\tau[f])]$. Z indukčního předpokladu $\forall x \forall y[h(f(x, y)) \equiv f(x, h(y))]$ je třeba dokázat $\forall x \forall y[h(\tau[f](x, y)) \equiv \tau[f](x, h(y))]$. To však plyne z toho, že:

$h(\tau[f](x, y)) \equiv$	
$\equiv h($*if* $p(x)$ *then* y *else* $h(f(k(x), y))) \equiv$	
$\equiv$ *if* $p(x)$ *then* $h(y)$ *else* $h(h(f(k(x), y))) \equiv$	
$\equiv$ *if* $p(x)$ *then* $h(y)$ *else* $h(f(k(x), h(y))) \equiv$	indukční předpoklad
$\equiv \tau[f](x, h(y))$.	

∎

Příklad 5.24

Rekurzivní program nad Σ^* (kde Σ je daná konečná abeceda)

P: $F(x, y) \Leftarrow$ *if* $x = \Lambda$ *then* y *else* *začátek*(x) . $F($*zbytek*$(x), y)$

definuje funkci *zřetězení* $f_P(x, y)$, zde označenou $x*y$; viz čl. 1.2 a odst. 1.4.1. Například $ab*cd$ je $abcd$. Chceme dokázat, že funkce $*$ je asociativní, tj. že

$$\forall x \forall y \forall z[(x*y)*z \equiv x*(y*z)].$$

Položme

$$\varphi(F): \quad \forall x \forall y \forall z[F(x, y)*z \equiv F(x, y*z)].$$

Protože výrazy $F(x, y)*z$ i $F(x, y*z)$ vznikly kompozicí funkční proměnné F a monotónní funkce $*$, definují spojité funkcionály a $\varphi(F)$ je přípustný predikát. V důkazu využijeme ještě skutečnost, že pro všechna z je $\omega*z$ rovno ω, neboť

$$\omega*z \equiv \textit{if } \omega = \Lambda \textit{ then } z \textit{ else začátek}(\omega) . (\textit{zbytek}(\omega)*z) \equiv$$
$$\equiv \textit{if } \omega \textit{ then } z \textit{ else začátek}(\omega) . (\textit{zbytek}(\omega)*z) \equiv$$
$$\equiv \omega.$$

Krok 1. $\varphi(\Omega)$: $\forall x \forall y \forall z[\Omega(x, y)*z \equiv \Omega(x, y*z)]$. Důkaz vyplývá z toho, že $\Omega(x, y)*z \equiv \omega*z \equiv \omega \equiv \Omega(x, y*z)$.

Krok 2. $\forall f[\varphi(f) \supset \varphi(\tau[f])]$: Z indukčního předpokladu $\forall x \forall y \forall z[f(x, y)*z \equiv f(x, y*z)]$ je třeba dokázat $\forall x \forall y \forall z[\tau[f](x, y)*z \equiv \tau[f](x, y*z)]$. To však plyne z toho, že:

[1]) Kvantifikátor $\forall x$ se vztahuje na všechna $x \in D^-$ (nejen $x \in D$); v tomto smyslu je třeba chápat univerzální kvantifikátory v celém zbytku kapitoly.

[2]) Poznamenejme, že zvolený tvar φ je podstatný; důkaz nepracuje pro $\varphi'(F)$: $\forall x[F(F(x)) \equiv F(x)]$.

$\tau[f](x, y)*z \equiv$
$\equiv (\text{if } x = \Lambda \text{ then } y \text{ else začátek}(x).f(\text{zbytek}(x), y))*z \equiv$ definice τ
$\equiv \text{if } x = \Lambda \text{ then } y*z \text{ else } (\text{začátek}(x).f(\text{zbytek}(x), y))*z \equiv$ přenos $*$ z za if
$\equiv \text{if } x = \Lambda \text{ then } y*z \text{ else začátek}(x).(f(\text{zbytek}(x), y)*z) \equiv$ definice $*$
$\equiv \text{if } x = \Lambda \text{ then } y*z \text{ else začátek}(x).f(\text{zbytek}(x), y*z) \equiv$ indukční předpoklad
$\equiv \tau[f](x, y*z)$ definice τ

■

Dosud uvedené příklady ukázaly užitečnost výpočetní indukce při dokazování řady vlastností programů; problémem však v jistém smyslu zůstává dokazování ukončení, resp. totální korektnosti. Abychom ukázali, že pro jistou pevnou funkci g (různou od Ω) je $g \sqsubseteq f_P$, nelze jednoduše za $\varphi(F)$ zvolit $g \sqsubseteq F$, neboť $\varphi(\Omega)$ je pak vždy *nepravda*.

Je však snadné rozšířit uvedenou techniku na dokazování některých vztahů mezi dvěma danými rekurzivními programy: Nechť jsou dány rekurzivní programy P_1 a P_2:

P_1: $F_1(\bar{x}) \Leftarrow \tau_1[F_1](\bar{x})$

a

P_2: $F_2(\bar{x}) \Leftarrow \tau_2[F_2](\bar{x})$

a nechť $\omega(F_1, F_2)$ je nějaký přípustný predikát, např. konjunkce nerovnosti tvaru $\alpha_i[F_1, F_2] \sqsubseteq \beta_i[F_1, F_2]$, kde α_i, β_i jsou spojité funkcionály. Při důkazu pravdivosti formule $\omega(f_{P_1}, f_{P_2})$ je možné využít tohoto tvrzení:

Jestliže platí $\varphi(\Omega, \Omega)$ *a* $\forall f_1 \forall f_2[\varphi(f_1, f_2) \supset \varphi(\tau_1[f_1], \tau_2[f_2])$, *pak platí i* $\omega(f_{P_1}, f_{P_2})$.

Příklad 5.25

Dokažme, že pro programy P_1, P_2 nad přirozenými čísly:

P_1: $F_1(x, y, z) \Leftarrow \text{if } x = 0 \text{ then } y \text{ else } F_1(x - 1, y + z, z)$,
P_2: $F_2(x, y) \Leftarrow \text{if } x = 0 \text{ then } y \text{ else } F_2(x - 1, y + 2x - 1)$

platí vztah

$$\forall x[f_{P_1}(x, 0, x) \equiv f_{P_2}(x, 0)].$$

(Obě tyto funkce jsou totožné s x^2, to však nebude předmětem důkazu.)

Položme

$$\varphi(F_1, F_2): \quad \forall x \forall y[F_1(y, x(x - y), x) \equiv F_2(y, x^2 - y^2)]$$

a dokazujme pomocí výpočetní indukce $\varphi(f_{P_1}, f_{P_2})$, což se pro $y = x$ zjednodušuje na $\forall x[f_{P_1}(x, 0, x) \equiv f_{P_2}(x, 0)]$.

Krok 1. $\varphi(\Omega, \Omega)$: $\forall x \forall y[\Omega(y, x(x - y), x) \equiv \Omega(y, x^2 - y^2)]$. Důkaz je evidentní, neboť $\omega \equiv \omega$.

Krok 2. $\forall f_1 \forall f_2[\varphi(f_1, f_2) \supset \varphi(\tau_1[f_1], \tau_2[f_2])]$: Z indukčního předpokladu $\forall x \forall y[f_1(y, x(x - y), x) \equiv f_2(y, x^2 - y^2)]$ je třeba dokázat $\forall x \forall y[\tau_1[f_1](y, x(x - y), x) \equiv \tau_2[f_2](y, x^2 - y^2)]$. To však plyne z toho, že:

$\tau_1[f_1](y, x(x - y), x) \equiv$
$\equiv \text{if } y = 0 \text{ then } x(x - y) \text{ else } f_1(y - 1, x(x - y) + x, x) \equiv$
$\equiv \text{if } y = 0 \text{ then } x^2 \text{ else } f_1(y - 1, x(x - (y - 1)), x) \equiv$
$\equiv \text{if } y = 0 \text{ then } x^2 \text{ else } f_2(y - 1, x^2 - (y - 1)^2) \equiv$ indukční předpoklad
$\equiv \text{if } y = 0 \text{ then } x^2 - y^2 \text{ else } f_2(y - 1, (x^2 - y^2) + 2y - 1) \equiv$
$\equiv \tau_2[f_2](y, x^2 - y^2)$.

■

Metodu výpočetní indukce lze zobecnit i na případ rekurzivních programů sestávajících ze systému rekurzivních definic[1])

$F_1(\bar{x}) \Leftarrow \tau_1[F_1, F_2, \ldots, F_m](\bar{x})$,
$F_2(\bar{x}) \Leftarrow \tau_2[F_1, F_2, \ldots, F_m](\bar{x})$,
. . . . . .
$F_m(\bar{x}) \Leftarrow \tau_m[F_1, F_2, \ldots, F_m](\bar{x})$.

Je-li nyní $\varphi(F_1, \ldots, F_n)$ nějaký přípustný predikát, můžeme dokazovat vztah $\varphi(f_{\tau_1}, f_{\tau_2}, \ldots, f_{\tau_m})$ na základě tohoto tvrzení:

Jestliže platí $\varphi(\Omega, \Omega, \ldots, \Omega)$ *a* $\forall f_1 \forall f_2 \ldots \forall f_m[\varphi(f_1, f_2, \ldots, f_m) \supset \supset \varphi(\tau_1[f], \tau_2[f], \ldots, \tau_m[f])]$, *pak platí i* $\varphi(f_{\tau_1}, f_{\tau_2}, \ldots, f_{\tau_m})$.

Příklad 5.26

Je dán rekurzivní program

$F_1(x) \Leftarrow \text{if } p(x) \text{ then } F_1(F_2(h(x))) \text{ else } F_2(g(x))$,
$F_2(x) \Leftarrow \text{if } q(x) \text{ then } k(F_2(F_1(x))) \text{ else } k(h(x))$,
$F_3(x) \Leftarrow \text{if } p(x) \text{ then } F_3(k(F_4(h(x)))) \text{ else } k(F_4(g(x)))$,
$F_4(x) \Leftarrow \text{if } q(x) \text{ then } k(F_4(F_3(x))) \text{ else } h(x)$.

Chceme dokázat, že $\forall x[f_{\tau_1}(x) \equiv f_{\tau_3}(x)]$.

Položme

$$\varphi(F_1, F_2, F_3, F_4): \quad \forall x\{[F_1(x) \equiv F_3(x)] \wedge [F_2(x) \equiv k(F_4(x))]\}$$

a dokazujme $\varphi(f_{\tau_1}, f_{\tau_2}, f_{\tau_3}, f_{\tau_4})$ pomocí výpočetní indukce.

Krok 1. $\varphi(\Omega, \Omega, \Omega, \Omega)$: $\forall x\{[\Omega(x) \equiv \Omega(x)] \wedge [\Omega(x) \equiv k(\Omega(x))]\}$. Důkaz plyne z toho, že $k(\omega) \equiv \omega$.

Krok 2. $\forall f_1 \forall f_2 \forall f_3 \forall f_4[\varphi(f_1, f_2, f_3, f_4) \supset \varphi(\tau_1[f], \tau_2[f], \tau_3[f], \tau_4[f])]$: Z indukčního předpokladu $\forall x\{[f_1(x) \equiv f_3(x)] \wedge [f_2(x) \equiv k(f_4(x))]\}$ je třeba dokázat $\forall x\{[\tau_1[f](x) \equiv \tau_3[f](x)] \wedge [\tau_2[f](x) \equiv k(\tau_4[f](x))]\}$. To však plyne z toho, že:

$\tau_1[f](x) \equiv$
$\equiv \text{if } p(x) \text{ then } f_1(f_2(h(x))) \text{ else } f_2(g(x)) \equiv$
$\equiv \text{if } p(x) \text{ then } f_3(k(f_4(h(x)))) \text{ else } k(f_4(g(x))) \equiv$ indukční předpoklad
$\equiv \tau_3[f](x)$,

[1]) Doporučujeme již nyní si porovnat tyto definice s odst. 4.4.1 ve smyslu dohody zavedené před cv. 5.17. Pozn. překl.

$\tau_2[f](x) \equiv$
$\equiv$ *if* $q(x)$ *then* $k(f_2(f_1(x)))$ *else* $k(h(x)) \equiv$
$\equiv k($*if* $q(x)$ *then* $f_2(f_1(x))$ *else* $h(x)) \equiv$
$\equiv k($*if* $q(x)$ *then* $k(f_4(f_3(x)))$ *else* $h(x)) \equiv$ indukční předpoklad
$\equiv k(\tau_4[f](x))$.

■

5.3.2. Úplná výpočetní indukce[1])

Přeformulujeme nyní pravidlo postupné výpočetní indukce (věta 5.5) do tvaru, který odpovídá běžné úplné matematické indukci nad oborem přirozených čísel. Za tím účelem označme $\tau^i[\Omega]$ symbolem f^i, takže $f^0 = \Omega$ a $f^i = \tau[f^{i-1}]$ pro $i \geqq 1$. Platí:

Věta 5.6 (Úplná výpočetní indukce, Morris). *Je-li P rekurzivní program a $\varphi(F)$ přípustný predikát a platí-li $\forall i\{[\forall j(j < i \supset \varphi(f^j))] \supset \varphi(f^i)\}$, pak platí i $\varphi(f_P)$.*

K důkazu $\varphi(f_P)$ tedy stačí dokázat, že pro libovolné $i \geqq 0$ platí $\varphi(f^i)$ za předpokladu, že platí $\varphi(f^j)$ pro všechna $j < i$. Poněvadž pro $i = 0$ neexistuje v uvažovaném oboru j takové, že $j < i$, je třeba $\varphi(\Omega)$ dokázat „nepodmíněně".

Věta 5.6 je teoreticky silnější než věta 5.5, která požadovala důkaz implikace $\varphi(f) \supset \varphi(\tau[f])$ pro všechny monotónní funkce (nad oborem programu)[2]), zatímco nyní se stačí omezit na všechny funkce tvaru $\tau^i[\Omega]$. Z praktického hlediska mají však obě tvrzení stejnou sílu.

Tvrzení věty 5.6 může být opět zobecněno na případ více programů. Jsou-li např. dány dva rekurzivní programy P_1 a P_2:

P_1: $F_1(\bar{x}) \Leftarrow \tau_1[F_1](\bar{x})$,
P_2: $F_2(\bar{x}) \Leftarrow \tau_2[F_2](\bar{x})$

a přípustný predikát $\varphi(F_1, F_2)$, platí:

Jestliže platí $\forall i\{\forall j[j < i \supset \varphi(f_1^j, f_2^j)] \supset \varphi(f_1^i, f_2^i)\}$, *pak platí* $\varphi(f_{P_1}, f_{P_2})$.

Příklad 5.27 (Morris).
Dokažme, že pro rekurzivní programy P_1, P_2 (viz př. 5.23)

P_1: $F_1(x, y) \Leftarrow$ *if* $p(x)$ *then* y *else* $h(F_1(k(x), y))$,
P_2: $F_2(x, y) \Leftarrow$ *if* $p(x)$ *then* y *else* $F_2(k(x), h(y))$

platí: $\forall x \forall y[f_{P_1}(x, y) \equiv f_{P_2}(x, y)]$.

(A) Důkaz postupnou výpočetní indukcí. Omezíme-li se na výpočetní indukci popsanou ve větě 5.5, musíme dokázat silnější tvrzení než požadujeme (to se ostatně při důkazech indukcí stává často, neboť při důkazu silnějšího tvrzení

[1]) V orig. „complete computational induction" Pozn. překl.
[2]) Viz poznámku za formulací věty 5.5. Pozn. překl.

můžeme využít silnějšího indukčního předpokladu). V našem případě budeme dokazovat

$$\forall x \forall y\{[f_{P_1}(x, y) \equiv f_{P_2}(x, y)] \wedge [f_{P_2}(x, h(y)) \equiv h(f_{P_2}(x, y))]\}.$$

Za tím účelem položme

$$\varphi(F_1, F_2): \quad \forall x \forall y\{[F_1(x, y) \equiv F_2(x, y)] \wedge [F_2(x, h(y)) \equiv h(F_2(x, y))]\}.$$

Krok 1. $\varphi(\Omega, \Omega)$: $\forall x \forall y\{[\Omega(x, y) \equiv \Omega(x, y)] \wedge [\Omega(x, h(y) \equiv h(\Omega(x, y))]\}$.
Důkaz plyne okamžitě z toho, že $h(\omega) \equiv \omega$.

Krok 2. $\forall f_1 \forall f_2[\varphi(f_1, f_2) \supset \varphi(\tau_1[f_1], \tau_2[f_2])$. Z indukčního předpokladu $\forall x \forall y\{[f_1(x, y) \equiv f_2(x, y)] \wedge [f_2(x, h(y)) \equiv h(f_2(x, y))]\}$ je třeba dokázat

$$\forall x \forall y\{[\tau_1[f_1](x, y) \equiv \tau_2[f_2](x, y)] \wedge [\tau_2[f_2](x, h(y)) \equiv h(\tau_2[f_2](x, y))]\}.$$

To však plyne z toho, že

$\tau_1[f_1](x, y) \equiv$
$\equiv$ *if* $p(x)$ *then* y *else* $h(f_1(k(x), y)) \equiv$
$\equiv$ *if* $p(x)$ *then* y *else* $h(f_2(k(x), y)) \equiv$ indukční předpoklad
$\equiv$ *if* $p(x)$ *then* y *else* $f_2(k(x), h(y)) \equiv$ indukční předpoklad
$\equiv \tau_2[f_2](x, y)$,
$\tau_2[f_2](x, h(y)) \equiv$
$\equiv$ *if* $p(x)$ *then* $h(y)$ *else* $f_2(k(x), h(h(y))) \equiv$
$\equiv$ *if* $p(x)$ *then* $h(y)$ *else* $h(f_2(k(x), h(y))) \equiv$ indukční předpoklad
$\equiv h($*if* $p(x)$ *then* y *else* $f_2(k(x), h(y))) \equiv$
$\equiv h(\tau_2[f_2](x, y))$.

(B) Důkaz úplnou výpočetní indukcí. Pomocí této varianty výpočetní indukce lze požadované tvrzení dokázat přímo: tj. dokážeme, že platí

$$\forall x \forall y[f_{P_1}(x, y) \equiv f_{P_2}(x, y)]$$

tím, že postupujeme úplnou výpočetní indukcí vzhledem k predikátu

$$\varphi(F_1, F_2): \quad \forall x \forall y[F_1(x, y) \equiv F_2(x, y)].$$

Důkaz probíhá ve třech krocích; ukážeme, že platí $\varphi(f_1^0, f_2^0)$ a $\varphi(f_1^1, f_2^1)$, a pak obecně $\varphi(f_1^i, f_2^i)$ pro $i \geqq 2$. [Případy $i = 0$ a $i = 1$ vyšetřujeme zvlášť proto, že k důkazu vztahu $\varphi(f_1^i, f_2^i)$ využijeme indukčního předpokladu pro $i - 1$ a $i - 2$.]

Krok 1. $\varphi(f_1^0, f_2^0)$: $\forall x \forall y[f_1^0(x, y) = f_2^0(x, y)]$
Evidentní z definice f^0.

Krok 2. $\varphi(f_1^1, f_2^1)$: $\forall x \forall y[f_1^1(x, y) = f_2^1(x, y)]$
To platí, neboť:

$f_1^1(x, y) \equiv$ *if* $p(x)$ *then* y *else* $h(f_1^0(k(x), y)) \equiv$
$\equiv$ *if* $p(x)$ *then* y *else* $\omega \equiv$ je $h(\omega) \equiv \omega$
$\equiv$ *if* $p(x)$ *then* y *else* $f_2^0(k(x), h(y)) \equiv$
$\equiv f_2^1(x, y)$.

Krok 3. $\forall j[j < i \supset \varphi(f_1^j, f_2^j)] \supset \varphi(f_1^i, f_2^i)$ pro $i \geq 2$:

Z indukčního předpokladu $\forall x \forall y[f_1^{i-2}(x, y) \equiv f_2^{i-2}(x, y)]$

a $\forall x \forall y[f_1^{i-1}(x, y) \equiv f_2^{i-1}(x, y)]$

chceme dokázat $\forall x \forall y[f_1^i(x, y) \equiv f_2^i(x, y)]$

To lze, neboť:

$f_1^i(x, y) \equiv$ *if* $p(x)$ *then* y *else* $h(f_1^{i-1}(k(x), y)) \equiv$ — definice f_1^i

$\equiv$ *if* $p(x)$ *then* y *else* $h(f_2^{i-1}(k(x), y)) \equiv$ — indukční předpoklad pro $i - 1$

$\equiv$ *if* $p(x)$ *then* y *else* $h($*if* $p(k(x))$ *then* y *else* $f_2^{i-2}(k(k(x), h(y))) \equiv$ — definice f_2^{i-1}

$\equiv$ *if* $p(x)$ *then* y *else* $h($*if* $p(k(x))$ *then* y *else* $f_1^{i-2}(k(k(x)), h(y))) \equiv$ — indukční předpoklad pro $i - 2$

$\equiv$ *if* $p(x)$ *then* y *else* (*if* $p(k(x))$ *then* $h(y)$ *else* $h(f_1^{i-2}(k(k(x)), h(y))))) \equiv$ — přenos h za *if*

$\equiv$ *if* $p(x)$ *then* y *else* $f_1^{i-1}(k(x), h(y)) \equiv$ — definice f_1^{i-1}

$\equiv$ *if* $p(x)$ *then* y *else* $f_2^{i-1}(k(x), h(y)) \equiv$ — indukční předpoklad pro $i - 1$

$\equiv f_2^i(x, y)$. — definice f_2^i

∎

Úplnou výpočetní indukci lze dále zobecnit na případ systému rekurzivních definic. Buď $\varphi(F_1, F_2, \ldots, F_m)$ přípustný predikát a položme

$$\langle f_1^0, f_2^0, \ldots, f_m^0 \rangle = \langle \Omega, \Omega, \ldots, \Omega \rangle,$$
$$\langle f_1^{i+1}, f_2^{i+1}, \ldots, f_m^{i+1} \rangle = \langle \tau_1[\bar{f}^i], \tau_2[\bar{f}^i], \ldots, \tau_m[\bar{f}^i] \rangle,$$

kde τ_j jsou funkcionály pravých stran rekurzivních definic. Důkaz toho, že platí $\varphi(f_{\tau_1}, f_{\tau_2}, \ldots, f_{\tau_m})$ lze vést na základě tohoto tvrzení:

Jestliže platí $\forall i\{[\forall j(j < i \supset \varphi(f_1^j, f_2^j, \ldots, f_m^j))] \supset \varphi(f_1^i, f_2^i, \ldots, f_m^i)\}$, *pak platí i* $\varphi(f_{\tau_1}, f_{\tau_2}, \ldots, f_{\tau_m})$.

5.3.3. Další varianty indukce

Dokážeme několik dalších tvrzení užitečných při dokazování vlastností rekurzivních programů.

Věta 5.7 (Transformace zachovávající pevný bod[1]) – Vuillemin). *Nechť jsou dány rekurzivní programy* P: $F(\bar{x}) \Leftarrow \tau[F](\bar{x})$ a P': $F(\bar{x}) \Leftarrow \tau'[F](\bar{x})$, *kde term* $\tau'[F]$ *vznikl z termu* $\tau[F]$ *náhradou některých výskytů symbolu* F *symbolem* f_τ *nebo termem* $\tau[F]$. *Pak platí* $\forall \bar{x}[f_P(\bar{x}) \equiv f_{P'}(\bar{x})]$.

Důkaz. Píšeme $\tau[F]$ ve tvaru $\tau[F, F]$, kde druhý výskyt F v zápisu využijeme k označení těch výskytů symbolu F, které mají být nahrazeny. Položme

[1]) V orig. „fixpoint invariant transformations". Pozn. překl.

$\tau_1[F] = \tau[F, \tau[F, F]]$,
$\tau_2[F] = \tau[F, f_\tau]$.

Chceme dokázat, že $f_\tau \equiv f_{\tau_1} \equiv f_{\tau_2}$; provedeme to ve dvou krocích:

Krok 1. Z definice τ_1 a τ_2 vyplývá, že $f_\tau \equiv \tau_1[f_\tau] \equiv \tau_2[f_\tau]$[1]); je tedy f_τ pevným bodem τ_1 i τ_2, a je tedy více nebo stejně definováno jako f_{τ_1} i jako f_{τ_2}.

Krok 2. $f_\tau \sqsubseteq f_{\tau_1}$ a $f_\tau \sqsubseteq f_{\tau_2}$: Tyto vztahy dokážeme postupnou výpočetní indukcí; položíme

$$\varphi(F, F_1, F_2): \quad (F \sqsubseteq F_1) \wedge (F \sqsubseteq F_2) \wedge (F \sqsubseteq f_\tau) \wedge (F \sqsubseteq \tau[F, F]),$$

Zřejmě platí $\varphi(\Omega, \Omega, \Omega)$. Předpokládejme tedy dále, že platí $\varphi(f, f_1, f_2)$ a dokažme, že platí i $\varphi(\tau[f], \tau_1[f_1], \tau_2[f_2])$. Poněvadž $f \sqsubseteq \tau[f, f]$ a τ je monotónní vzhledem k druhému argumentu, je $\tau[f, f] \sqsubseteq \tau[f, \tau[f, f]]$. Z indukčního předpokladu a monotónnosti τ vyplývá $\tau[f, \tau[f, f]] \sqsubseteq \tau[f_1, \tau[f_1, f_1]]$, takže máme

$$\tau[f] \equiv \tau[f, f] \sqsubseteq \tau[f, \tau[f, f]] \sqsubseteq \tau[f_1, \tau[f_1, f_1]] \equiv \tau_1[f_1],$$

a tedy $\tau[f] \sqsubseteq \tau_1[f_1]$. Podobně z toho, že $f \sqsubseteq f_\tau$ a $f \sqsubseteq f_2$ dostaneme

$$\tau[f] \equiv \tau[f, f] \sqsubseteq \tau[f, f_\tau] \sqsubseteq \tau[f_2, f_\tau] \equiv \tau_2[f_2],$$

a tedy $\tau[f] \sqsubseteq \tau_2[f_2]$. Z $f \sqsubseteq f_\tau$ plyne i

$$\tau[f] \equiv \tau[f, f] \sqsubseteq \tau[f_\tau, f_\tau] \equiv f_\tau$$

a tím i $\tau[f] \sqsubseteq f_\tau$. Konečně z $f \sqsubseteq \tau[f, f]$ vyplývá

$$\tau[f, f] \sqsubseteq \tau[\tau[f, f], \tau[f, f]].$$

∎

Příklad 5.28

Pro rekurzivní programy P_1, P_2 nad oborem přirozených čísel

P_1: $F(x) \Leftarrow$ *if* $x > 10$ *then* $x - 10$ *else* $F(F(x + 13))$,
P_2: $F(x) \Leftarrow$ *if* $x > 10$ *then* $x - 10$ *else* $F(x + 3)$

chceme dokázat, že pro všechna $x \in N$ je

$$f_{P_1}(x) \equiv f_{P_2}(x).$$

Nahradíme-li v P_1 výraz $F(x + 13)$ výrazem $\tau[F](x + 13)$, dostaneme

P_1': $F(x) \Leftarrow$ *if* $x > 10$ *then* $x - 10$ *else* $F($*if* $x + 13 > 10$ *then* $x + 13 - 10$ *else* $F(F(x + 13 + 13)))$.

Podle věty 5.7 je $(\forall x \in N)[f_{P_1}(x) \equiv f_{P_1'}(x)]$. Poněvadž $x \geq 0$, je vždy $x + 13 > 10$, takže $F($*if* $x + 13 > 10$ *then* $x + 13 - 10$ *else* $F(F(x + 13 + 13)))$ lze zjednodušit

[1]) Je $f_\tau \equiv \tau[f_\tau]$ podle definice pevného bodu a $\tau_i[f_\tau]$ $(i = 1, 2)$ je identické s $\tau[f_\tau, f_\tau]$, což je totožné s $\tau[f_\tau]$. Pozn. překl.

na $F(x+3)$. Je tedy $(\forall x \in N)[f_{P_1}(x) \equiv f_{P_2}(x)]$ a odtud plyne i žádaný vztah $(\forall x \in N)[f_{P_1}(x) \equiv f_{P_2}(x)]$.

■

Příklad 5.29
Pro rekurzivní program

$$F_1(x) \Leftarrow \textit{if } p(x) \textit{ then } F_3(F_2(F_2(f(x)))) \textit{ else } g(x),$$
$$F_2(x) \Leftarrow \textit{if } q(x) \textit{ then } F_3(h(x)) \textit{ else } k(x),$$
$$F_3(x) \Leftarrow \textit{if } p(x) \textit{ then } F_1(F_4(f(x))) \textit{ else } g(x),$$
$$F_4(x) \Leftarrow \textit{if } q(x) \textit{ then } F_2(F_3(h(x))) \textit{ else } F_2(k(x))$$

chceme dokázat, že je

$$\forall x[f_{\tau_1}(x) \equiv f_{\tau_3}(x)].$$

Za tím účelem nejprve pozměníme definice F_1 a F_4 takto:

$$F_1(x) \Leftarrow \textit{if } p(x) \textit{ then } F_3(f_{\tau_2}(F_2(f(x)))) \textit{ else } g(x),$$
$$F_4(x) \Leftarrow \textit{if } q(x) \textit{ then } f_{\tau_2}(F_3(h(x))) \textit{ else } f_{\tau_2}(k(x)).$$

Podle věty 5.7 touto úpravou nebyly změněny nejmenší pevné body uvažovaných funkcionálů. Nyní dokážeme metodou postupné výpočetní indukce, že platí

$$\forall x\{[f_{\tau_1}(x) \equiv f_{\tau_3}(x)] \wedge [f_{\tau_2}(f_{\tau_2}(x)) \equiv f_{\tau_4}(x)]\}.$$

Položme proto

$$\varphi(F_1, F_2, F_3, F_4): \quad (F_1 \equiv F_3) \wedge (f_{\tau_2} F_2 \equiv F_4)$$

a dokažme ve dvou krocích:

Krok 1. $\varphi(\Omega, \Omega, \Omega, \Omega)$: $\forall x\{[\Omega(x) \equiv \Omega(x)] \wedge [f_{\tau_2}(\Omega(x)) \equiv \Omega(x)]\}$. To zřejmě platí, protože $f_{\tau_2}(\omega) \equiv \omega$.

Krok 2. $\forall \bar{f}[\varphi(f_1, f_2, f_3, f_4) \supset \varphi(\tau_1'[\bar{f}], \tau_2[\bar{f}], \tau_3[\bar{f}], \tau_4'[\bar{f}])]$;[1][2])

Z indukčního předpokladu $\forall x\{[f_1(x) \equiv f_3(x)] \wedge [f_{\tau_2}(f_2(x)) \equiv f_4(x)]\}$
máme dokázat $\forall x\{[\tau_1'[\bar{f}](x) \equiv \tau_3[\bar{f}](x)] \wedge [f_{\tau_2}(\tau_2[\bar{f}](x)) \equiv \tau_4'[\bar{f}](x)]\}$.

To však plyne z toho, že

$$\begin{aligned}
\tau_1'[\bar{f}](x) &\equiv \textit{if } p(x) \textit{ then } f_3(f_{\tau_2}(f_2(f(x)))) \textit{ else } g(x) \equiv \\
&\equiv \textit{if } p(x) \textit{ then } f_1(f_4(f(x))) \textit{ else } g(x) \equiv \quad \text{indukční předpoklad} \\
&\equiv \tau_3[\bar{f}](x), \\
f_{\tau_2}(\tau_2[\bar{f}](x)) &\equiv f_{\tau_2}(\textit{if } q(x) \textit{ then } f_3(h(x)) \textit{ else } k(x)) \equiv \\
&\equiv \textit{if } q(x) \textit{ then } f_{\tau_2}(f_3(h(x))) \textit{ else } f_{\tau_2}(k(x)) \equiv \\
&\equiv \tau_4'[\bar{f}](x).
\end{aligned}$$

[1]) $\forall \bar{f}$ zde zastupuje $\forall f_1 \forall f_2 \forall f_3 \forall f_4$ a $\tau_i[\bar{f}]$ zastupuje $\tau_i[f_1, f_2, f_3, f_4]$.
[2]) τ_1' a τ_4' jsou funkcionály upravených rekurzívních definic pro F_1 a F_4. Pozn. překl.

Čtenář by si tu měl uvědomit potíže související s přímým důkazem vztahu $\forall x[f_{\tau_1}(x) \equiv f_{\tau_3}(x)]$, který by nepoužil předběžných transformací podle věty 5.7.

■

Další prostředky k důkazu vlastností rekurzívních programů poskytuje tato forma výpočetní indukce[1]):

Věta 5.8 (Park). *Nechť je dán rekurzivní program* P: $F(\bar{x}) \Leftarrow \tau[F](\bar{x})$ *nad oborem* D *a funkce* $g \in [(D_1^-)^n \to D_2^+]$. *Jestliže* $\tau[g] \sqsubseteq g$, *pak* $f_P \sqsubseteq g$.

Důkaz. Použijeme postupné výpočetní indukce na predikát

$$\varphi(F): \quad F \sqsubseteq g.$$

Zřejmě platí $\varphi[\Omega]$, takže zbývá dokázat, že $\forall f[\varphi(f) \supset \varphi(\tau[f])]$. Předpokládejme tedy $f \sqsubseteq g$ a dokazujme $\tau[f] \sqsubseteq g$. Z monotónnosti τ a z indukčního předpokladu $f \sqsubseteq g$ plyne, že $\tau[f] \sqsubseteq \tau[g]$. Podle předpokladu věty je však $\tau[g] \sqsubseteq g$, takže vskutku $\tau[f] \sqsubseteq g$.

■

Příklad 5.30
Pro rekurzivní program P nad oborem přirozených čísel[2])

$$P: \quad F(x) \Leftarrow \textit{if } x > 100 \textit{ then } x - 10 \textit{ else } F(F(x + 11))$$

chceme dokázat, že pro každé celé číslo x je

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

kde

$$g(x): \quad \textit{if } x > 100 \textit{ then } x - 10 \textit{ else } 91.$$

Podle věty 5.8 stačí ukázat, že $\tau[g] \sqsubseteq g$, tj. že

$$\begin{aligned}
&\textit{if } x > 100 \textit{ then } x - 10 \textit{ else} \\
&\qquad [\textit{if } [\textit{if } x + 11 > 100 \textit{ then } x + 11 - 10 \textit{ else } 91] > 100 \\
&\qquad \textit{then } [\textit{if } x + 11 > 100 \textit{ then } x + 11 - 10 \textit{ else } 91] - 10 \textit{ else } 91] \sqsubseteq \\
&\sqsubseteq \textit{if } x > 100 \textit{ then } x - 10 \textit{ else } 91.
\end{aligned}$$

Důkaz tohoto tvrzení je přímý, uvážíme-li možné případy: $x > 100$, $89 < x \leqq 100$ a $x \leqq 89$.

■

Další věta popisuje metodu *rekurzivní indukce*, která představuje první metodu navrženou k dokazování vlastností rekurzivních programů[3]).

[1]) V orig. nazvaná „fixpoint induction", zde název nepřekládáme, poněvadž odpovídající český termín, např. „indukce pevného bodu", se nám nezdá být výstižný. Částečně to ostatně platí i o původním terminu anglickém. Pozn. překl.
[2]) Viz př. 5.14. Pozn. překl.
[3]) I název odráží (pouze) tuto historickou skutečnost, a byl proto ponechán. Pozn. překl.

Věta 5.9 (Rekurzivní indukce, McCarthy). *Nechť jsou dány dvě funkce* $f_1, f_2 \in [(D_1^+)^n \to D_2^+]$ *a podmnožina* $S \subset (D_1^+)^n$. *Aby byla zaručena platnost vztahu* $f_1(\bar{x}) \equiv f_2(\bar{x})$ *pro všechna* $\bar{x} \in S$, *stačí nalézt spojitý funkcionál* τ *takový, že* (1) $f_1 \equiv \tau[f_1]$; (2) $f_2 \equiv \tau[f_2]$; (3) $f_\tau(\bar{x}) \not\equiv \omega$ *pro všechna* $\bar{x} \in S$.

To znamená f_1 a f_2 mají být pevnými body vhodného spojitého funkcionálu τ a $f_\tau(\bar{x})$ má být „definováno" pro všechna $\bar{x} \in S$.

Důkaz. Poněvadž f_τ je nejmenší pevný bod funkcionálu τ a f_1, f_2 jsou pevné body téhož funkcionálu, je $f_\tau(\bar{x}) \sqsubseteq f_1(\bar{x})$ a $f_\tau(\bar{x}) \sqsubseteq f_2(\bar{x})$ pro všechna $\bar{x}$ z $(D_1^+)^n$. Poněvadž navíc pro všechna $\bar{x} \in S$ je $f_\tau(\omega) \not\equiv \omega$, je pro všechna $\bar{x} \in S$ i

$$f_\tau(\bar{x}) \equiv f_1(\bar{x}) \equiv f_2(\bar{x}).$$

∎

Pečlivý čtenář si jistě povšiml, že o funkcionálu τ stačilo místo (1) a (2) předpokládat platnost slabších podmínek: (1′) $f_1 \sqsupseteq \tau[f_1]$ (2′) $f_2 \sqsupseteq \tau[f_2]$. Hlavní problém metody rekurzivní indukce je však v předpokladu (3), který musí být zachován. K důkazu ekvivalence dvou funkcí je zapotřebí navíc dokázat ukončení. Metodu rekurzivní indukce tedy nelze využít k důkazu ekvivalence rekurzivních programů, u nichž obory, funkce a predikáty nejsou specifikovány[1]).

Příklad 5.31

Funkci *inverze*(x)[2]) nad Σ^* lze definovat jako $f_P(x, \Lambda)$, kde

$$P: \quad F(x, y) \Leftarrow \textit{if } x = \Lambda \textit{ then } y \textit{ else } F(\textit{zbytek}(x), \textit{začátek}(x) \,.\, y).$$

Pomocí metody rekurzivní indukce chceme dokázat, že

$$\textit{inverze}(x * y) \equiv \textit{inverze}(y) * \textit{inverze}(x) \text{ pro všechna } x, y \in \Sigma^*:$$

zde $*$ je *funkce zřetězení*, tj. nejmenší pevný bod rekurzivního programu

$$G(x, y) \Leftarrow \textit{if } x = \Lambda \textit{ then } y \textit{ else začátek}(x) \,.\, G(\textit{zbytek}(x), y).$$

Zvolme funkcionál τ definovaný takto:

$$\tau[F](x, y): \quad \textit{if } x = \Lambda \textit{ then inverze}(y) \textit{ else } F(\textit{zbytek}(x), y) * \textit{začátek}(x).$$

Využijeme-li známé vlastnosti funkcí $*$ a *inverze*, že totiž $\textit{inverze}(\sigma \,.\, w) \equiv \textit{inverze}(w) * \sigma$ pro všechna $\sigma \in \Sigma$ a $w \in \Sigma^*$, dostaneme:

Krok 1. $\textit{inverze}(x * y)$ je pevný bod funkcionálu τ, protože

$$\begin{aligned}\textit{inverze}(x * y) &\equiv \textit{inverze}(\textit{if } x = \Lambda \textit{ then } y \textit{ else začátek}(x) \,.\, (\textit{zbytek}(x) * y)) \equiv \\ &\equiv \textit{if } x = \Lambda \textit{ then inverze}(y) \textit{ else inverze}(\textit{začátek}(x) \,.\, (\textit{zbytek}(x) * y)) \equiv \\ &\equiv \textit{if } x = \Lambda \textit{ then inverze}(y) \textit{ else inverze}(\textit{zbytek}(x) * y) * \textit{začátek}(x).\end{aligned}$$

[1]) Tedy vlastně rekurzivních schémat jako P_1 a P_2 z př. 5.27. Pozn. překl.

[2]) Viz př. 1.19 a př. 5.24. Pozn. překl.

Krok 2. $\textit{inverze}(y) * \textit{inverze}(x)$ je pevný bod funkcionálu τ, protože

$$\begin{aligned}&\textit{inverze}(y) * \textit{inverze}(x) \equiv \\ &\equiv \textit{if } x = \Lambda \textit{ then inverze}(y) * \Lambda \textit{ else inverze}(y) * \textit{inverze}(x) \equiv \\ &\equiv \textit{if } x = \Lambda \textit{ then inverze}(y) \textit{ else inverze}(y) * (\textit{inverze}(\textit{zbytek}(x)) * \textit{začátek}(x)) \equiv \\ &\equiv \textit{if } x = \Lambda \textit{ then inverze}(y) \textit{ else } (\textit{inverze}(y) * \textit{inverze}(\textit{zbytek}(x))) * \textit{začátek}(x).\end{aligned}$$

(Poslední rovnost se opírá o asociativnost funkce $*$.)

Krok 3. $f_\tau(x, y) \not\equiv \omega$ pro všechna $x, y \in \Sigma^*$, jak lze snadno ukázat indukcí podle x.

∎

5.3.4. Strukturní indukce

Princip úplné matematické indukce pro dokazování pravdivosti tvrzení nad oborem přirozených čísel lze formulovat takto: Abychom dokázali, že $\psi(n)$ je *pravda* pro všechna n, stačí, když dokážeme, že pro libovolné i pravdivost $\psi(i)$ vyplývá z předpokladu pravdivosti $\psi(j)$ pro všechna $j < i$. Formálněji zapsáno: *Jestliže platí* $(\forall i \in N)\{[(\forall j \in N)(j < i \supset \psi(j))] \supset \psi(i)\}$, *pak platí i* $(\forall n \in N)\psi(n)$. Toto pravidlo nemusí platit v každém uspořádaném oboru: platí pro množinu N všech přirozených čísel (přirozeně uspořádanou relací $<$), neplatí však, když místo N vezmeme množinu všech celých čísel (opět s přirozeným uspořádáním): uvažujte např. ψ, které je vždy *nepravda*.

Podáme nyní charakteristiku uspořádaných oborů, pro něž pravidlo indukce platí a vyslovíme ho v obecném tvaru tzv. *strukturní indukce*, která zahrnuje postupnou i úplnou indukci nad přirozenými čísly jako zvláštní případy. Účinnost metody strukturní indukce ukážeme pak na příkladech důkazů vlastností rekurzivních programů.

Metodu strukturní indukce je možné aplikovat na každý obor, který je dobře fundovanou množinou $(S, \prec)$ ve smyslu odst. 3.1.2, tj. množinou s ostrým částečným uspořádáním, která neobsahuje žádnou nekonečnou klesající posloupnost $a_0 \succ a_1 \succ a_2 \succ \ldots$ prvků S (viz odst. 3.1.2).

Věta 5.10 (Strukturní indukce, Burstall). *Nechť* $(S, \prec)$ *je dobře fundovaná množina a* φ *totální predikát nad* S:

$$\textit{jestliže } (\forall a \in S)\{[(\forall b \in S)(b \prec a \supset \varphi(b))] \supset \varphi(a)\}, \textit{ pak } (\forall c \in S)\varphi(c).$$

Můžeme-li tedy dokázat pro každé a z S pravdivost $\varphi(a)$ z pravdivosti $\varphi(b)$ pro všechna $b \prec a$, je $\varphi(c)$ *pravda* pro všechna $c \in S$[1]).

Důkaz. Ukážeme, že za předpokladů věty neexistuje prvek z S takový, že φ je pro něj *nepravda*. Uvažujme množinu

$$A = \{a \mid a \in S \wedge \varphi(a) \text{ je } \textit{nepravda}\}$$

[1]) Jde tedy o verzi známé *noetherovské indukce*; viz např. [Cohn 1965].

Předpokládejme, že A není prázdná a označme a_0 nějaký prvek z A takový, že pro všechna $a \in A$ platí $a \not\prec a_0$; takový prvek musí existovat, neboť v opačném případě by bylo možné sestrojit z prvků A nekonečnou klesající posloupnost. Pro každý prvek b z S takový, že $b \prec a$, je $\varphi(b)$ *pravda*[1]), tj. platí: $(\forall b \in S)(b \prec a_0 \supset \varphi(b))$. Podle předpokladu věty však v tomto případě musí platit i $\varphi(a_0)$, což odporuje předpokladu $a_0 \in A$. Množina A je proto prázdná a $\varphi(c)$ je *pravda* pro všechny prvky $c \in S$.

∎

Poznamenejme, že neexistuje-li v S žádné b takové, že $b \prec a$, pak výrok $(\forall b \in S)(b \prec a \supset \varphi(b))$ je triviálně pravdivý. Pro prvky a s touto vlastností je tedy třeba dokázat $\varphi(a)$ přímo.

Než přejdeme k příkladům užití strukturní indukce, připomeňme (viz odst. 3.2.2), že je-li $(S, \prec)$ dobře fundovaná množina, je takovou i množina $(S^n, \prec_n)$, kde S^n je množina uspořádaných n-tic prvků množiny S a $\prec_n$ je lexikografické částečné uspořádání definované vztahem $\langle a_1, \ldots, a_n \rangle \prec_n \langle b_1, \ldots, b_n \rangle$, právě když

$$a_1 = b_1, \ldots, a_{i-1} = b_{i-1} \quad \text{a} \quad a_i \prec b_i \quad \text{pro nějaké } i, \quad 1 \leqq i \leqq n.$$

Ve všech uváděných příkladech se využívá pouze toho, že f_P je pevný bod, tj. že $\tau[f_P] \equiv f_P$. Uvedené výsledky platí tedy pro všechny pevné body funkcionálu τ a nikoli jen pro nejmenší pevný bod.

Příklad 5.32

Pro rekurzivní program P nad oborem přirozených čísel

$$P: \quad F(x, y) \Leftarrow \textit{if } x = 0 \textit{ then } y + 1 \textit{ else if } y = 0 \textit{ then } F(x - 1, 1) \textit{ else } F(x - 1, F(x, y - 1))$$

je $f_P(x, y)$ tzv. *Ackermannova funkce* (viz př. 3.9 a cv. 3.15). Chceme ukázat, že $f_P(x, y)$ je totální, tj. že $f_P(x, y) \not\equiv \omega$ pro všechna $x, y \in N$. K důkazu použijeme metodu strukturní indukce nad oborem $(N^2, <_2)$, kde $<_2$ je lexikografické uspořádání dvojic přirozených čísel.

Krok 1. Je-li $x = 0$, je $f_P(x, y)$ zřejmě definováno.

Krok 2. Je-li $x \neq 0$ a $y = 0$, pak z toho, že $\langle x - 1, 1 \rangle <_2 \langle x, y \rangle$ a z indukčního předpokladu plyne, že $f_P(x - 1, 1)$ je definováno; odtud plyne, že i $f_P(x, y)$ je definováno.

Krok 3. Je-li $x \neq 0$ a $y \neq 0$, pak $\langle x, y - 1 \rangle <_2 \langle x, y \rangle$, a tedy $f_P(x, y - 1)$ je podle indukčního předpokladu definováno. Bez ohledu na hodnotu $f_P(x, y - 1)$ je však $\langle x - 1, f_P(x, y - 1) \rangle <_2 \langle x, y \rangle$ a dokazované tvrzení vyplývá opět z indukčního předpokladu.

∎

[1]) Jinak by b patřilo do A. Pozn. překl.

Další příklad využívá strukturní indukci nad oborem přirozených čísel N s přirozeným uspořádáním, tj. běžnou *úplnou matematickou indukci*.

Příklad 5.33 (Cadiou)

Pro rekurzivní programy P_1, P_2 nad oborem přirozených čísel

$$P_1: \quad F_1(x) \Leftarrow \textit{if } x = 0 \textit{ then } 1 \textit{ else } x \,.\, F_1(x - 1),$$
$$P_2: \quad F_2(x, y) \Leftarrow \textit{if } x = y \textit{ then } 1 \textit{ else } F_2(x, y + 1) \,.\, (y + 1)$$

jsou funkce $f_{P_1}(x)$ a $f_{P_2}(x, 0)$ totožné s funkcí faktoriál $x!$. Výpočty však probíhají rozdílně: u $f_{P_1}(x)$ jde výpočet „zhora dolů" od x k 0, u $f_{P_2}(x, 0)$ jde „zdola nahoru", od 0 k x. Chceme ukázat, že

$$f_{P_2}(x, 0) \equiv f_{P_1}(x) \quad \text{pro všechna} \quad x \in N.$$

Položme

$$\varphi(x): \quad (\forall y \in N)[f_{P_2}(x + y, y) \,.\, f_{P_1}(y) \equiv f_{P_1}(x + y)]$$

a dokazujme platnost formule $\varphi(x)$ pro každé $x \in N$ (na N uvažujeme přirozené uspořádání):

Krok 1. $\varphi(0)$ je $(\forall y \in N)[f_{P_2}(y, y) \,.\, f_{P_1}(y) \equiv f_{P_1}(y)]$, což platí podle definice f_{P_2}.

Krok 2. Předpokládejme, že $x > 0$ a že $\varphi(x')$ platí pro všechna $x' < x$; dokažme, že platí i $\varphi(x)$. Pro všechna $y \in N$ je:

$f_{P_2}(x + y, y) \,.\, f_{P_1}(y) \equiv$
$\equiv f_{P_2}(x + y, y + 1) \,.\, (y + 1) \,.\, f_{P_1}(y) \equiv$ podle definice f_{P_2} (neboť $x + y > y$)
$\equiv f_{P_2}(x + y, y + 1) \,.\, f_{P_1}(y + 1) \equiv$ podle definice f_{P_1} (neboť $y + 1 > 0$)
$\equiv f_{P_2}((x - 1) + (y + 1), y + 1) \,.\, f_{P_1}(y + 1) \equiv$ neboť $x > 0$
$\equiv f_{P_1}((x - 1) + (y + 1)) \equiv$ indukční předpoklad (neboť $x - 1 < x$)
$\equiv f_{P_1}(x + y).$

Věta 5.10 zaručuje platnost $\varphi(x)$ pro všechna $x \in N$. Speciálně pro $y = 0$ je tedy $f_{P_2}(x + 0, 0) \,.\, f_{P_1}(0) \equiv f_{P_1}(x + 0)$. Poněvadž $f_{P_1}(0) \equiv 1$, je $f_{P_2}(x, 0) \equiv f_{P_1}(x)$ pro všechna $x \in N$, což jsme měli dokázat.

∎

V příkladech 5.32 a 5.33 jsme opřeli důkazy indukcí o nejpřirozenější uspořádání uvažovaného oboru; v př. 5.34 se ukazuje vhodné použít netradiční uspořádání.

Příklad 5.34 (Burstall)

„Funkce 91"[1]) je definována jako nejmenší pevný bod f_P rekurzivního programu P nad oborem celých čísel I:

$$P: \quad F(x) \Leftarrow \textit{if } x > 100 \textit{ then } x - 10 \textit{ else } F(F(x + 11)).$$

[1]) Viz př. 5.8 a př. 5.14. Pozn. překl.

Chceme ukázat, že

$f_P(x) \equiv g(x)$ pro všechna celá čísla x,

kde

$g(x)$ je *if* $x > 100$ *then* $x - 10$ *else* 91.

Důkaz metodou strukturní indukce využije dobře fundovanou množinu $(I, \prec)$, kde relace $\prec$ je definována takto:

$x \prec y$, právě když $y < x \leqq 101$

(zde $<$ označuje přirozené uspořádání množiny celých čísel); je tedy $101 \prec 100 \prec 99 \prec \ldots$, ale $102 \not\prec 101$ ap. Snadno lze ověřit, že $(I, \prec)$ je skutečně dobře fundovaná množina.

Předpokládejme nyní, že $f_P(y) \equiv g(y)$ platí pro všechna $y \in I$ taková, že $y \prec x$ a dokazujme, že platí $f_P(x) \equiv g(x)$.

Krok 1. Je-li $x > 100$, vyplývá vztah $f_P(x) \equiv x - 10 \equiv g(x)$ ihned z definice obou funkcí.

Krok 2. Je-li $100 \geqq x \geqq 90$, plyne z definice f_P přímo, že $f_P(x) \equiv f_P(f_P(x + 11)) \equiv f_P(x + 1)$; poněvadž $x + 1 \prec x$, lze z indukčního předpokladu odvodit $f_P(x) \equiv f_P(x + 1) \equiv g(x + 1)$. Jelikož však $x + 1 \leqq 101$, z definice g vyplývá $g(x + 1) \equiv 91 \equiv g(x)$; tedy $f_P(x) \equiv g(x)$.

Krok 3. Je-li $x < 90$, je $f_P(x) \equiv f_P(f_P(x + 11))$ podle definice f_P a poněvadž $x + 11 \prec x$, podle indukčního předpokladu máme $f_P(x) \equiv f_P(f_P(x + 11)) \equiv f_P(g(x + 11))$. Poněvadž $x + 11 \leqq 100$, je $g(x + 11) \equiv 91$ podle definice g, a tedy platí $f_P(x) \equiv f_P(91)$. Podle indukčního předpokladu je $f_P(91) \equiv g(91)$ a z definice g víme, že $g(91) \equiv 91 \equiv g(x)$. Tedy máme $f_P(x) \equiv f_P(91) \equiv g(91) \equiv 91 \equiv g(x)$, což jsme měli dokázat.

Tvrzení jsme mohli dokázat i použitím strukturní indukce nad oborem přirozených čísel s běžným uspořádáním $<$, kdybychom uvažovali složitější predikát

$\varphi(n)$: $(\forall x \in I)[x > 100 - n \supset f_P(x) \equiv g(x)]$.

Detaily takového důkazu jsou shodné s úvahami předloženého důkazu.

∎

Významným příkladem dobře fundované množiny je i množina $(\Sigma^*, \prec)$, kde Σ je daná konečná abeceda a $x \prec y$, právě když slovo x je podslovem slova y[1]). V následujícím příkladě použijeme metodu strukturní indukce nad touto množinou, a to v této modifikaci: Platí-li $\varphi(\Lambda)$ a $(\forall x \in \Sigma^*)[x \neq \Lambda \wedge \varphi(zbytek(x)) \supset \varphi(x)]$, platí i $(\forall x \in \Sigma^*)\varphi(x)$.

Příklad 5.35 (Ness)

Uvažujme opět[2]) funkci *inverze*(x), která je nejmenším pevným bodem $f_P(x, \Lambda)$ rekurzivního programu

P: $F(x, y) \Leftarrow$ *if* $x = \Lambda$ *then* y *else* $F(zbytek(x), začátek(x) \,.\, y)$.

[1]) To je $y = uxv$ pro vhodná slova $u, v \in \Sigma^*$, z nichž alespoň jedno je neprázdné. Pozn. překl.

[2]) Viz př. 5.31. Pozn. překl.

Chceme dokázat, že *inverze*(x) je totální funkce, tj. že *inverze*$(x) \not\equiv \omega$ pro všechna $x \in \Sigma^*$ a že pro všechna $x \in \Sigma^*$ platí také *inverze*(*inverze*$(x)) \equiv x$. Důkazem těchto vlastností ještě ovšem nedokážeme, že jde nutně o skutečnou inverzi slov: existuje řada dalších funkcí, která uvedené požadavky splňují – např. identická funkce.

Krok 1. Uvažujme predikát

$\varphi(x)$: $(\forall y \in \Sigma^*)[f_P(x, y)$ je definováno$]$

a ukažme pomocí něho, že *inverze*(x) je všude definováno.

(a) Je-li $x = \Lambda$, je $f_P(x, y) \equiv y$ a $f_P(x, y)$ je tedy definováno pro všechna $y \in \Sigma^*$.

(b) Je-li $x \neq \Lambda$, můžeme předpokládat [protože $zbytek(x) \prec x$], že $f_P(zbytek(x), z)$ je definováno pro všechna $z \in \Sigma^*$, takže $f_P(zbytek(x), začátek(x) \,.\, y)$ je definováno pro všechna $y \in \Sigma^*$. Je tedy $f_P(x, y)$ definováno pro všechna $y \in \Sigma^*$.

Metoda strukturní indukce tak zaručuje, že $f_P(x, y)$ je definováno pro všechna $x, y \in \Sigma^*$; tím i *inverze*(x) je definováno pro všechna $x \in \Sigma^*$, neboť *inverze*$(x) \equiv f_P(x, \Lambda)$.

Krok 2. Abychom dokázali, že platí $(\forall x \in \Sigma^*)[inverze(inverze(x)) \equiv x]$, položme

$\varphi(x)$: $(\forall y \in \Sigma^*)[inverze(f_P(x, y)) \equiv f_P(y, x)]$.

(a) Je-li $x = \Lambda$, je pro každé $y \in \Sigma^*$

$inverze(f_P(\Lambda, y)) \equiv inverze(y) \equiv f_P(y, \Lambda)$.

(b) Je-li $x \neq \Lambda$, je pro každé $y \in \Sigma^*$

$inverze(f_P(x, y)) \equiv$

$\equiv inverze(f_P(zbytek(x), začátek(x) \,.\, y)) \equiv$ definice f_P (neboť $x \neq \Lambda$)

$\equiv f_P(začátek(x) \,.\, y, zbytek(x)) \equiv$ indukční předpoklad [neboť $zbytek(x) \prec x$]

$\equiv f_P(y, začátek(x) \,.\, zbytek(x)) \equiv$ definice f_P [neboť $začátek(x) \,.\, y \neq \Lambda$]

$\equiv f_P(y, x)$.

Je tedy $inverze(f_P(x, y)) \equiv f_P(y, x)$ pro všechna $x, y \in \Sigma^*$ a tím speciálně i $inverze(f_P(x, \Lambda)) \equiv f_P(\Lambda, x)$, takže pro všechna $x \in \Sigma^*$

$inverze(inverze(x)) \equiv inverze(f_P(x, \Lambda)) \equiv f_P(\Lambda, x) \equiv x$,

což jsme měli dokázat.

∎

Analogicky lze dokázat i další vlastnosti funkce *inverze*. V př. 5.36 využijeme toho, že pro všechna $a, b \in \Sigma$ a $w \in \Sigma^*$ je:

1. $inverze(w * a) \equiv a \,.\, inverze(w)$,
2. $inverze(a \,.\, w) \equiv inverze(w) * a$,
3. $inverze(a \,.\, (w * b)) \equiv b \,.\, (inverze(w) * a)$,

kde $*$ je funkce zřetězení (viz př. 5.24).

Příklad 5.36 (Ashcroft)
Rekurzivní program nad Σ^*

P: $F(x) \Leftarrow$ *if* $x = \Lambda$ *then* Λ
else if $zbytek(x) = \Lambda$ *then* x
else $začátek(F(zbytek(x))) \,.\, F(začátek(x) \,.\, F(zbytek(F(zbytek(x)))))$

definuje funkci *inverze*, tj.

$$f_P(x) \equiv inverze(x) \quad \text{pro všechna} \quad x \in \Sigma^*$$

(všimněme si, že tato definice nepoužívá žádné další pomocné funkce).

Uvedená skutečnost je celkem snadno pochopitelná: výraz

α: $inverze(zbytek(inverze(zbytek(x))))$

reprezentuje slovo x bez prvního a posledního písmene, takže

β: $začátek(x) \,.\, \alpha$

je x bez posledního písmene; poslední písmeno je

γ: $začátek(inverze(zbytek(x)))$.

Je-li tedy $x \neq \Lambda$ a $zbytek(x) \neq \Lambda$, je $inverze(x) \equiv \gamma \,.\, inverze(\beta)$.

Formální důkaz využije této charakteristiky prvků z Σ^*: *Pro libovolné* $x \in \Sigma^*$ *je buď* $x = \Lambda$, *nebo* $x \in \Sigma$ [*tj.* $x \neq \Lambda$ *a* $zbytek(x) = \Lambda$], *nebo* $x = a \,.\, (w * b)$ *pro* $a, b \in \Sigma$ *a* $w \in \Sigma^*$. (Toto lemma se snadno dokáže strukturní indukcí.) Metodou strukturní indukce nyní dokážeme

$$(\forall x \in \Sigma^*)\,[f_P(x) \equiv inverze(x)].$$

Indukci vedeme nad oborem $(\Sigma^*, \prec)$, kde $\prec$ je (ostré) částečné uspořádání definované vztahem

$$x \prec y, \quad \text{právě když} \quad |x| < |y|\,[1]).$$

Je zřejmé, že $(\Sigma^*, \prec)$ je dobře fundovaná množina.

Na základě zmíněného lemmatu rozdělíme důkaz do tří částí:

Krok 1. Je-li $x = \Lambda$, je $f_P(x) \equiv \Lambda \equiv inverze(x)$.

Krok 2. Je-li $x \in \Sigma$, je $f_P(x) \equiv x \equiv inverze(x)$.

Krok 3. Je-li $x = a \,.\, (w * b)$ pro jistá $a, b \in \Sigma$, $w \in \Sigma^*$, je

$f_P(x) \equiv$
$\equiv začátek(f_P(zbytek(x))) \,.\, f_P(začátek(x) \,.\, f_P(zbytek(f_P(zbytek(x))))) \equiv$
definice f_P
$\equiv začátek(f_P(w * b)) \,.\, f_P(a \,.\, f_P(zbytek(f_P(w * b)))) \equiv$
$začátek(x)$ je a, $zbytek(x)$ je $w * b$
$\equiv začátek(inverze(w * b)) \,.\, f_P(a \,.\, f_P(zbytek(inverze(w * b)))) \equiv$
indukční předpoklad (neboť $w * b \prec x$)
$\equiv začátek(b \,.\, inverze(w)) \,.\, f_P(a \,.\, f_P(zbytek(b \,.\, inverze(w)))) \equiv$
vlastnost 1[1])
$\equiv b \,.\, f_P(a \,.\, f_P(inverze(w))) \equiv$ vlastnosti začátku a zbytku
$\equiv b \,.\, f_P(a \,.\, inverze(inverze(w))) \equiv$ indukční předpoklad [neboť $inverze(w) \prec x$]
$\equiv b \,.\, f_P(a \,.\, w) \equiv$ vlastnosti *inverze* dokázané v př. 5.35
$\equiv b \,.\, inverze(a \,.\, w) \equiv$ indukční předpoklad (neboť $a \,.\, w \prec x$)
$\equiv b \,.\, (inverze(w) * a) \equiv$ vlastnost 2
$\equiv inverze(x).$ vlastnost 3

Je tedy $f_P(x) \equiv inverze(x)$ pro všechna $x \in \Sigma^*$, jak jsme měli ukázat. ∎

[1]) Připomeňme, že pro každé slovo $w \in \Sigma^*$ označuje $|w|$ délku slova w, tj. počet výskytů písmen ve w; speciálně je $|\Lambda| = 0$.

CVIČENÍ

Cv. 5.1 (Monotónní funkce). Ukažte, že tyto funkce jsou monotónní:

(a) „Sekvenční" *if-then-else* (viz př. 5.3).

(b) „Paralelní" *if-then-else*, definované stejně jako „sekvenční" s tou výjimkou, že (*if* ω *then* d *else* d) $\equiv d$ pro všechna $d \in D^+$.

(c) „Paralelní" násobení celých čísel, kde

$(0 * \omega) \equiv (\omega * 0) \equiv 0$,
$(a * \omega) \equiv (\omega * a) \equiv \omega$ pro všechna $a \neq 0$,
$(\omega * \omega) \equiv \omega$.

(d) „Symetrické" varianty predikátů *nebo* a *a*:

(*pravda nebo* ω) $\equiv$ (ω *nebo pravda*) $\equiv$ *pravda*.
(*nepravda nebo* ω) $\equiv$ (ω *nebo nepravda*) $\equiv \omega$,
(ω *nebo* ω) $\equiv \omega$,
(*pravda a* ω) $\equiv$ (ω *a pravda*) $\equiv \omega$,
(*nepravda a* ω) $\equiv$ (ω *a nepravda*) $\equiv$ *nepravda*.
(ω *a* ω) $\equiv \omega$.

Cv. 5.2 (Funkcionály).

(a) Které z uvedených funkcionálů nad $[I^+ \to I^+]$ jsou monotónní a které spojité?

(i) *if* (F je totální) *then* ω *else* 0,
(ii) *if* (F je totální) *then* 0 *else* ω,
(iii) *if* (F je totální) *then* 0 *else* 0

[zde „*F je totální*" znamená „$F(x) \not\equiv \omega$ pro všechna $x \in I^+$"].

(b) Uvažujte funkcionál $\tau[F](x)$: $F(F(x))$ a funkce

$f(x)$: *if* $x \equiv \omega$ *then* 0 *else* ω,
$g(x)$: *if* $x \equiv \omega$ *then* 0 *else* 1.

[1]) Viz text před příkladem. Podobně dále. Pozn. překl.

A.1.2. Obecný postup

Zobecníme nyní postup použitý v předchozím příkladě na případ libovolného programu. Nechť je dán program P, vstupní predikát φ a výstupní predikát ψ. Při důkazu parciální korektnosti programu P vzhledem k φ a ψ metodou podcílových podmínek postupujme takto:

Krok 1. V prvním kroku rozdělíme všechny cykly programu užitím konečné množiny dělicích bodů. K této množině přidáme (výchozí) bod S umístěný za symbol START a (koncové) body Z umístěné před každý symbol STOP. Pro každou cestu α mezi body i a j, která neobsahuje další body, zkonstruujeme predikát $R_\alpha(\bar{x}, \bar{y})$ [je-li $i = S$, pak jde o $R_\alpha(\bar{x})$] a funkci $r_\alpha(\bar{x}, \bar{y})$ [je-li $i = S$, pak jde o $r_\alpha(\bar{x})$] (viz odst. 3.1.1).

Krok 2. V dalším kroku přiřadíme každému dělicímu bodu i predikát $Q_i(\bar{x}, \bar{y}, \bar{z})$ [pro $i = S$ predikát $Q_S(\bar{x}, \bar{z})$], nazývaný *podcílová podmínka*, který charakterizuje parciální korektnost podprogramu i. Predikát Q_i má tedy mít tu vlastnost, že kdykoliv běh programu prochází bodem i s průběžnými hodnotami proměnných $\bar{x}$, resp. $\bar{y}$ rovnými $\bar{\xi}$ resp. $\bar{\eta}$ a další výpočet se ukončí s hodnotou proměnné $\bar{z}$ rovnou $\bar{\zeta}$, je $Q_i(\bar{\xi}, \bar{\eta}, \bar{\zeta})$ pravdivé.

Krok 3. Ve třetím kroku sestrojíme tyto verifikační podmínky:

(a) $\forall \bar{x} \forall \bar{z} Q_Z(\bar{x}, \bar{z}, \bar{z})$;

(b) $\forall \bar{x} \forall \bar{z}[\varphi(\bar{x}) \wedge Q_S(\bar{x}, \bar{z}) \supset \psi(\bar{x}, \bar{z})]$;

(c) pro každou cestu α z dělicího bodu i do dělicího bodu j, která neobsahuje další dělicí body

$$\forall \bar{x} \forall \bar{y} \forall \bar{z}[R_\alpha(\bar{x}, \bar{y}) \wedge Q_j(\bar{x}, r_\alpha(\bar{x}, \bar{y}), \bar{z}) \supset Q_i(\bar{x}, \bar{y}, \bar{z})].$$

Speciálně pro $i = S$ má podmínka tvar

$$\forall \bar{x} \forall \bar{z}[R_\alpha(\bar{x}) \wedge Q_j(\bar{x}, r_\alpha(\bar{x}), \bar{z}) \supset Q_S(\bar{x}, \bar{z})].$$

V závěrečném kroku dokážeme, že všechny verifikační podmínky jsou pravdivé. Podaří-li se nám to, pak máme zaručeno, že všechny podcílové podmínky jsou skutečně pravdivé v odpovídajících bodech. Speciálně je tedy Q_S pravdivá v S a z platnosti verifikační podmínky (b) plyne, že program P je parciálně korektní vzhledem k φ a ψ.

Celý postup důkazu parciální korektnosti můžeme formulovat ve tvaru věty.

Věta A.1 (Metoda podcílových podmínek — Morris, Wegbreit). *Je-li dán program P, vstupní predikát $\varphi(\bar{x})$ a výstupní predikát $\psi(\bar{x}, \bar{z})$, sestrojte postupně*

(1) *množinu dělicích bodů,*

(2) *vhodné podcílové podmínky,*

(3) *verifikační podmínky.*

Jsou-li všechny verifikační podmínky pravdivé, je program P parciálně korektní vzhledem k predikátům φ a ψ.

Kroky tohoto postupu, s výjimkou nalezení vhodných podcílových podmínek, lze snadno zmechanizovat. K nalezení podcílové podmínky je opět nutná hluboká znalost funkce a smyslu celého programu. Na rozdíl od metody induktivních podmínek mají však podcílové podmínky jiný charakter. Při jejich tvorbě se nesoustřeďujeme na invariantní, tj. neměnné vlastnosti programu, ale hlavní důraz je kladen na to, co je programem měněno, tj. na dynamické vlastnosti programu.

Poznamenejme, že platí obrácené tvrzení k větě A.1, tj. je-li program P parciálně korektní vzhledem k φ a ψ, pak existují dělicí body a podcílové podmínky tak, že odpovídající verifikační podmínky jsou pravdivé.

A.1.3. Příklady

Ukažme na závěr tohoto článku použití metody podcílových podmínek na dvou příkladech.

Příklad A.1

Dokažme parciální korektnost programu P_2 z obr. 3.6 vzhledem ke vstupnímu predikátu

$$\varphi(x_1, x_2): \quad x_2 \geqq 0$$

a výstupnímu predikátu

$$\psi(x_1, x_2, z): \quad z = x_1^{x_2}.$$

Použijeme dělicí body stejně jako u př. 3.2. Dělicím bodům A, B, C přiřadíme tyto podcílové podmínky:

$$Q_A: \quad x_2 \geqq 0 \supset z = x_1^{x_2},$$
$$Q_B: \quad y_2 \geqq 0 \supset z = y_3 \cdot y_1^{y_2},$$
$$Q_C: \quad y_3 = z.$$

Připomeňme, že užitím dělicích bodů jsme získali celkem čtyři cesty, kterým odpovídají tyto predikáty R a funkce r:

cesta α (z A do B):

$$R_\alpha: \quad T, \qquad r_\alpha: \quad (x_1, x_2, 1);$$

cesta β_1 (z B do B přes příkaz 1):

$$R_{\beta_1}: \quad y_2 \neq 0 \wedge \textit{liché}(y_2), \qquad r_{\beta_1}: \quad (y_1, (y_2 - 1), y_3 y_1);$$

cesta β_2 (z B do B přes příkaz 2):

$$R_{\beta_2}: \quad y_2 \neq 0 \wedge \textit{sudé}(y_2), \qquad r_{\beta_2}: \quad (y_1 y_1, y_2/2, y_3);$$

cesta γ (z B do C):

$$R_\gamma: \quad y_2 = 0, \qquad r_\gamma: \quad y_3.$$

Je třeba dokázat, že pro všechna celá čísla $x_1, x_2, y_1, y_2, y_3, z$ jsou pravdivé verifikační podmínky

(a) $Q_C(\bar{x}, \bar{z}, \bar{z})$, tj. $z = z$;

(b) $\varphi(\bar{x}) \wedge Q_A(\bar{x}, \bar{z}) \supset \psi(\bar{x}, \bar{z})$,

tj.

$$x_2 \geqq 0 \wedge (x_2 \geqq 0 \supset z = x_1^{x_2}) \supset z = x_1^{x_2};$$

(c) pro cestu α:

$$Q_B(x_1, x_2, x_1, x_2, 1, z) \supset Q_A(x_1, x_2, z),$$

tj.

$$(x_2 \geqq 0 \supset z = 1 \,.\, x_1^{x_2}) \supset (x_2 \geqq 0 \supset z = x_1^{x_2});$$

pro cestu β_1:

$$y_2 \neq 0 \wedge \mathit{liché}(y_2) \wedge Q_B(x_1, x_2, y_1, y_2 - 1, y_1 y_3, z) \supset Q_B(x_1, x_2, y_1, y_2, y_3, z),$$

tj.

$$y_2 \neq 0 \wedge \mathit{liché}(y_2) \wedge ((y_2 - 1) \geqq 0 \supset z = y_3 y_1 \,.\, y_1^{(y_2 - 1)}) \supset$$
$$\supset (y_2 \geqq 0 \supset z = y_3 y_1^{y_2});$$

pro cestu β_2:

$$y_2 \neq 0 \wedge \mathit{sudé}(y_2) \wedge Q_B(x_1, x_2, y_1 y_1, y_2/2, y_3, z) \supset Q_B(x_1, x_2, y_1, y_2, y_3, z),$$

tj.

$$y_2 \neq 0 \wedge \mathit{sudé}(y_2) \wedge (y_2/2 \geqq 0 \supset z = y_3 (y_1 y_1)^{y_2/2}) \supset (y_2 \geqq 0 \supset z = y_3 y_1^{y_2});$$

pro cestu γ:

$$y_2 = 0 \wedge Q_C(y_3, z) \supset Q_B(x_1, x_2, y_1, y_2, y_3, z),$$

tj.

$$y_2 = 0 \wedge z = y_3 \supset (y_2 \geqq 0 \supset z = y_3 y_1^{y_2}).$$

Poněvadž jsou všechny verifikační podmínky pravdivé, je program P_2 parciálně korektní vzhledem k predikátům φ a ψ. ∎

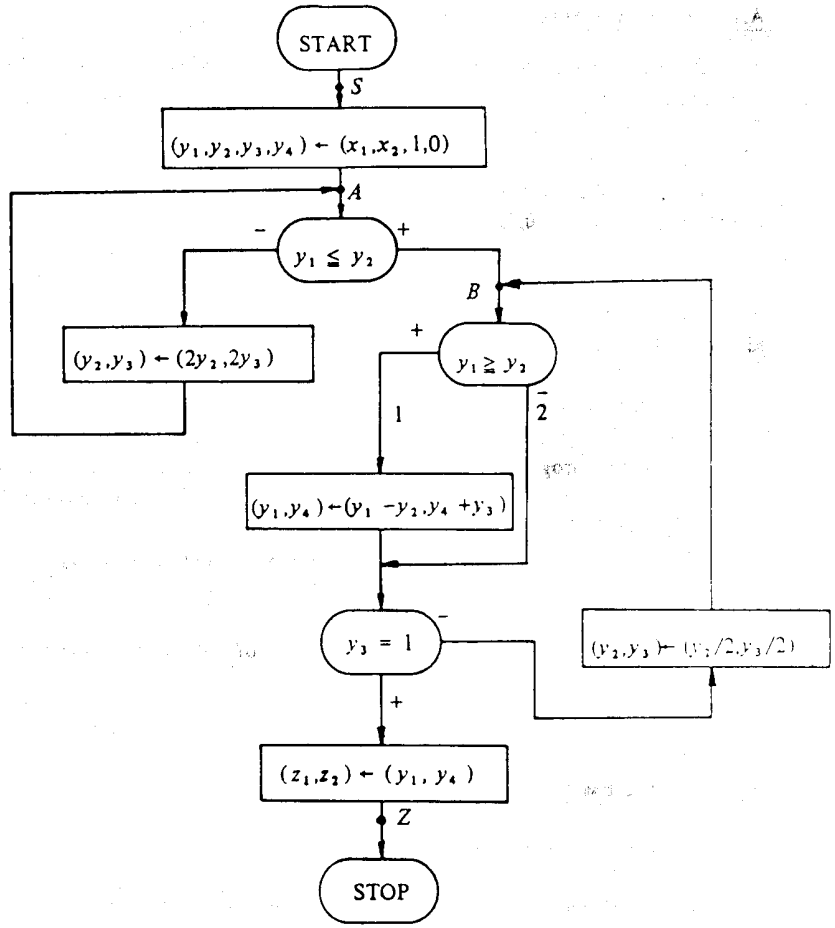
Příklad A.2

Dokažme parciální korektnost programu P nad oborem celých čísel z obr. A.2 vzhledem ke vstupnímu predikátu

$$\varphi(x_1, x_2): \quad x_1 \geqq 0 \wedge x_2 > 0$$

a výstupnímu predikátu

$$\psi(x_1, x_2, z_1, z_2): \quad z_1 = z_2 x_2 + z_1 \wedge 0 \leqq z_1 < x_2.$$



Obr. A.2. Program P pro výpočet podílu a zbytku při dělení čísla x_1 číslem x_2 (parciální korektnost)

Rozdělme nejprve cykly programu užitím dělicích bodů A a B. Pro získané cesty sestrojíme predikáty R a funkce r:

cesta α (z S do A):

R_α: T, $\qquad r_\alpha$: $(x_1, x_2, 1, 0)$;

cesta β (z A do A):

R_β: $y_1 > y_2$, $\qquad r_\beta$: $(y_1, 2y_2, 2y_3, y_4)$;

cesta γ (z A do B):

R_γ: $y_1 \leqq y_2$, $\qquad r_\gamma$: (y_1, y_2, y_3, y_4);

cesta δ_1 (z B do B podél hrany 1):

$$R_{\delta_1}: \quad y_1 \geqq y_2 \wedge y_3 \neq 1, \qquad r_{\delta_1}: \quad (y_1 - y_2, y_2/2, y_3/2, y_4 + y_3);$$

cesta δ_2 (z B do B podél hrany 2):

$$R_{\delta_2}: \quad y_1 < y_2 \wedge y_3 \neq 1, \qquad r_{\delta_2}: \quad (y_1, y_2/2, y_3/2, y_4);$$

cesta ε_1 (z B do Z podél hrany 1):

$$R_{\varepsilon_1}: \quad y_1 \geqq y_2 \wedge y_3 = 1, \qquad r_{\varepsilon_1}: \quad (y_1 - y_2, y_4 + y_3);$$

cesta ε_2 (z B do Z podél hrany 2):

$$R_{\varepsilon_2}: \quad y_1 < y_2 \wedge y_3 = 1, \qquad r_{\varepsilon_2}: \quad (y_1, y_4).$$

V důkazu použijeme tyto podcílové podmínky:

$$Q_S: \quad x_1 \geqq 0 \wedge x_2 > 0 \supset x_1 = z_2 x_2 + z_1 \wedge 0 \leqq z_1 < x_2,$$

$$Q_A: \quad y_1 \geqq 0 \wedge y_2 > 0 \wedge y_3 > 0 \wedge y_4 \geqq 0 \supset y_3 y_1 = (z_2 - y_4) y_2 + z_1 y_3 \wedge 0 \leqq z_1 y_3 < y_2,$$

$$Q_B: \quad 0 \leqq y_1 \leqq y_2 \wedge y_3 > 0 \wedge y_4 \geqq 0 \supset y_3 y_1 = (z_2 - y_4) y_2 + z_1 y_3 \wedge 0 \leqq z_1 y_3 < y_2,$$

$$Q_Z: \quad z_1 = y_1 \wedge z_2 = y_4.$$

V dalším kroku důkazu sestrojíme verifikační podmínky a ověříme jejich pravdivost pro všechna celá čísla $x_1, x_2, y_1, y_2, y_3, y_4, z_1, z_2$.

(a) $z_1 = z_1 \wedge z_2 = z_2$;

(b) $x_1 \geqq 0 \wedge x_2 > 0 \wedge Q_S(x_1, x_2, z_1, z_2) \supset \psi(x_1, x_2, z_1, z_2)$[1];

(c) pro cestu α:

$$Q_A(x_1, x_2, x_1, x_2, 1, 0, z_1, z_2) \supset Q_S(x_1, x_2, z_1, z_2),$$

pro cestu β:

$$y_1 > y_2 \wedge Q_A(x_1, x_2, y_1, 2y_2, 2y_3, y_4, z_1, z_2) \supset Q_A(\bar{x}, \bar{y}, \bar{z}),$$

tj.

$$y_1 > y_2 \wedge (y_1 \geqq 0 \wedge 2y_2 > 0 \wedge 2y_3 > 0 \wedge y_4 \geqq 0 \supset 2y_3 y_1 = (z_2 - y_4) 2y_2 + z_1 2y_3 \wedge 0 \leqq z_1 2y_3 < 2y_2) \supset$$
$$\supset (y_1 \geqq 0 \wedge y_2 > 0 \wedge y_3 > 0 \wedge y_4 \geqq 0 \supset y_3 y_1 = (z_2 - y_4) y_2 + z_1 y_3 \wedge 0 \leqq z_1 y_3 < y_2),$$

pro cestu γ:

$$y_1 \leqq y_2 \wedge Q_B(\bar{x}, \bar{y}, \bar{z}) \supset Q_A(\bar{x}, \bar{y}, \bar{z}),$$

pro cestu δ_1:

$$y_1 \geqq y_2 \wedge y_3 \neq 1 \wedge Q_B(x_1, x_2, y_1 - y_2, y_2/2, y_3/2, y_4 + y_3, z_1, z_2) \supset Q_B(\bar{x}, \bar{y}, \bar{z}),$$

[1] Verifikační podmínky, jejichž pravdivost je zřejmá, nebudeme rozepisovat podrobně.

tj.

$$y_1 \geqq y_2 \wedge y_3 \neq 1 \wedge (0 \leqq y_1 - y_2 \leqq y_2/2 \wedge y_3/2 > 0 \wedge y_4 + y_3 \geqq 0 \supset$$
$$\supset y_3/2 . (y_1 - y_2) = (z_2 - y_4 - y_3) . y_2/2 + z_1 y_3/2 \wedge 0 \leqq z_1 y_3/2 < y_2/2) \supset$$
$$\supset (0 \leqq y_1 \leqq y_2 \wedge y_3 > 0 \wedge y_4 \geqq 0 \supset$$
$$\supset y_3 y_1 = (z_2 - y_4) y_2 + z_1 y_2 \wedge 0 \leqq z_1 y_3 < y_2),$$

pro cestu δ_2:

$$y_1 < y_2 \wedge y_3 \neq 1 \wedge Q_B(x_1, x_2, y_1, y_2/2, y_3/2, y_4, z_1, z_2) \supset Q_B(\bar{x}, \bar{y}, \bar{z}),$$

pro cestu ε_1:

$$y_1 \geqq y_2 \wedge y_3 = 1 \wedge z_1 = y_1 - y_2 \wedge z_2 = y_4 + y_3 \supset Q_B(\bar{x}, \bar{y}, \bar{z}),$$

tj.

$$y_1 \geqq y_2 \wedge y_3 = 1 \wedge z_1 = y_1 - y_2 \wedge z_2 = y_4 + y_3 \supset$$
$$\supset (0 \leqq y_1 \leqq y_2 \wedge y_3 > 0 \wedge y_4 \geqq 0 \supset y_3 y_1 =$$
$$= (z_2 - y_4) y_2 + z_1 y_3 \wedge 0 \leqq z_1 y_3 < y_2),$$

pro cestu ε_2:

$$y_1 < y_2 \wedge y_3 = 1 \wedge z_1 = y_1 \wedge z_2 = y_4 \supset Q_B(\bar{x}, \bar{y}, \bar{z}).$$

Snadno se ověří, že všechny verifikační podmínky jsou pravdivé, a tedy program P je parciálně korektní vzhledem k predikátům φ a ψ.

■

A.2. METODA PODCÍLOVÝCH PRAVIDEL

A.2.1. Princip metody

Metodu podcílových podmínek lze aplikovat na důkaz parciální korektnosti strukturovaných programů. Uvedeme axiomatický systém, obdobný systému z odst. 3.3.3. Jediným podstatným rozdílem bude jiné pravidlo pro příkaz cyklu.

K důkazu parciální korektnosti strukturovaných programů uvažujeme *induktivní výraz*

$$\{p(\bar{x}, \bar{y})\}\, B\{q(\bar{x}, \bar{y}, \bar{y}')\},$$

který vyjadřuje tvrzení, že kdykoliv je $p(\bar{x}, \bar{y})$ pravdivé před provedením segmentu B a výpočet segmentu B končí, pak po jeho ukončení platí $q(\bar{x}, \bar{y}, \bar{y}')$, kde $\bar{y}'$ je hodnota programového vektoru po provedení B. Induktivní výraz

$$\{\varphi(\bar{x})\}\, P\{\psi(\bar{x}, \bar{z})\}$$

tedy vyjadřuje parciální korektnost programu P vzhledem ke vstupnímu predikátu φ a výstupnímu predikátu ψ.

Uvedeme pravidla, jejichž pomocí můžeme získávat induktivní výrazy pro větší segmenty na základě již získaných induktivních výrazů pro malé segmenty.

Podcílová pravidla:

1. *Pravidlo přiřazení*

$$\frac{\forall\bar{x}\forall\bar{y}[p(\bar{x},\bar{y}) \supset q(\bar{x},\bar{y},f(\bar{x},\bar{y}))]}{\{p(\bar{x},\bar{y})\}\ \bar{y} \leftarrow f(\bar{x},\bar{y})\ \{q(\bar{x},\bar{y},\bar{y}')\}}.$$

2. *Pravidlo sekvence*

$$\frac{\begin{array}{c}\{p_1(\bar{x},\bar{y})\}\ B_1\{q_1(\bar{x},\bar{y},\bar{y}')\},\\ \{p_2(\bar{x},\bar{y})\}\ B_2\{q_2(\bar{x},\bar{y},\bar{y}')\},\\ \forall\bar{x}\forall\bar{y}\forall\bar{y}'[q_1(\bar{x},\bar{y},\bar{y}') \supset p_2(\bar{x},\bar{y}')],\\ \forall\bar{x}\forall\bar{y}\forall\bar{y}'\forall\bar{y}''[q_1(\bar{x},\bar{y},\bar{y}') \wedge q_2(\bar{x},\bar{y},\bar{y}'') \supset q(\bar{x},\bar{y},\bar{y}'')]\end{array}}{\{p_1(\bar{x},\bar{y})\}\ B_1; B_2\{q(\bar{x},\bar{y},\bar{y}')\}}.$$

3. *Pravidla podmínky*

$$\frac{\begin{array}{c}\{p(\bar{x},\bar{y}) \wedge t(\bar{x},\bar{y})\}\ B_1\{q(\bar{x},\bar{y},\bar{y}')\},\\ \{p(\bar{x},\bar{y}) \wedge \sim t(\bar{x},\bar{y})\}\ B_2\{q(\bar{x},\bar{y},\bar{y}')\}\end{array}}{\{p(\bar{x},\bar{y})\}\ \textbf{if}\ t(\bar{x},\bar{y})\ \textbf{then}\ B_1\ \textbf{else}\ B_2\{q(\bar{x},\bar{y},\bar{y}')\}}.$$

a

$$\frac{\begin{array}{c}\{p(\bar{x},\bar{y}) \wedge t(\bar{x},\bar{y})\}\ B\{q(\bar{x},\bar{y},\bar{y}')\},\\ \forall\bar{x}\forall\bar{y}[p(\bar{x},\bar{y}) \wedge \sim t(\bar{x},\bar{y}) \supset q(\bar{x},\bar{y},\bar{y})]\end{array}}{\{p(\bar{x},\bar{y})\}\ \textbf{if}\ t(\bar{x},\bar{y})\ \textbf{do}\ B\{q(\bar{x},\bar{y},\bar{y}')\}}.$$

4. *Pravidlo cyklu*

$$\frac{\begin{array}{cc}\{p(\bar{x},\bar{y}) \wedge t(\bar{x},\bar{y})\}\ B\{q_1(\bar{x},\bar{y},\bar{y}')\}, & (4.1)\\ \forall\bar{x}\forall\bar{y}\forall\bar{y}'\forall\bar{y}''[p(\bar{x},\bar{y}) \wedge t(\bar{x},\bar{y}) \wedge q_1(\bar{x},\bar{y},\bar{y}') \wedge q(\bar{x},\bar{y}',\bar{y}'') \supset q(\bar{x},\bar{y},\bar{y}'')], & (4.2)\\ \forall\bar{x}\forall\bar{y}[p(\bar{x},\bar{y}) \wedge \sim t(\bar{x},\bar{y}) \supset q(\bar{x},\bar{y},\bar{y})] & (4.3)\end{array}}{\{p(\bar{x},\bar{y})\}\ \textbf{while}\ t(\bar{x},\bar{y})\ \textbf{do}\ B\{q(\bar{x},\bar{y},\bar{y}')\}}.$$

Uvedená pravidla se liší od pravidel ukončení pouze v pravidle cyklu. Induktivní výraz (4.1) charakterizuje činnost segmentu B. Podmínka (4.2) pak reprezentuje vlastní induktivní krok a je společně s podmínkou (4.3) přímou aplikací verifikačních podmínek pro cykl u metody podcílových podmínek. Jako dodatečná pravidla můžeme použít i pravidla konsekvence, disjunkce a konjunkce z odst. 3.3.3.

Techniku důkazu parciální korektnosti strukturovaného programu můžeme vyjádřit ve tvaru věty.

Věta A.2 (Metoda podcílových pravidel). *Buď dán strukturovaný program P, vstupní predikát $\varphi(\bar{x})$ a výstupní predikát $\psi(\bar{x},\bar{z})$. Jestliže lze postupnou aplikací uvedených podcílových pravidel odvodit induktivní výraz*

$$\{\varphi(\bar{x})\}\ P\{\psi(\bar{x},\bar{z})\},$$

pak je program P parciálně korektní vzhledem k predikátům φ a ψ.

A.2.2. Příklad

Ukažme si použití metody podcílových pravidel na konkrétním příkladě.

Příklad A.3

Dokažme parciální korektnost programu P_9 nad oborem celých čísel z př. 3.13 vzhledem k vstupnímu predikátu

$$\varphi(x_1,x_2):\quad x_1 \geqq 0 \wedge x_2 > 0$$

a výstupnímu predikátu

$$\psi(x_1,x_2,z):\quad z = nsd(x_1,x_2).$$

Program P_9 je

START
$(y_1,y_2) \leftarrow (x_1,x_2)$;
while $y_1 \neq y_2$ **do**
begin
while $y_1 > y_2$ **do** $y_1 \leftarrow y_1 - y_2$;
while $y_2 > y_1$ **do** $y_2 \leftarrow y_2 - y_1$;
end;
$z \leftarrow y_1$
STOP

Naším cílem je odvodit induktivní výraz

$$\{x_1 \geqq 0 \wedge x_2 > 0\}\ P_9\{z = nsd(x_1,x_2)\}.$$

Důkaz rozdělíme do sedmnácti kroků:

1. Podle pravidla přiřazení je

$$\{y_1 = nsd(x_1,x_2)\}\ z \leftarrow y_1\{z = nsd(x_1,x_2)\}.$$

2. Podle pravidla přiřazení je

$$\{x_1 \geqq 0 \wedge x_2 > 0\}\ (y_1,y_2) \leftarrow (x_1,x_2)\ \{y_1 \geqq 0 \wedge y_2 > 0\}.$$

3. Podle pravidla přiřazení je

$$\{y_1 \geqq 0 \wedge y_2 > 0 \wedge y_1 > y_2\}\ y_1 \leftarrow y_1 - y_2\{y_1' = y_1 - y_2 \wedge y_2' = y_2\}.$$

4. Podle pravidla přiřazení je

$$\{y_1 \geqq 0 \wedge y_2 > 0 \wedge y_2 > y_1\}\ y_2 \leftarrow y_2 - y_1\{y_1' = y_1 \wedge y_2' = y_2 - y_1\}.$$

5. $y_1 \geqq 0 \wedge y_2 > 0 \wedge y_1 \neq y_2 \wedge y_1 \leqq y_2 \supset y_1 =$ Lemma 1
$= zbytek(y_1, y_2) \wedge y_2 = y_2$.

6. $y_1 \geqq 0 \wedge y_2 > 0 \wedge y_1 \neq y_2 \wedge y_1 > y_2 \wedge y_1' =$ Lemma 2
$= y_1 - y_2 \wedge y_2' = y_2 \wedge y_1'' = zbytek(y_1', y_2') \wedge y_2'' =$
$= y_2' \supset y_1'' = zbytek(y_1, y_2) \wedge y_2'' = y_2$.

7. Z kroku 3, Lemmatu 1 a Lemmatu 2 dostáváme podle pravidla cyklu

$\{y_1 \geqq 0 \wedge y_2 > 0 \wedge y_1 \neq y_2\}$
while $y_1 > y_2$ **do** $y_1 \leftarrow y_1 - y_2$
$\{y_1 = zbytek(y_1, y_2) \wedge y_2' = y_2\}$.

8. $y_1 \geqq 0 \wedge y_2 > 0 \wedge y_1 \geqq y_2 \supset y_1 = y_1 \wedge y_2 = zbytek(y_2, y_1)$. Lemma 3

9. $y_1 \geqq 0 \wedge y_2 > 0 \wedge y_1 < y_2 \wedge y_1' = y_1 \wedge y_2' =$ Lemma 4
$= y_2 - y_1 \wedge y_1'' = y_1' \wedge y_2'' = zbytek(y_2', y_1') \supset y_1'' =$
$= y_1 \wedge y_2'' = zbytek(y_2, y_1)$.

10. Z kroku 4, Lemmatu 3 a Lemmatu 4 dostáváme podle pravidla cyklu

$\{y_1 \geqq 0 \wedge y_2 > 0\}$
while $y_2 > y_1$ **do** $y_2 \leftarrow y_2 - y_1$
$\{y_1 = y_1 \wedge y_2' = zbytek(y_2, y_1)\}$.

11. $y_1' = zbytek(y_1, y_2) \wedge y_2' = y_2 \wedge y_2 > 0 \supset y_1' \geqq 0 \wedge y_2' > 0$. Lemma 5

12. $y_1' = zbytek(y_1, y_2) \wedge y_2' = y_2 \wedge y_2'' = zbytek(y_2', y_1') \wedge y_1'' =$ Lemma 6
$= y_1' \supset y_1'' = zbytek(y_1, y_2) \wedge y_2'' = zbytek(y_2, zbytek(y_1, y_2))$.

13. Z kroku 7, 10 a Lemmat 5, 6 dostáváme

$\{y_1 \geqq 0 \wedge y_2 > 0 \wedge y_1 \neq y_2\}$
while $y_1 > y_2$ **do** $y_1 \leftarrow y_1 - y_2$;
while $y_2 > y_1$ **do** $y_2 \leftarrow y_2 - y_1$
$\{y_1 = zbytek(y_1, y_2) \wedge y_2' = zbytek(y_2, zbytek(y_1, y_2))\}$.

14. $y_1 \geqq 0 \wedge y_2 > 0 \wedge y_1 = y_2 \supset y_1 = nsd(y_1, y_2)$. Lemma 7

15. $y_1 \geqq 0 \wedge y_2 > 0 \wedge y_1 \neq y_2 \wedge y_1' = zbytek(y_1, y_2) \wedge y_2' =$ Lemma 8
$= zbytek(y_2, zbytek(y_1, y_2)) \wedge y_1'' = nsd(y_1', y_2') \supset y_1'' =$
$= nsd(y_1, y_2)$.

16. Z kroku 13, Lemmat 7, 8 dostáváme podle pravidla cyklu

$\{y_1 \geqq 0 \wedge y_2 > 0\}$
while $y_1 \neq y_2$ **do begin**
⋮
end
$\{y_1 = nsd(y_1, y_2)\}$.

17. Z kroků 1, 2, 16 a zřejmých lemmat, která neuvádíme, dostáváme

$\{x_1 \geqq 0 \wedge x_2 > 0\}\, P_9 \{z = nsd(x_1, x_2)\}$.

Tím jsme ukončili důkaz parciální korektnosti programu P_9 vzhledem k φ a ψ.

K důkazu samotnému ještě poznamenejme, že důkazy jednotlivých lemmat jsme vynechali. Jediným problémem by mohl být důkaz lemmatu 8, které využívá této vlastnosti největšího společného dělitele:

$$nsd(y_1, y_2) = nsd(zbytek(y_1, y_2), y_2) = nsd(y_1, zbytek(y_2, y_1)).$$

■

A.3. METODA INTERMITENTNÍCH PODMÍNEK

A.3.1. Princip metody

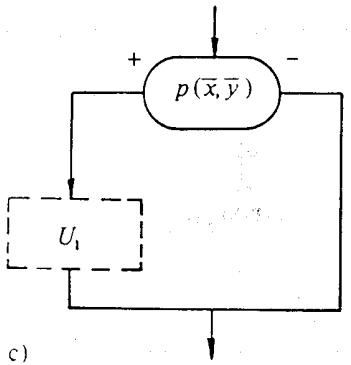
Verifikací programu rozumíme důkaz totální korektnosti programu.[1]) Jedinou metodou, která umožnila verifikovat program přímo, byla metoda pravidel ukončení. Důkaz totální korektnosti programu touto metodou je však vzhledem ke značné formálnosti dosti obtížný. V tomto odstavci proto ukážeme jinou metodu – metodu intermitentních podmínek, kterou lze dokázat totální korektnost programu rovněž přímo, a to přirozenějším (méně formálním) způsobem. Důkaz totální korektnosti programu je veden způsobem blízkým běžným matematickým důkazům. Tvrzení o totální korektnosti programu je vyjádřeno ve tvaru věty, k jejímuž důkazu užíváme pomocné věty – lemmata. Charakter metody intermitentních podmínek nedovoluje uvést metodu způsobem použitým u dřívějších metod, čtenáři nelze dát přesný návod, jak postupovat při důkazu. Omezíme se proto na uvedení základní myšlenky důkazu metodou intermitentních podmínek a na několik konkrétních příkladů.

Důkaz totální korektnosti programu metodou intermitentních podmínek požaduje, podobně jako u jiných metod, další dodatečné informace o programu vyjádřené ve tvaru podmínek. Tyto podmínky, nutné ke zvládnutí cyklů, však mají u metody intermitentních podmínek jiný charakter než např. u metody induktivních podmínek. Na induktivní podmínku připojenou k jistému místu v programu je kladen požadavek, aby podmínka byla splněna pro průběžné hodnoty proměnných *vždy*, když výpočet prochází tímto místem. Takový požadavek invariantnosti, kladený na induktivní podmínky, je příliš silný a lze v něm spatřovat i jednu z hlavních příčin obtížnosti nalezení vhodných induktivních podmínek.

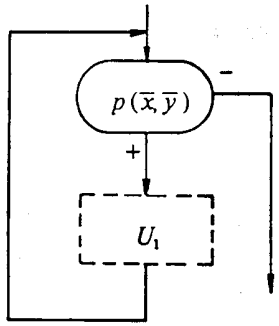
Hlavní myšlenkou metody intermitentních podmínek je právě odstranění požadavku invariantnosti. Podmínky, které přiřazujeme k dělicím bodům programu, nemusí být pravdivé vždy, ale pouze *někdy*. Přesněji řečeno, podmínky, které uvažujeme, jsou takového charakteru, že alespoň jednou musí výpočet projít dělicím bodem s tím, že podmínka bude pravdivá pro průběžné hodnoty proměnných. Při tom výpočet může projít dělicím bodem mnohokrát, aniž by podmínka byla pravdivá. Podmínky s uvedenou vlastností budeme nazývat *intermitentní podmínky*.[2])

[1]) Viz str. 160.

[2]) Z angl. intermittent – občasný.



c)



d)

(iii) Jsou-li $U_1, \ldots, U_m$, $m \geqq 0$, segmenty, V výchozí příkaz a Z příkaz zastavení, pak V; U_1; ...; U_m; Z je *strukturované schéma*.

Uvedené diagramy [spolu s konstatováním, že bod (iii) odpovídá diagramu analogickému k (ii) (a)] ukazují, že ke každému strukturovanému schématu (popř. jeho vhodné části) lze sestrojit vývojový diagram ve smyslu čl. 4.1, tj. jisté programové schéma. Třídu strukturovaných schémat proto považujeme za podtřídu programových schémat; podobně tomu bude i se schématy z odst. B.2.3. Ne každé programové schéma je strukturované (viz např. schéma z obr. 4.6a) – třída strukturovaných schémat je tedy vlastní podtřídou třídy programových schémat. Zajímá nás nyní otázka, zda-li tato třída je univerzální.

Chceme-li k programovému schématu S nalézt ekvivalentní strukturované programové schéma S', budeme při jeho konstrukci používat predikátové konstanty výchozího schématu S, definice programového schématu však nepřipouští kombinovat je do složitějších formulí. Ve větě B.3 ukážeme, že v některých kontextech „nevadí" používání spojek $\sim$, $\vee$, $\wedge$, obecně je však, striktně vzato, používat v ekvivalentních úpravách nesmíme. Odtud vyplyne, že není možné ke každému programovému schématu nalézt ekvivalentní „čisté" strukturované schéma: obohatíme-li však třídu strukturovaných schémat o celkem přirozenou a technicky nepříliš podstatnou možnost obecnějších „booleovských testů", podaří se převést každé schéma do kanonického strukturovaného tvaru (věta B.4).

Věta B.3. *Nechť p, q jsou predikátové konstanty a U, V segmenty. Pak existují segmenty ekvivalentní těmto zápisům:*

(a) **if** $\sim p(\bar{x}, \bar{y})$ **then** U **else** V

(b) **if** $p(\bar{x}, \bar{y}) \vee q(\bar{x}, \bar{y})$ **then** U **else** V a **while** $p(\bar{x}, \bar{y}) \vee q(\bar{x}, \bar{y})$ **do** U

(c) **if** $p(\bar{x}, \bar{y}) \wedge q(\bar{x}, \bar{y})$ **then** U **else** V

Na druhé straně však:

(d) *k programovému schématu (s booleovskými testy)*

S: START $y \leftarrow x$; **while** $p(y) \wedge q(y)$ **do** $y \leftarrow f(y)$; $z \leftarrow y$ **STOP**

neexistuje žádné ekvivalentní jednoduché strukturované programové schéma.

Důkaz.

(a) vyplývá přímo z definice podmíněného příkazu, která dovoluje záměnu větví „+" a „−";

(b) **if** $p(\bar{x}, \bar{y}) \vee q(\bar{x}, \bar{y})$ **then** U **else** V je ekvivalentní s

if $p(\bar{x}, \bar{y})$ **then** U **else if** $q(\bar{x}, \bar{y})$ **then** U **else** V;

podobně

while $p(\bar{x}, \bar{y}) \vee q(\bar{x}, \bar{y})$ **do** U

je ekvivalentní s

while $p(\bar{x}, \bar{y})$ **do** U;
while $q(\bar{x}, \bar{y})$ **do begin** U;
while $p(\bar{x}, \bar{y})$ **do** U
end

(c) **if** $p(\bar{x}, \bar{y}) \wedge q(\bar{x}, \bar{y})$ **then** U **else** V je ekvivalentní s

if $p(\bar{x}, \bar{y})$ **then if** $q(\bar{x}, \bar{y})$ **then** U **else** V **else** V;

(d) Předpokládejme, že ke schématu S existuje ekvivalentní jednoduché strukturované schéma S' a že má N příkazů. Uvažme Herbrandovu interpretaci $\mathscr{I}_1^*$, pro niž $p(„f^{(n)}(x)") = q(„f^{(n)}(x)") = pravda$ pro $0 \leqq n \leqq N + 1$ ($„f^{(0)}(x)" = „x"$) a $p(„f^{(N+2)}(x)") = q(„f^{(N+2)}(x)") = nepravda$. Je $h\langle S', \mathscr{I}_1^*, „x"\rangle = h\langle S, \mathscr{I}_1^*, „x"\rangle = „f^{(N+2)}(x)"$. Poněvadž schéma S' je jednoduché, je k výpočtu $„f^{(N+2)}(x)"$ zapotřebí provedení alespoň $N + 1$ příkazů, takže některý musí být proveden vícekrát. Poněvadž S' je strukturované, musí být takové opakování zajištěno v rámci vhodného příkazu **while** ... **do** .... Ze čtyř možností, totiž

while $p(y')$ **do** B, **while** $\sim p(y')$ **do** B, **while** $q(y')$ **do** B, **while** $\sim q(y')$ **do** B

rozebereme první, ostatní se vyšetří analogicky; y' je tu libovolná programová proměnná.

Uvažujme k tomu Herbrandovu interpretaci $\mathscr{I}_2^*$, v níž predikát q je interpretován stejně jako v $\mathscr{I}_1^*$, zatímco predikát p nabývá hodnoty *pravda* pro všechny hodnoty argumentu. Před tím, než výpočet pro $\langle S', \mathscr{I}_2^*, „x"\rangle$ zahájí uvažovaný cykl **while** $p(y')$ **do** B, probíhá stejně jako výpočet pro $\langle S', \mathscr{I}_1^*, „x"\rangle$, neboť hodnoty všech proměnných jsou zatím tvaru $„f^{(n)}(x)"$ pro $n \leqq N$ a pro ně obě interpretace splývají. Avšak $h\langle S, \mathscr{I}_2^*, „x"\rangle = „f^{(N+2)}(x)"$, zatímco hodnota $h\langle S', \mathscr{I}_2^*, „x"\rangle$ není definována, neboť $p(y') = pravda$ pro všechny hodnoty y' a výpočet pro $\langle S', \mathscr{I}_2^*, „x"\rangle$ neopustí uvažovaný cykl. Schéma S' tedy nemůže být ekvivalentní schématu S.

Jednoduchým důsledkem části (d) věty B.3 je skutečnost, že ani tak primitivní programové schéma, jako je schéma znázorněné na obr. B.5, nelze převést na ekvivalentní strukturované schéma. K tomuto faktu lze zaujmout dvě stanoviska: buď je možné povolit rozšíření pojmu programového schématu při zachování formální strukturace požadované definicí strukturovaných schémat, nebo naopak