

Programová schémata

Předmět : Teorie programů

Idea

Programové schéma je program, v němž není specifikován ani obor hodnot proměnných (je pouze označen symbolem D), ani funkce a predikáty, které v něm figurují (jsou označeny funkčními symboly f_1, f_2, \dots , a predikátovými symboly p_1, p_2, \dots). Programové schéma tedy reprezentuje celý soubor programů, z nichž každý lze získat ze schématu vhodnou interpretaci všech symbolů, které se v něm vyskytují, tj. stanovením množiny D a funkcí a predikátů odpovídajících použitým symbolům f_i, g_i .

Základní pojmy a vlastnosti programových schémat jsou uvedeny v čl. 4.1.

Programy vznikají interpretací symbolů programového schématu zcela obdobným způsobem, jako interpretací formulí predikátového počtu vznikají výroky. Vztahy mezi interpretovanými formulemi predikátového počtu a programy a vztahy mezi (neinterpretovanými) formulemi a programovými schématy jsou studovány v čl. 4.3.

Programové schéma popisuje strukturu programu a přesouva specifikaci řady

Základní pojmy - syntaxe

Abeceda Σ_S daného programového schématu S je konečná podmnožina následující množiny symbolů:

1. Konstanty:

- (a) n -árni funkční konstanty f_i^n ($i \geq 1, n \geq 0$); f_i^0 se nazývají individuové konstanty a označujeme je též a_i ,
- (b) n -árni predikátové konstanty p_i^n ($i \geq 1, n \geq 0$); p_i^0 se nazývají výrokové konstanty.

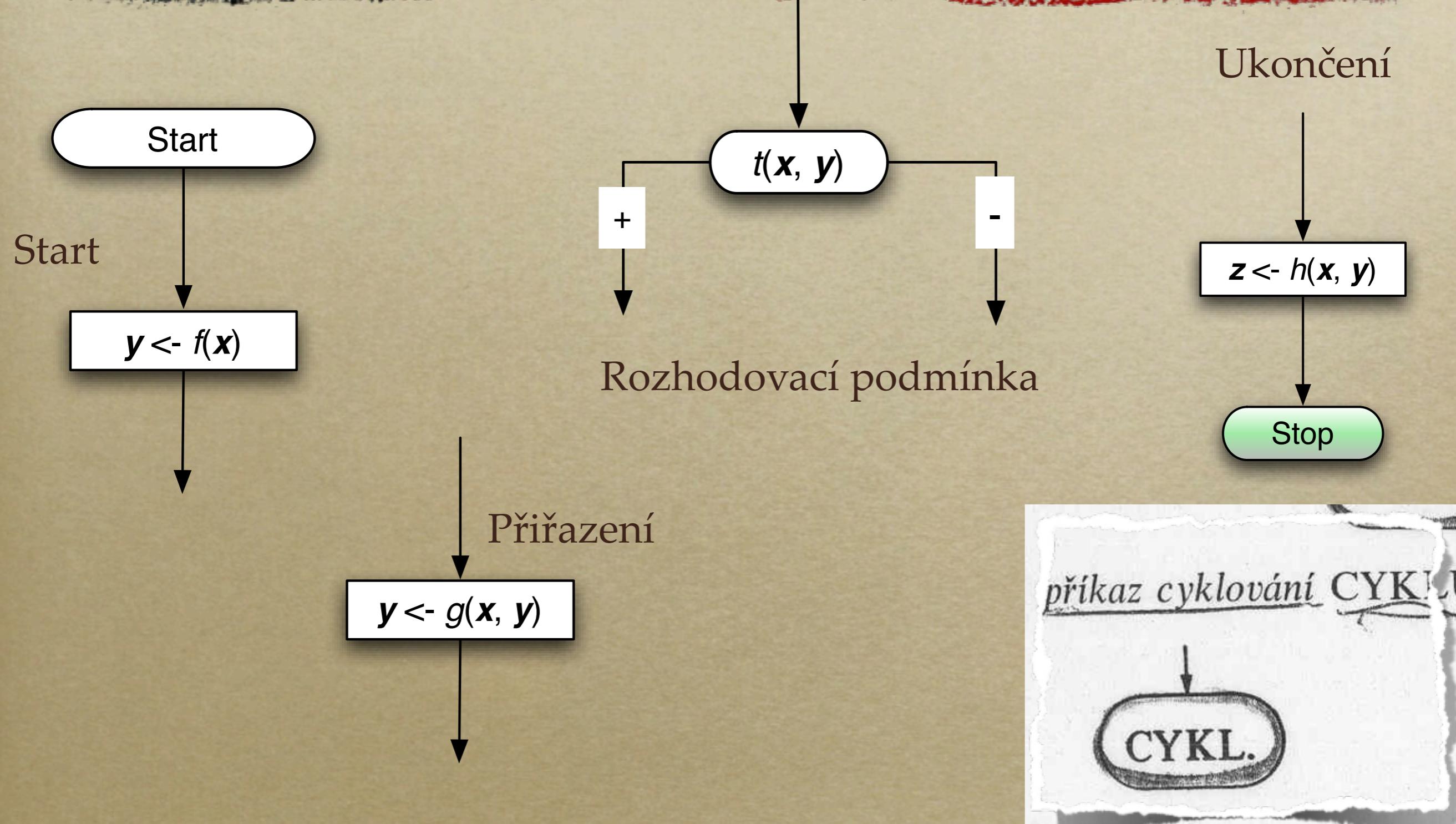
2. Individuové proměnné:

- (a) vstupní proměnné x_i ($i \geq 1$),
- (b) programové proměnné y_i ($i \geq 1$),
- (c) výstupní proměnné z_i ($i \geq 1$).

Počet vstupních proměnných \bar{x}^1 , programových proměnných \bar{y} a výstupních proměnných \bar{z} obsažených v Σ_S označíme po řadě a, b, c , $a, b, c \geq 0$. Dolní indexy u symbolů slouží k jejich rozlišení a budou vypuštěny v případě, že nebude hrozit nebezpečí z nedorozumění. Totéž platí o horních indexech u f_i^n a p_i^n , které označuj počet argumentů.

Takže σ nad Σ vznikne i něžnou konstrukcí z individuových proměnných

Základní pojmy - syntaxe



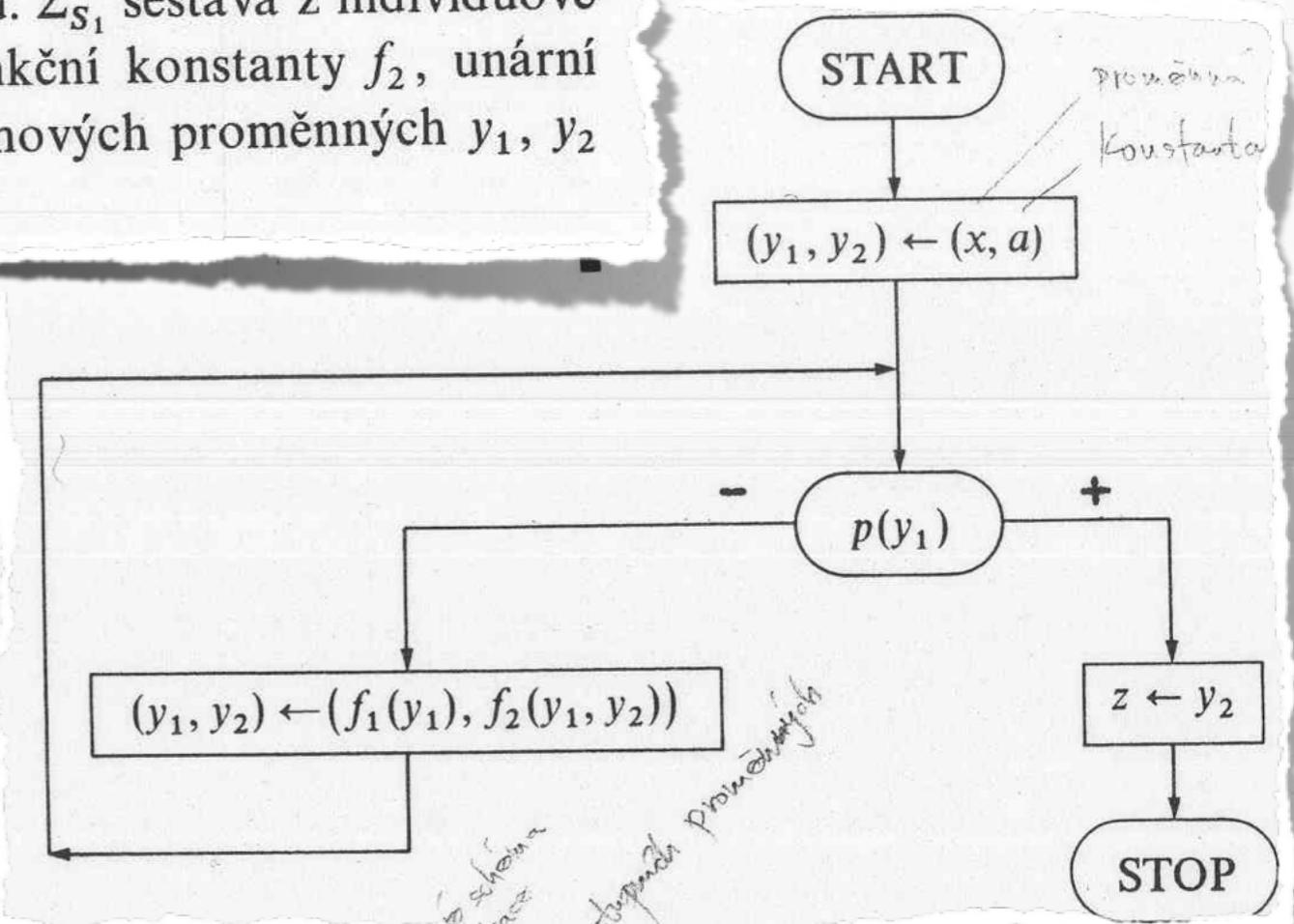
Základní pojmy - syntaxe

Všechny dříve uvedené termov a atomické formulace jsou nad Σ_S .

Programové schéma S nad abecedou Σ_S (stručně též: schéma) je vývojový diagram sestrojený z příkazů nad Σ_S tak, že příkaz START je použit právě jednou a každý přiřazovací a testovací příkaz leží na nějaké cestě od příkazu START k některému příkazu STOP nebo CYKLUJE.

Příklad 4.1

Schéma S_1 z obr. 4.1 využijeme i v dalším výkladu. Σ_{S_1} sestává z individuové konstanty a , unární funkční konstanty f_1 , binární funkční konstanty f_2 , unární predikátové konstanty p , vstupní proměnné x , programových proměnných y_1, y_2 a výstupní proměnné z .



Základní pojmy - sémantika (interpretace)

Interpretace \mathcal{I} programového schématu S sestává z

1. neprázdné množiny D nazývané *obor interpretace*,
2. interpretace konstant z Σ_S :

(a) Každé funkční konstantě f_i^n z Σ_S je přiřazena úplná funkce zobrazující D^n do D (je-li $n = 0$, je individuové konstantě f_i^0 přiřazen jistý prvek z D). $D^n \rightarrow D$

(b) Každé predikátové konstantě p_i^n z Σ_S je přiřazen úplný predikát zobrazující D^n do množiny $\{\text{pravda}, \text{nepravda}\}$ (je-li $n = 0$, je výrokové konstantě p_i^0 přiřazena pravdivostní hodnota *pravda* nebo *nepravda*). $D^n \rightarrow \{\text{pravda}, \text{nepravda}\}$

Dvojice $P = \langle S, \mathcal{I} \rangle$, kde S je programové schéma a \mathcal{I} je jeho interpretace, se nazývá **program**. Jsou-li dány vstupní hodnoty $\xi \in D^a$ vstupních proměnných \bar{x} schématu S (tj. *vstupní vektor*), program P může být proveden.

Základní pojmy - sémantika (interpretace)

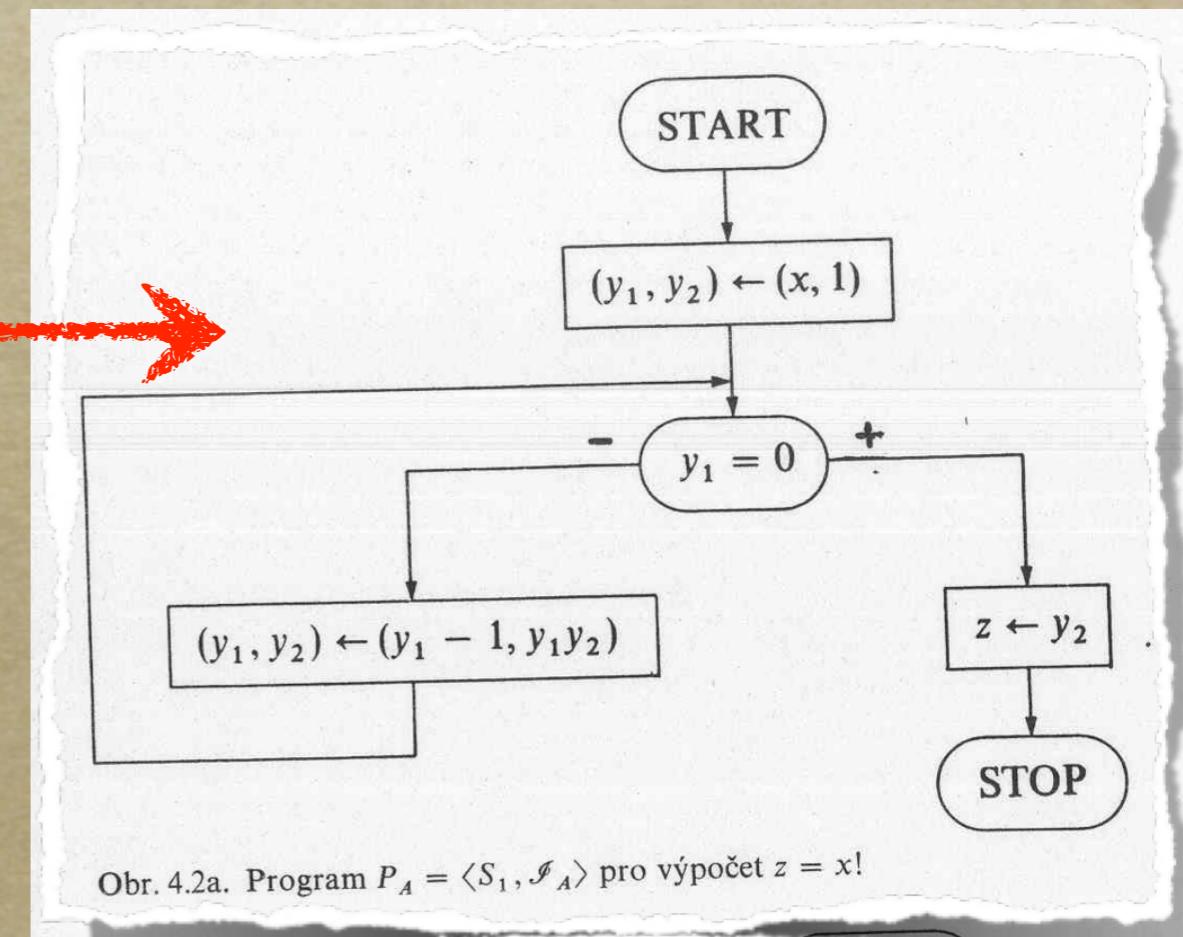
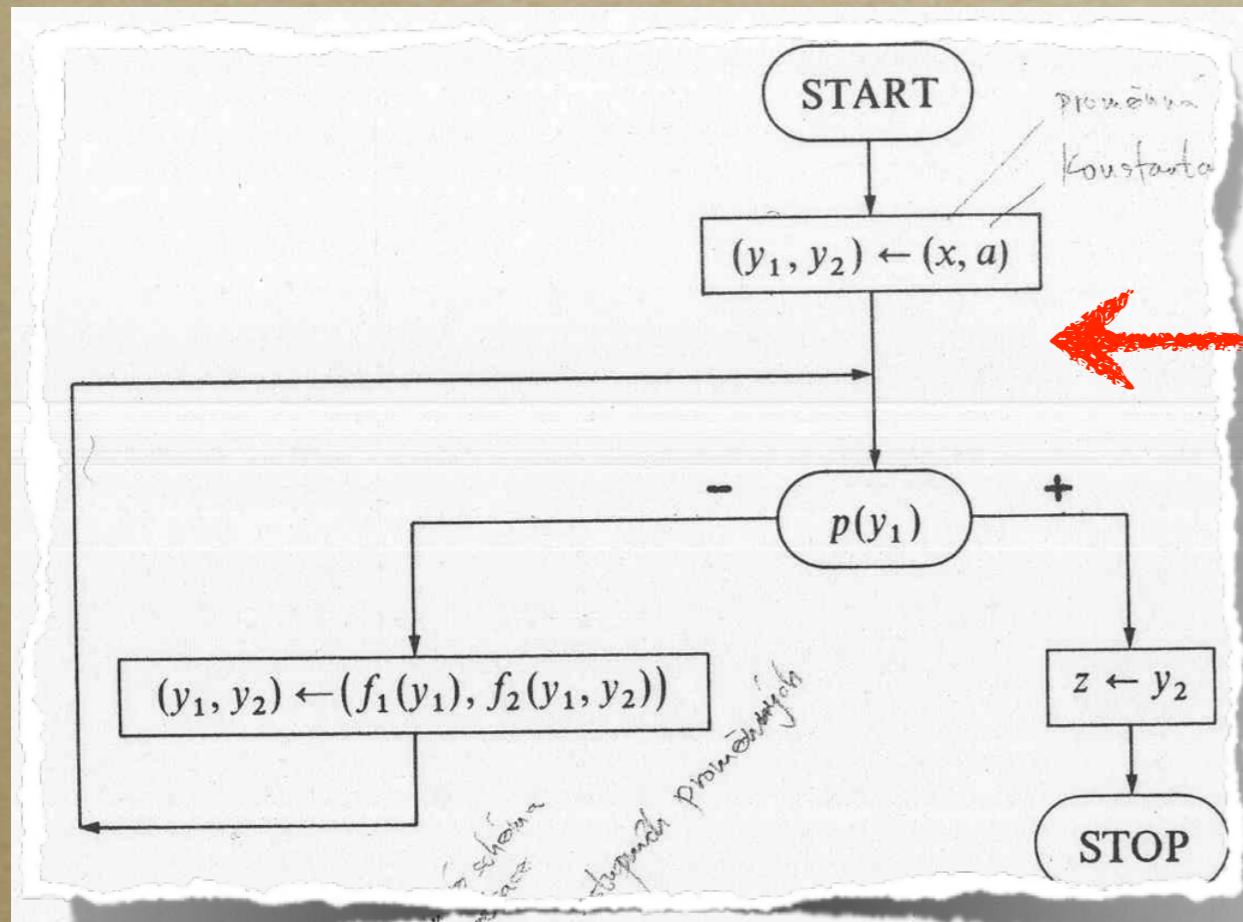
Výpočet pro $\langle S, \mathcal{I}, \xi \rangle$ probíhá obvyklým způsobem; začíná výchozím příkazem **START**, v němž dochází k inicializaci proměnných (y_1, \dots, y_b) hodnotami $(\tau_1(\xi), \dots, \tau_b(\xi))$, neboť $\bar{x} = \xi$.¹ Provedení přiřazovacího příkazu $(y_1, \dots, y_b) \leftarrow (\tau_1(\bar{x}, \bar{y}), \dots, \tau_b(\bar{x}, \bar{y}))$ v situaci, kdy $\bar{y} = \bar{\eta}$ pro jisté $\bar{\eta} \in D^b$, vede k hodnotě $(y_1, \dots, y_b) = (\tau_1(\xi, \bar{\eta}), \dots, \tau_b(\xi, \bar{\eta}))$; nové hodnoty y_1, \dots, y_b jsou vyčíslovány současně. Tedy např. výměnu hodnot proměnných y_1 a y_2 můžeme jednoduše provést příkazem $(y_1, y_2) \leftarrow (y_2, y_1)$. Výpočet končí, jakmile narazíme na příkaz zastavení (**STOP**) nebo cyklování (**CYKLUJE**). V prvním případě je po provedení příkazu **STOP** $\bar{z} = \xi \in D^c$; řekneme pak, že $h\langle S, \mathcal{I}, \xi \rangle$ ² je definováno a klademe $h\langle S, \mathcal{I}, \xi \rangle = \bar{z}$. V druhém případě (jestliže výpočet přejde k příkazu **CYKLUJE**) nebo v případě, že výpočet nikdy nekončí, není $h\langle S, \mathcal{I}, \xi \rangle$ definováno. Každý program tedy představuje částečné zobrazení z množiny D^a do množiny D^c .

Základní pojmy - sémantika (interpretace)

Dve z možných interpretací:

1. Interpretace \mathcal{I}_A : obor interpretace D je tvořen množinou přirozených čísel; a je 1; $f_1(y_1)$ je $y_1 - 1$; $f_2(y_1, y_2)$ je $y_1 y_2$; $p(y_1)$ je $y_1 = 0$. Program $P_A = \langle S_1, \mathcal{I}_A \rangle$ zobrazený na obr. 4.2a vyčísluje funkci faktoriál, tj. $z = x!$

2. Interpretace \emptyset : obor interpretace tvoří množina všech slov nad danou

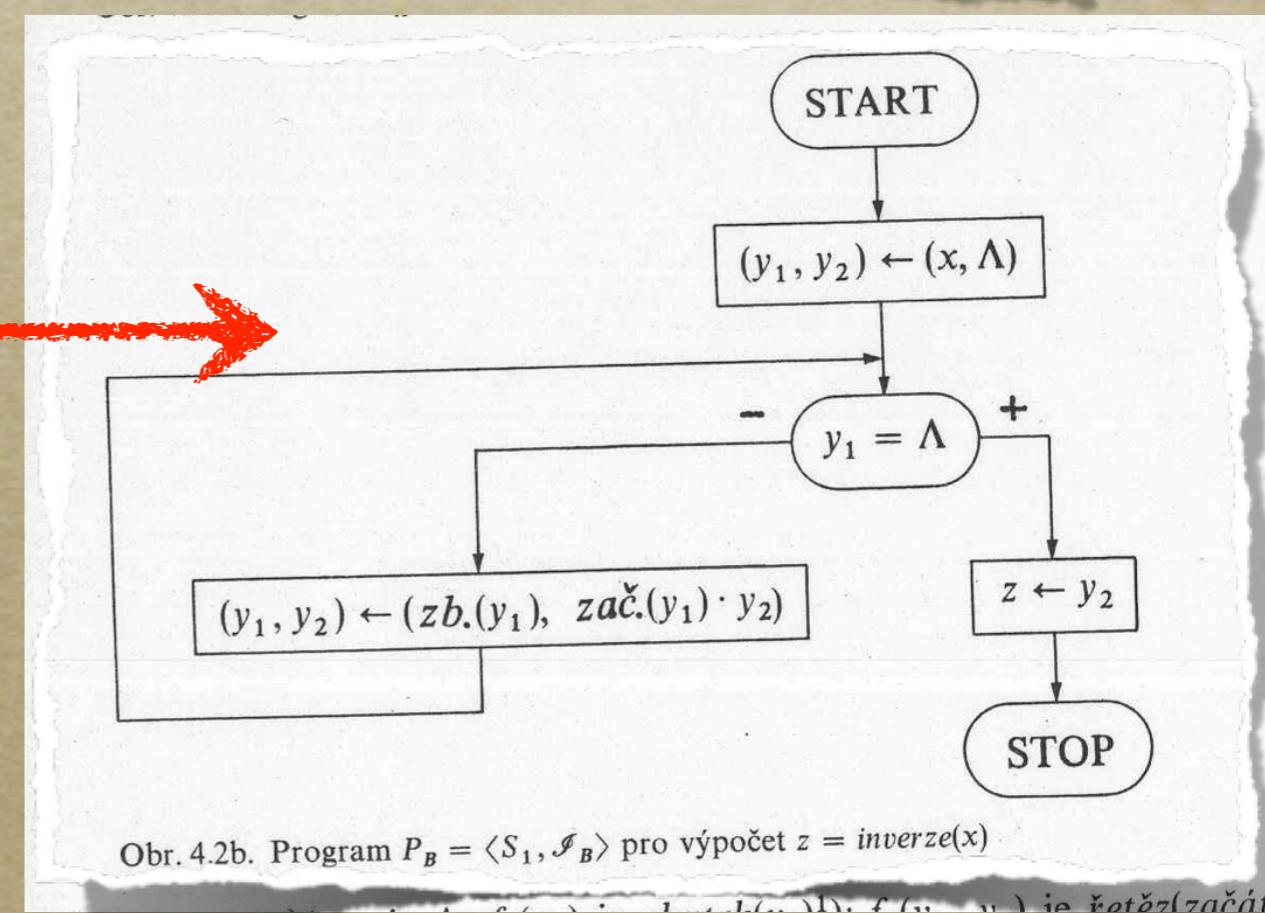
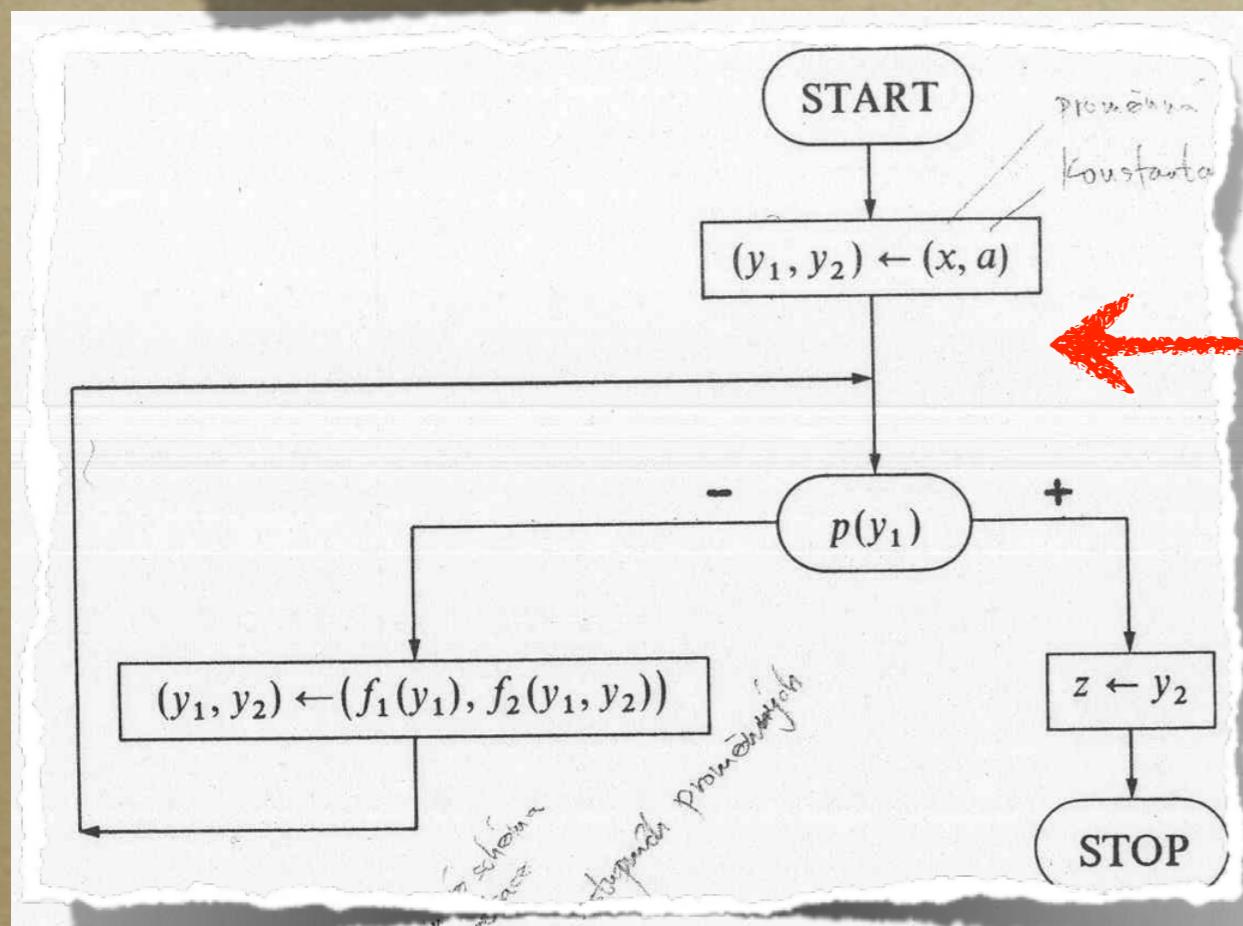


Obr. 4.2a. Program $P_A = \langle S_1, \mathcal{I}_A \rangle$ pro výpočet $z = x!$

Základní pojmy - sémantika (interpretace)

~~= $\langle S_1, \mathcal{I}_B \rangle$ zobrazený na obr. 4.2a využívající funkci funkce~~

2. Interpretace \mathcal{I}_B : obor interpretace tvoří množina všech slov nad danou konečnou abecedou $\Sigma_B = \{A, B, \dots, Z\}$ včetně prázdného slova Λ , tj. $D = \{A, B, \dots, Z\}^*$; a je Λ ; $f_1(y_1)$ je $zbytek(y_1)^1$; $f_2(y_1, y_2)$ je řetěz($začátek(y_1)$, y_2); $p(y_1)$ je $y_1 = \Lambda$. Program $P_B = \langle S_1, \mathcal{I}_B \rangle$ zobrazený na obr. 4.2b obrací pořadí písmen v daném slově, tj. $z = inverze(x)$.



Obr. 4.2b. Program $P_B = \langle S_1, \mathcal{I}_B \rangle$ pro výpočet $z = inverze(x)$

Základní pojmy - vlastnosti

(A) Ukončení a divergence

Řekneme, že daný program $\langle S, \mathcal{I} \rangle$

1. končí, jestliže pro každý vstupní vektor $\xi \in D^a$ je $h\langle S, \mathcal{I}, \xi \rangle$ definováno;
2. diverguje, jestliže pro žádný vstupní vektor $\xi \in D^a$ není $h\langle S, \mathcal{I}, \xi \rangle$ definováno.

Může se ovšem stát, že program $\langle S, \mathcal{I} \rangle$ ani nekončí, ani nediverguje, že totiž pro některé $\xi_1 \in D^a$ je $h\langle S, \mathcal{I}, \xi_1 \rangle$ definováno, zatímco pro jiné $\xi_2 \in D^a$ není $h\langle S, \mathcal{I}, \xi_2 \rangle$ definováno.

Řekneme, že dané schéma S

1. končí, jestliže pro každou interpretaci \mathcal{I} program $\langle S, \mathcal{I} \rangle$ končí;
2. diverguje, jestliže pro každou interpretaci \mathcal{I} program $\langle S, \mathcal{I} \rangle$ diverguje.

Základní pojmy - ekvivalence

(B) Ekvivalence

Nejvýznamnějším vztahem mezi dvojicí schémat je bezpochyby jejich ekvivalence; její definice však vyžaduje několik pomocných pojmu.

Řekneme, že schémata S a S' jsou kompatibilní, mají-li společný vektor vstupních proměnných \bar{x} i vektor výstupních proměnných \bar{z} . Dva programy $\langle S, \mathcal{I} \rangle$ a $\langle S', \mathcal{I}' \rangle$ jsou kompatibilní, jsou-li kompatibilní schémata S a S' a interpretace \mathcal{I} a \mathcal{I}' mají týž obor.¹⁾

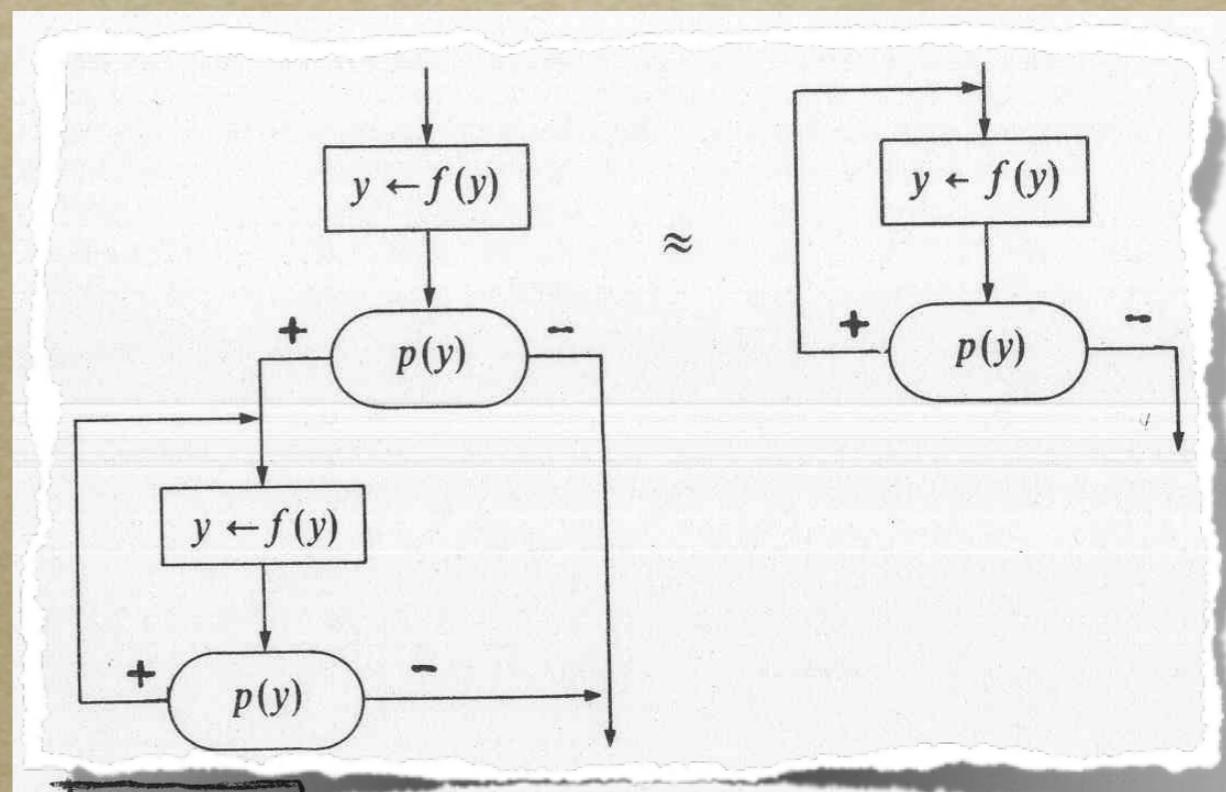
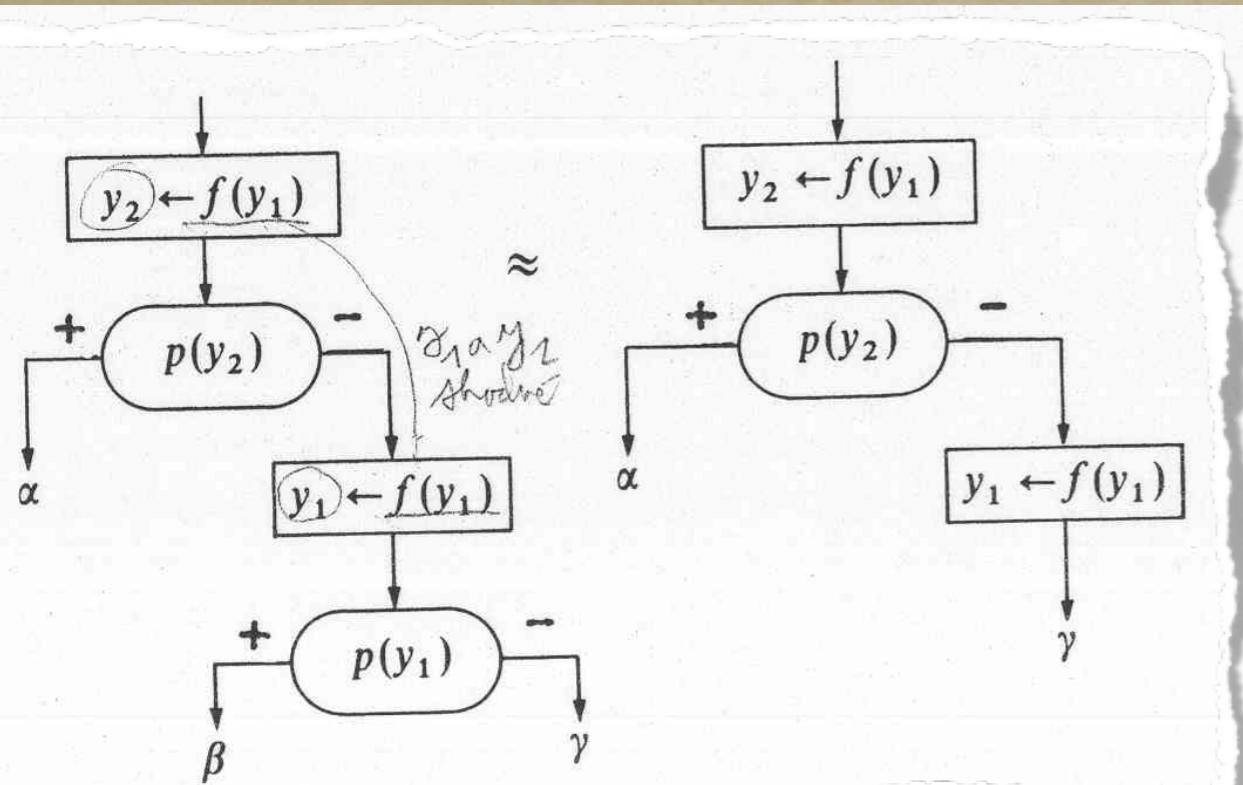
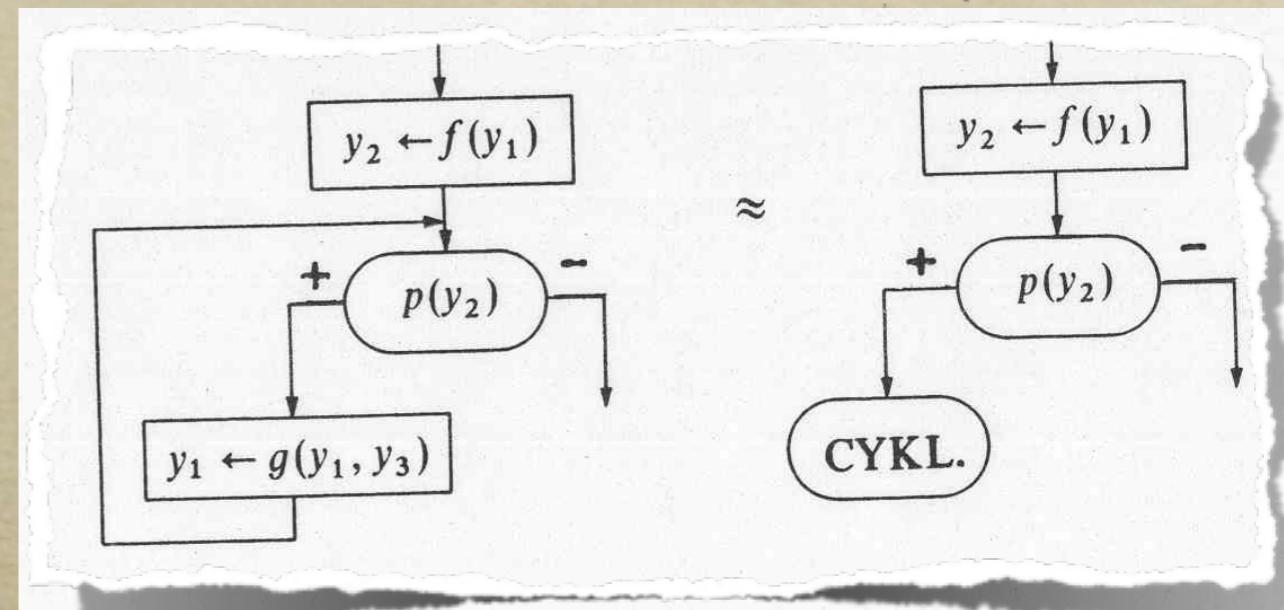
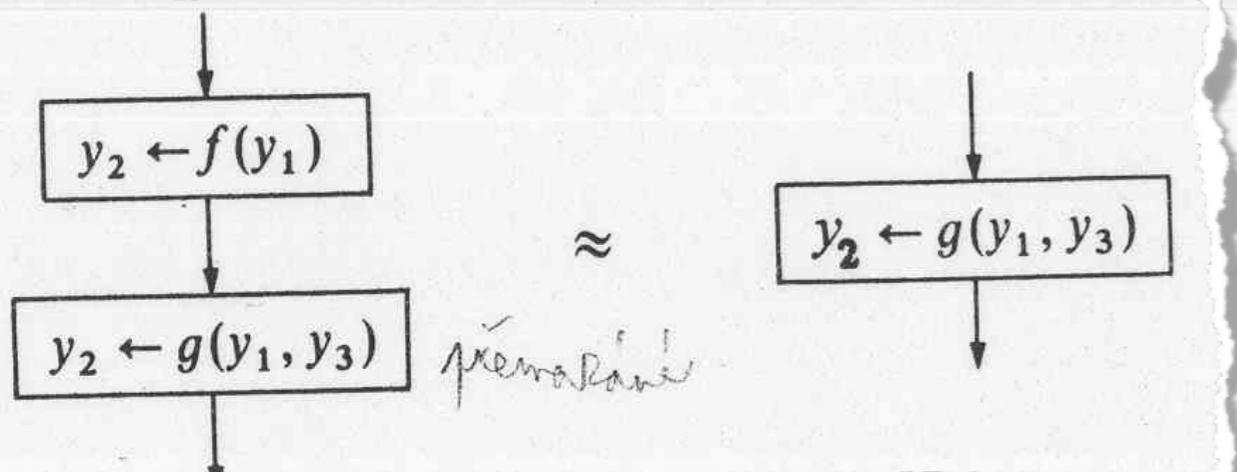
Řekneme, že dva kompatibilní programy $\langle S, \mathcal{I} \rangle$ a $\langle S', \mathcal{I}' \rangle$ jsou ekvivalentní (a píšeme $\langle S, \mathcal{I} \rangle \approx \langle S', \mathcal{I}' \rangle$), jestliže pro každý vstupní vektor $\xi \in D^a$ je $h\langle S, \mathcal{I}, \xi \rangle \equiv h\langle S', \mathcal{I}', \xi \rangle$ ¹⁾, tj. jestliže jsou buď hodnoty $h\langle S, \mathcal{I}, \xi \rangle$ i $h\langle S', \mathcal{I}', \xi \rangle$ nedefinovány, nebo jsou obě definovány a platí $h\langle S, \mathcal{I}, \xi \rangle = h\langle S', \mathcal{I}', \xi \rangle$.

Řekneme, že dvě kompatibilní schémata S a S' jsou ekvivalentní (a píšeme

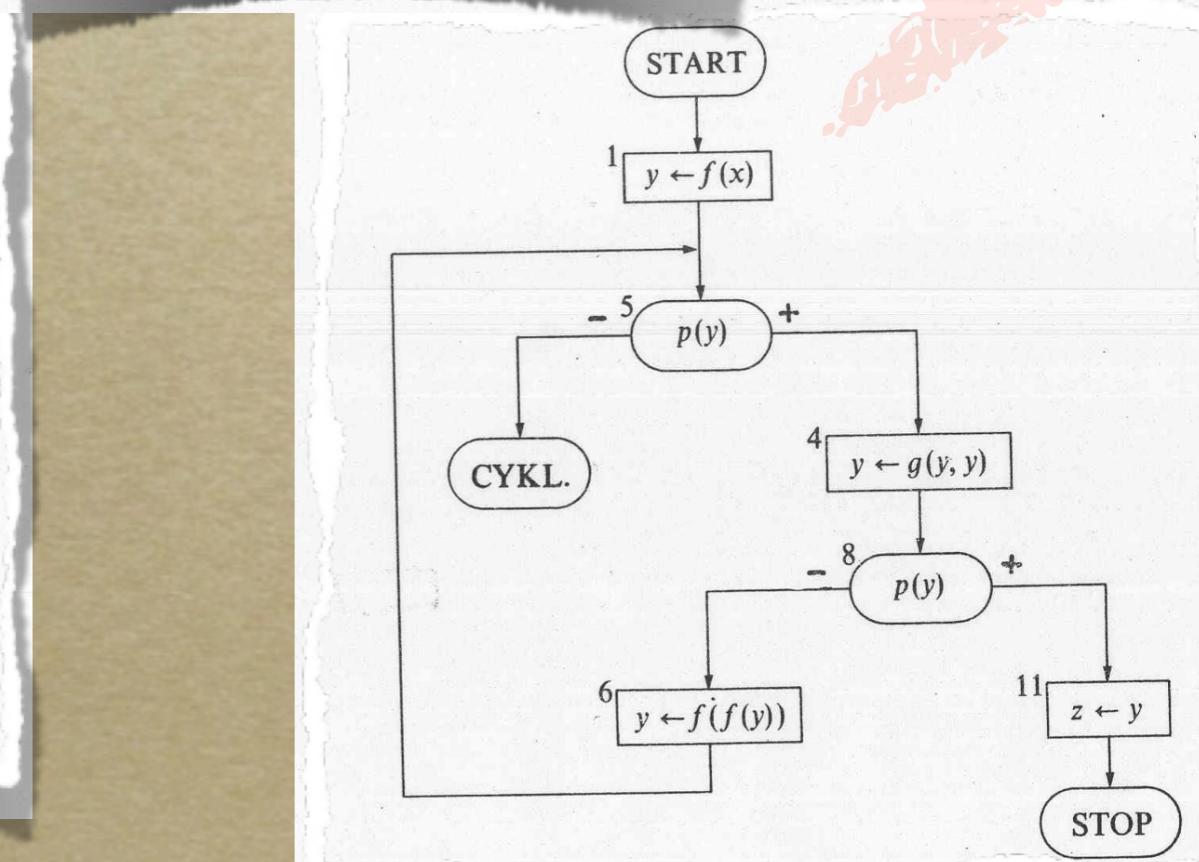
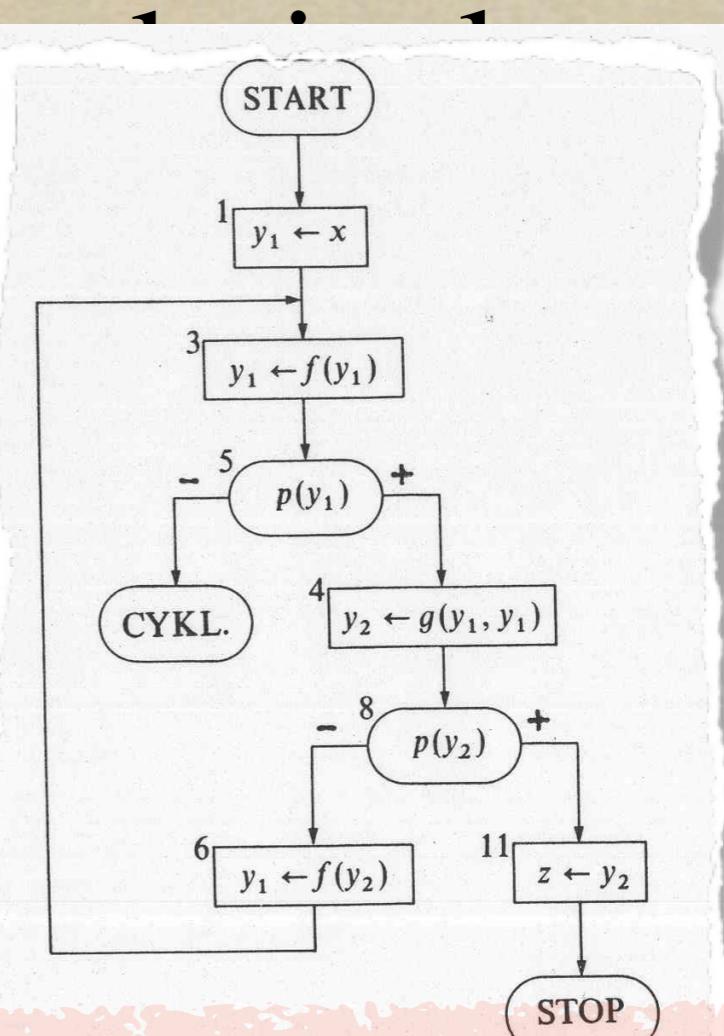
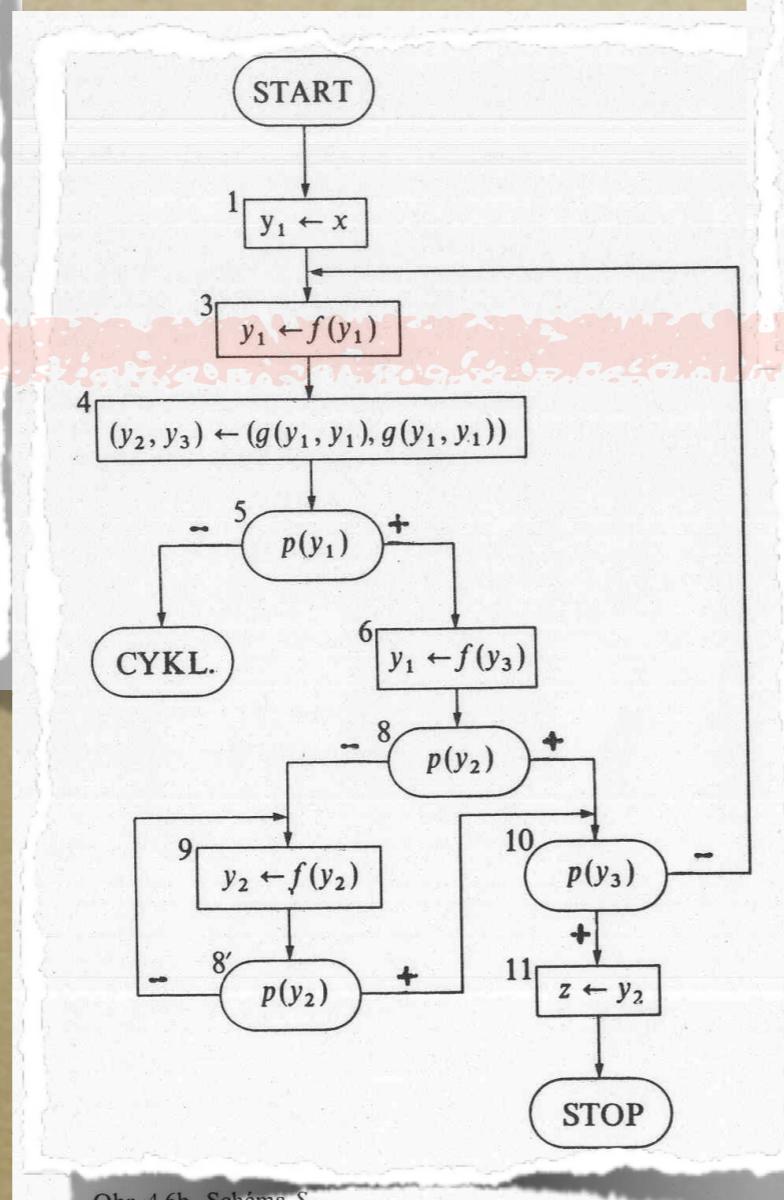
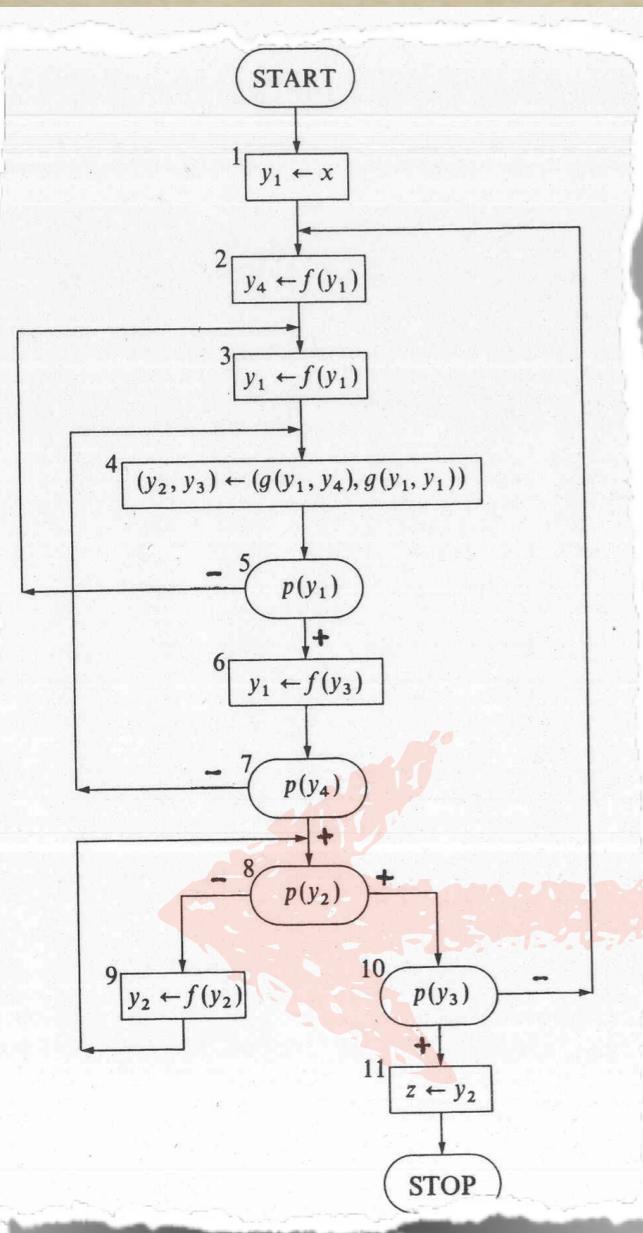
$S \approx S'$), jestliže pro každou interpretaci \mathcal{I} schémat S, S' jsou programy $\langle S, \mathcal{I} \rangle$ a $\langle S', \mathcal{I} \rangle$ ekvivalentní.

Definované ekvivalence jsou ekvivalence v matematickém slova smyslu, tj. jsou to relace reflexívni, symetrické i tranzitivní. Tranzitivita ekvivalence lze využít při dokazování ekvivalence dvou schémat pomocí řetězu jednoduchých úprav, z nichž každá zachovává ekvivalenci. Tak lze například použít transformace, opírající se o tyto ekvivalence:

Základní pojmy - ekvivalence



Základní pojmy



Obr. 4.6b. Schéma S1

Obr. 4.6e. Schéma S4

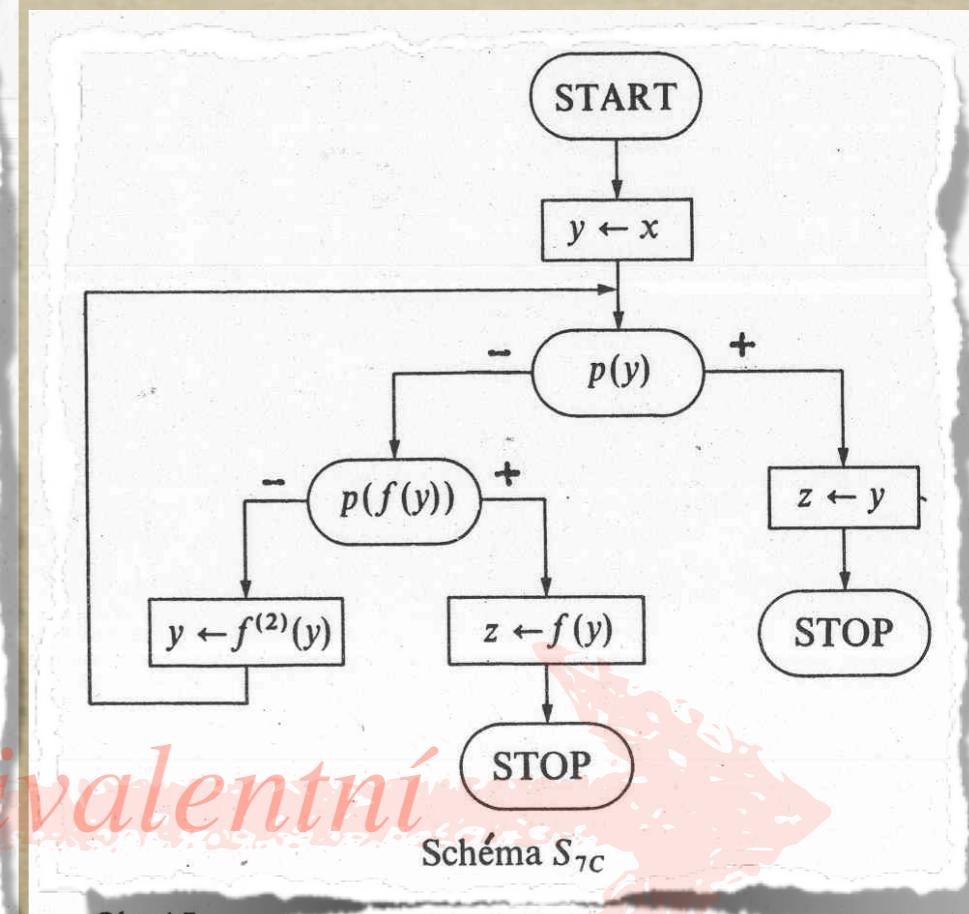
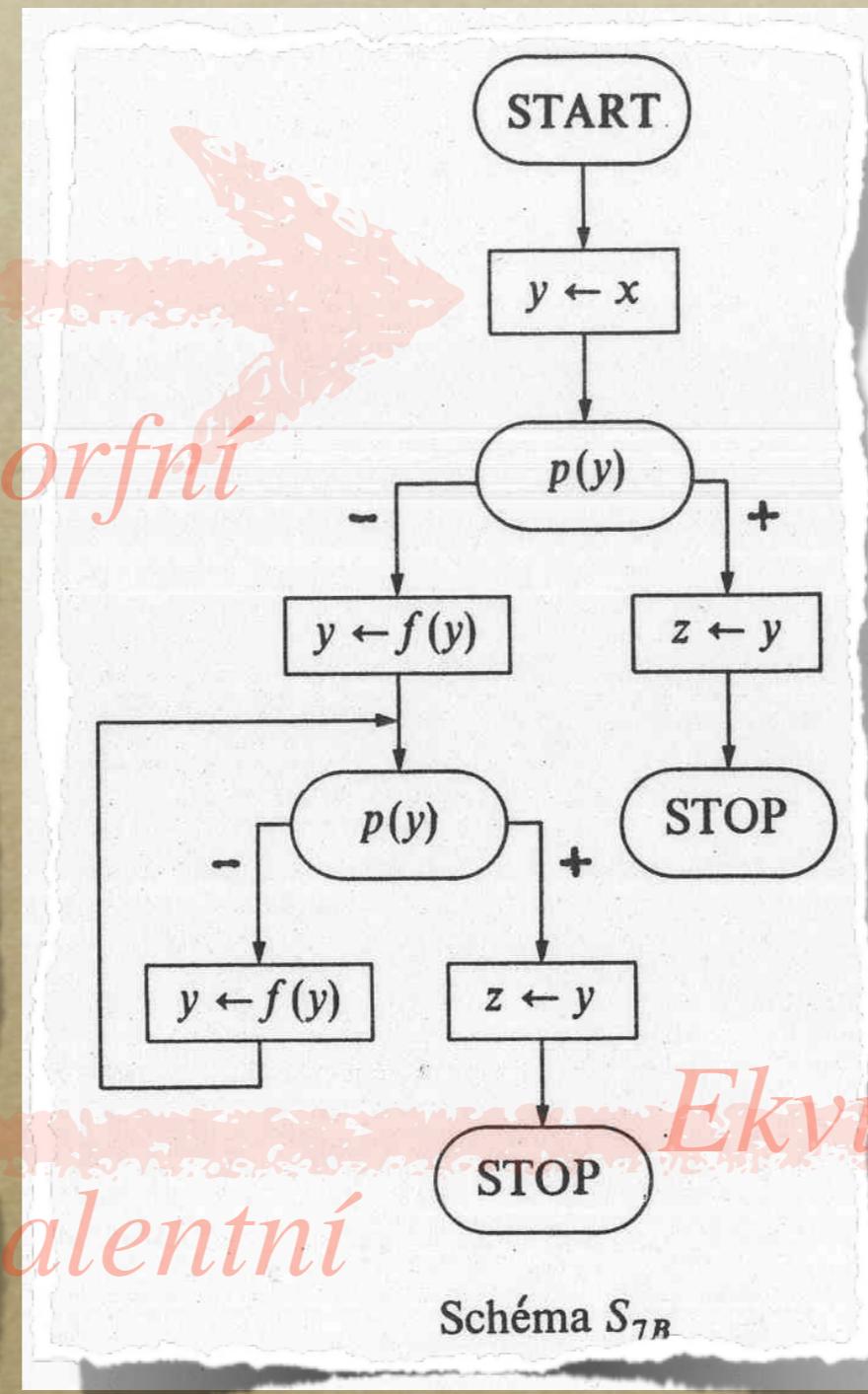
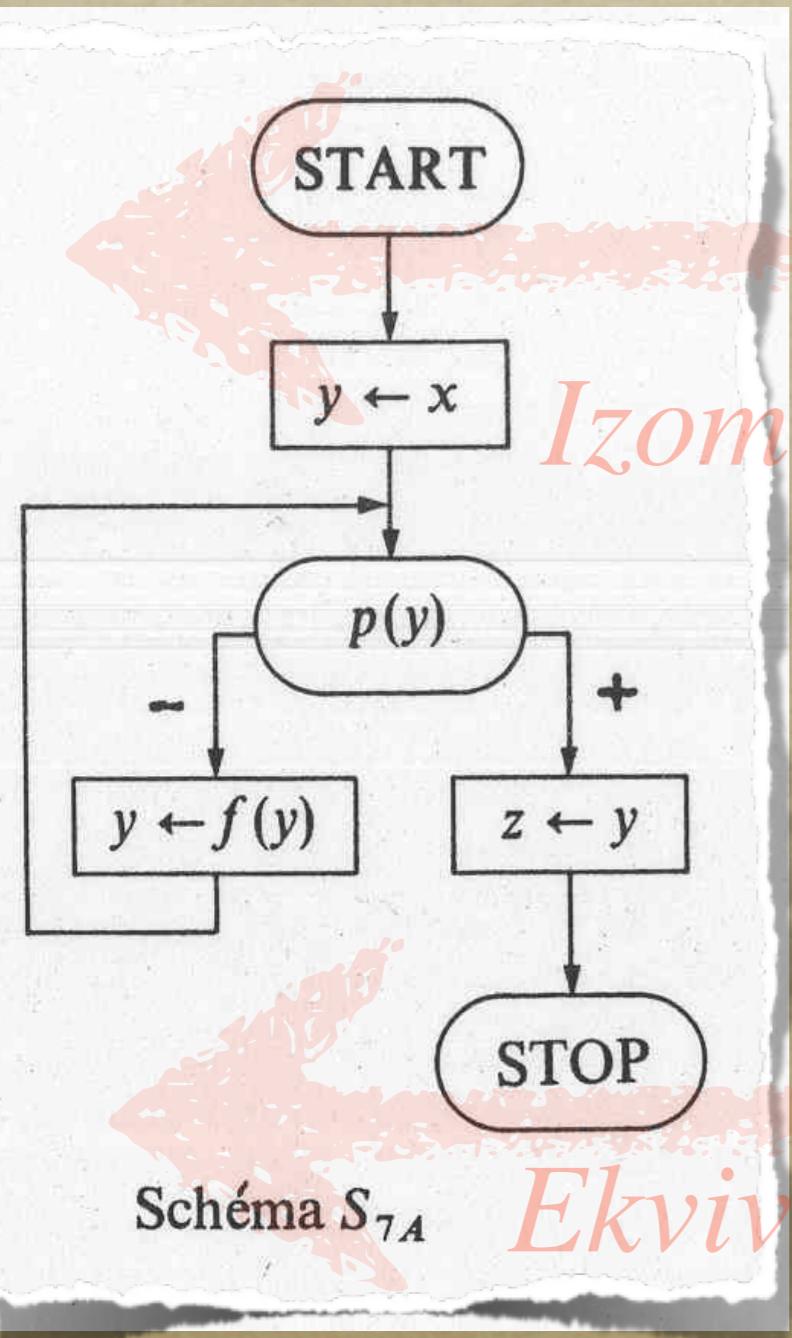
Základní pojmy - izomorfismus

Někdy nás zajímá nejen to, zda daná dvě schémata S a S' dávají v dané interpretaci tytéž výsledné hodnoty, ale také to, zda tyto hodnoty byly počítány stejným způsobem. Dostáváme se tak k silnější formě ekvivalence, tzv. izomorfismu. Řekneme, že kompatibilní schémata S a S' jsou izomorfní²⁾ (a píšeme $S \approx S'$), jestliže pro každou interpretaci \mathcal{I} a každý vstupní vektor $\xi \in D^a$ je (konečná či nekonečná) posloupnost příkazů prováděných při výpočtu pro $\langle S, \mathcal{I}, \xi \rangle$ tatáž jako posloupnost příkazů prováděna při výpočtu pro $\langle S', \mathcal{I}, \xi \rangle$.

Příklad 4.7

Schémata S_{7A} , S_{7B} a S_{7C} z obr. 4.7 jsou kompatibilní. Je zřejmé, že $S_{7A} \approx S_{7B} \approx S_{7C}$, ale $S_{7A} \approx S_{7B} \not\approx S_{7C}$.

Základní pojmy - izomorfizmus



Herbrandovy interpretace

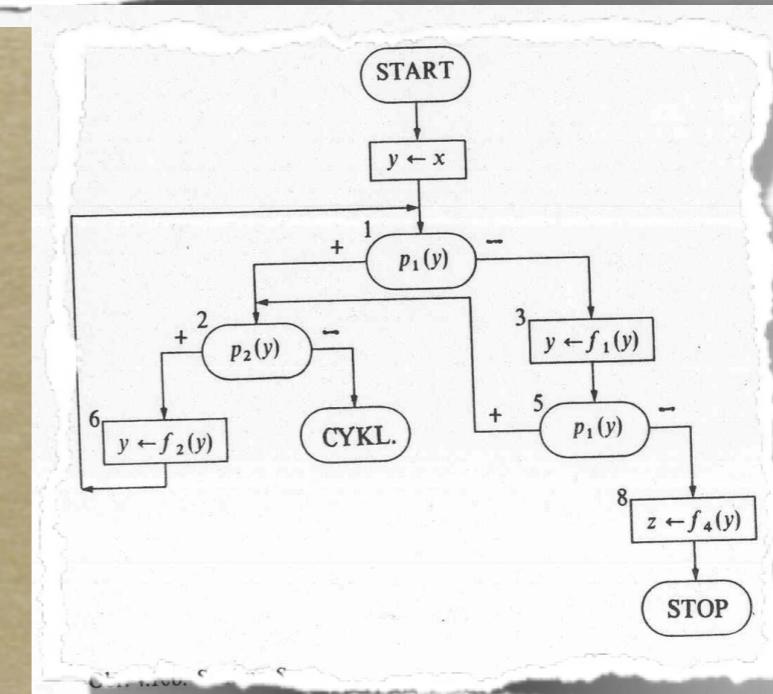
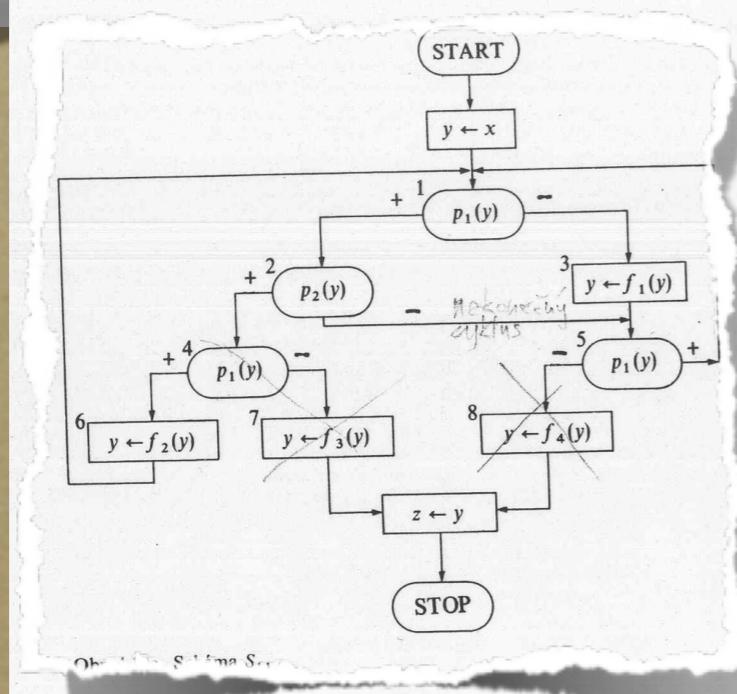
• Základní vlastnosti programových schémat, jakými jsou ukončení, divergence, ekvivalence či izomorfismus, závisejí na chování daného schématu ve všech interpretacích nad všemi možnými obory. Dokazování těchto vlastností by se samozřejmě značně usnadnilo, kdybychom se mohli omezit na vhodný speciální obor interpretace takový, že chování schémat v různých interpretacích nad tímto speciálním oborem dostatečně charakterizuje chování schémat ve všech interpretacích nad všemi možnými obory. Takový pevný obor interpretace je skutečně možné ke každému schématu najít; nazývá se Herbrandovo univerzum schématu S a označuje se H_S .

Herbrandovo univerzum H_S sestává ze všech výrazů definovaných rekurzivně

Volná schémata

Významnou třídu schémat tvoří tzv. volná schémata: jsou důležitá tím, že v mnoha případech je podstatně jednodušší dokázat, že jistou vlastnost má schéma, které je volné, než libovolné obecné schéma. Začneme příkladem.

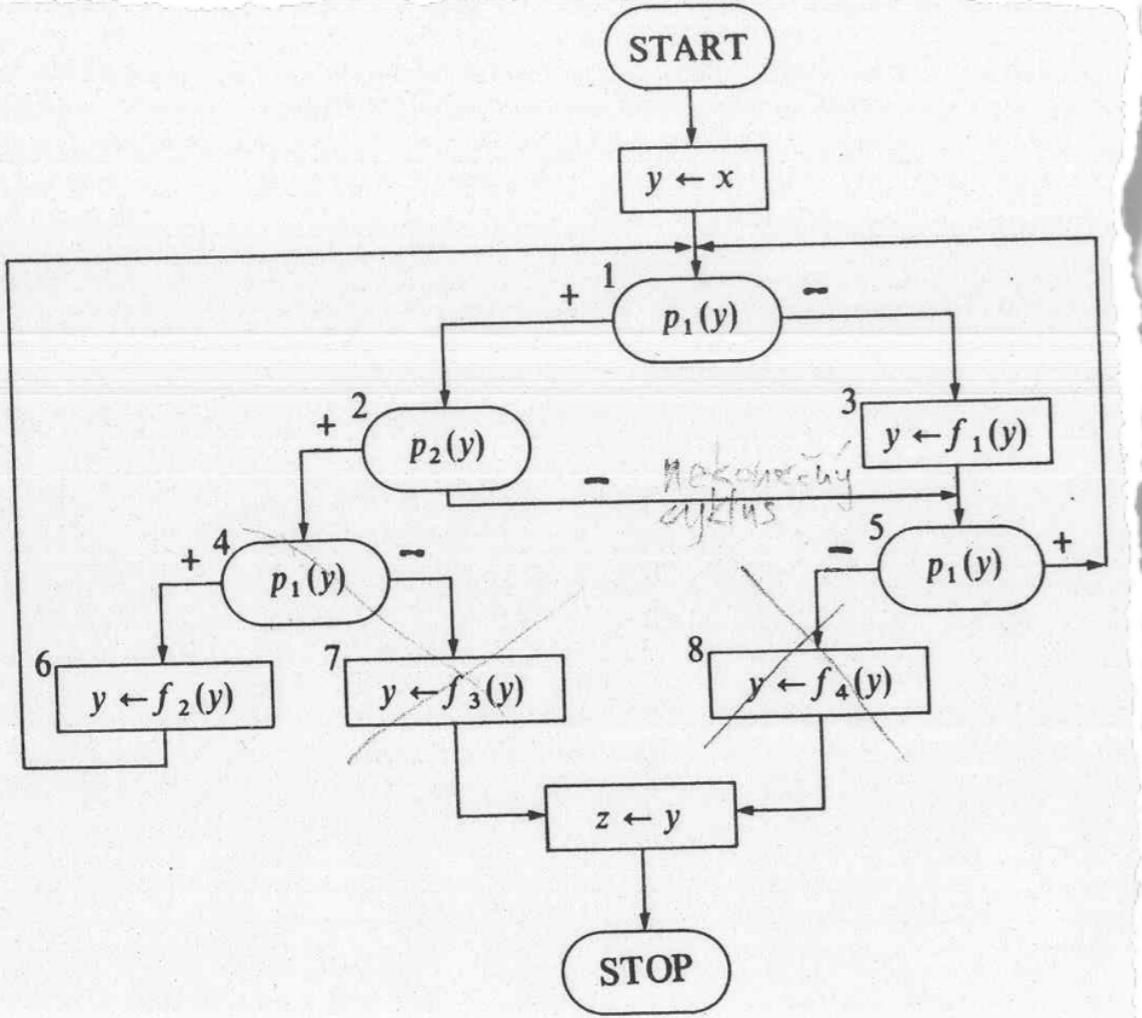
Uvažujme schéma S_{11A} z obr. 4.10a. Žádný výpočet podle tohoto schématu neobsahuje posloupnost příkazů ... 1, 2, 4, 7, ... ani posloupnost ... 1, 2, 5, 8, ... ani posloupnost ... 5, 1, 3, ...; žádný konečný výpočet neobsahuje posloupnost ... 1, 2, 5, 1. Odtud snadno nahlédneme, že schéma S_{11A} je ekvivalentní schématu S_{11B} z obr. 4.10b.



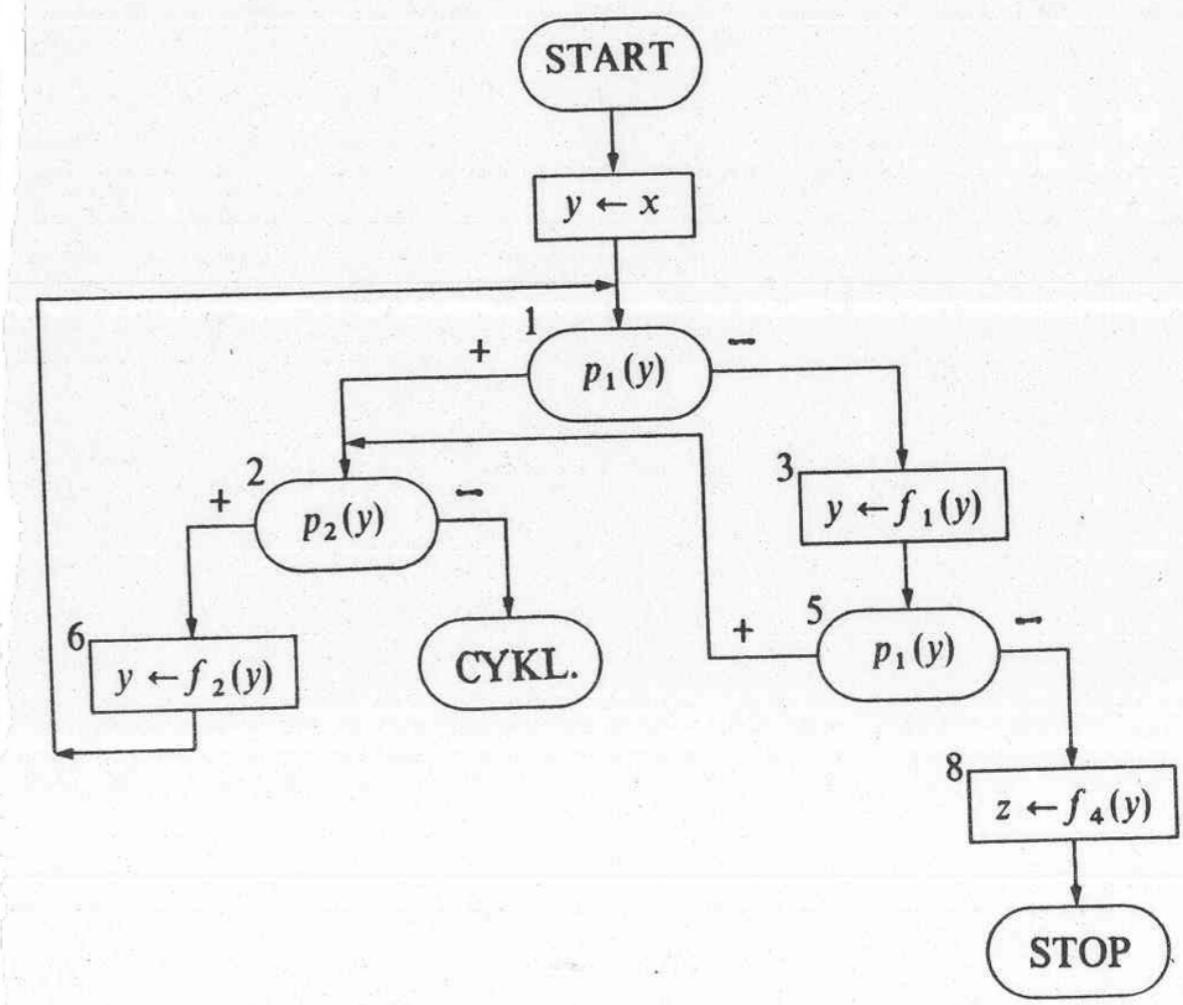
Volná schémata

Upravené 4.11

Uvažujme schéma S_{11A} z obr. 4.10a. Žádný výpočet podle tohoto schématu neobsahuje posloupnost příkazů ... 1, 2, 4, 7, ... ani posloupnost ... 1, 2, 5, 8, ... ani posloupnost ... 5, 1, 3, ...; žádný konečný výpočet neobsahuje posloupnost ... 1, 2, 5, 1. Odtud snadno nahlédneme, že schéma S_{11A} je ekvivalentní schématu S_{11B} z obr. 4.10b.



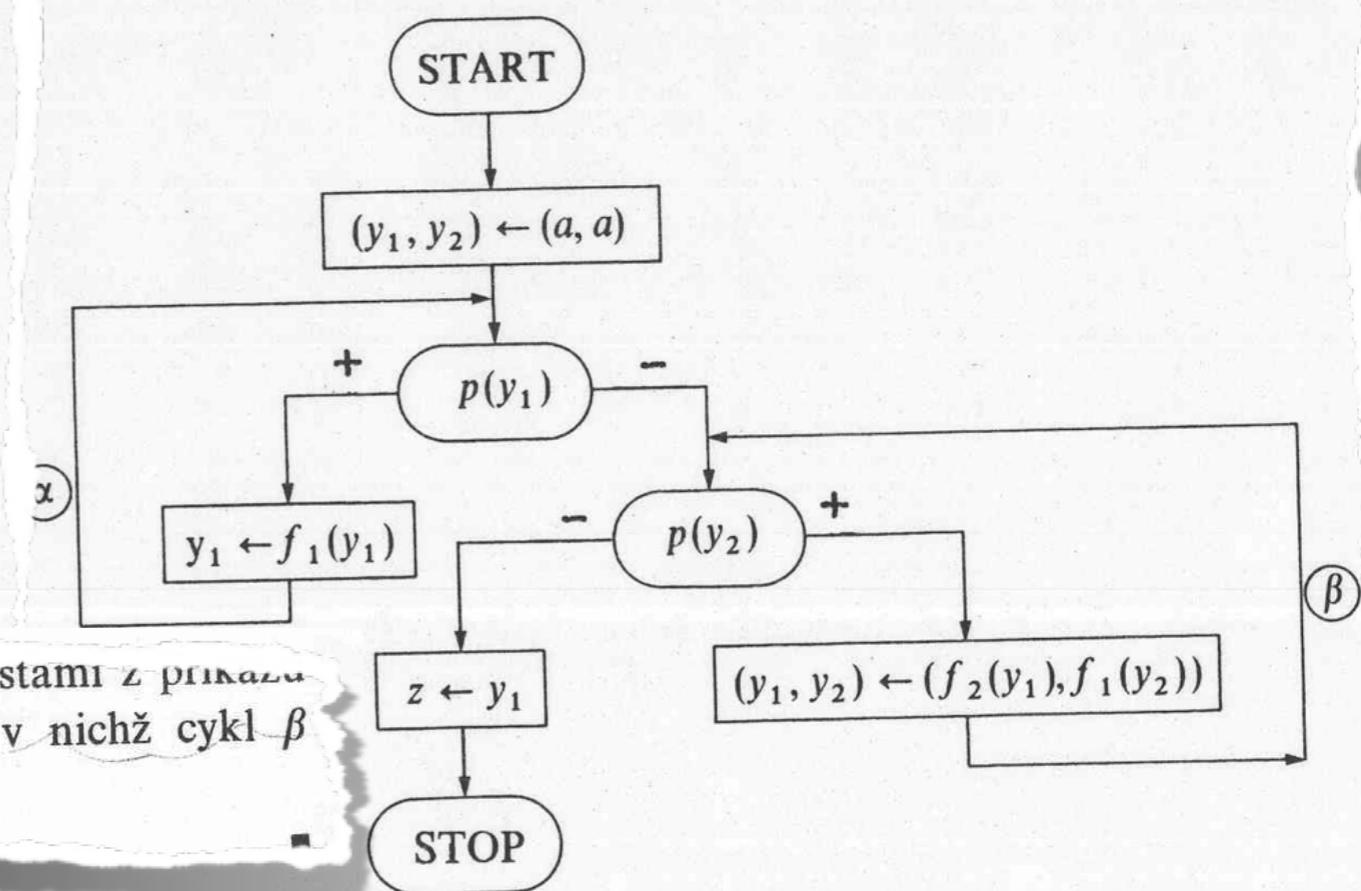
Obr. 4.10a. Schéma S_{11A}



Obr. 4.10b. Schéma S_{11B}

Volná schémata

Ke schématu S_{11A} existují „nemožné“ posloupnosti příkazů v tom smyslu, že žádný výpočet nemůže probíhat (v jisté své části) podle nich. O schématech s takovými nemožnými posloupnostmi řekneme, že nejsou volná. Ve schématu S_{11B} žádnou takovou nemožnou posloupnost neobjevíme: řekneme, že schéma je volné. Formálně řečeno, schéma S se nazývá volné, jestliže každá konečná cesta jeho vývojovým diagramem, začínající příkazem START, je počátečním úsekem nějakého výpočtu. Z této definice vyplývá, že u volného schématu i každá ne-konečná cesta jeho vývojovým diagramem, začínající příkazem START, popisuje jistý výpočet [viz cv. 4.6(b)].



Příklad 4.12

Schéma S_{12} z obr. 4.11 není volné: mezi všemi možnými cestami z příkazu **START** do příkazu **STOP** pouze ty popisují možné výpočty, v nichž cykl β je opakován tolirkát, kolikrát proběhl cykl α .

Obr. 4.11. Schéma S_{12} , které není volné

Volná schémata

Třída volných schémat má řadu zajímavých vlastností; např. platí:

s br 232 1. *Problém ukončení pro volná schémata je rozhodnutelný.*

Toto tvrzení plyně ze skutečnosti, že volné schéma nekončí, právě když buď obsahuje příkaz CYKLUJE nebo jeho vývojový diagram obsahuje cykl.

232 2. *Problém divergence pro volná schémata je rozhodnutelný.*

To plyně ze skutečnosti, že volné schéma nediverguje, právě když obsahuje příkaz zastavení STOP.

3. *Problém zdali dané schéma je volné není (ani) parciálně rozhodnutelný.*

Stromová schémata

Stromové schéma. T nad konečnou abecedou Σ_T je (event. nekonečný) vývojový diagram²) který má tvar stromu, tj. který je sestrojen z příkazů nad Σ_T tak, že

1. obsahuje právě jeden výchozí příkaz START;
2. ke každému příkazu existuje právě jedna cesta z příkazu START;
3. každý list stromu je buď příkaz zastavení STOP, nebo příkaz cyklování CYKLUJE.

Každé konečné stromové schéma, tj. stromové schéma s konečným počtem příkazů, je programové schéma ve smyslu dřívější definice.

Je zřejmé, že problémy ukončení, divergence, ekvivalence i izomorfismu pro konečná stromová schémata jsou rozhodnutelné.

Metodu, umožňující rozhodnout zda dvě kompatibilní konečná stromová schémata jsou ekvivalentní, ukážeme na příkladě.

Stromová schémata

[Příklad 4.14]

Uvažujme konečná stromová schémata S_{14A} a S_{14B} (obr. 4.14a, b); symboly e_1, e_2, e_3 zde zastupují po řadě výrazy „ $f(x_1, x_2)$ “, „ $g(x_2)$ “, „ $f(f(x_1, x_2), g(x_2))$ “. Výslednou hodnotu proměnné z schématu S_{14A} v libovolné Herbrandově interpretaci lze psát ve tvaru

if $p(e_2) = p(e_3) = \text{nepravda}$ then „ a “ else „ b “³)

Výsledné hodnoty proměnné z , ke kterým dojdeme při sledování jednotlivých cest ve schématu S_{14B} mohou být shrnuty do této tabulky:

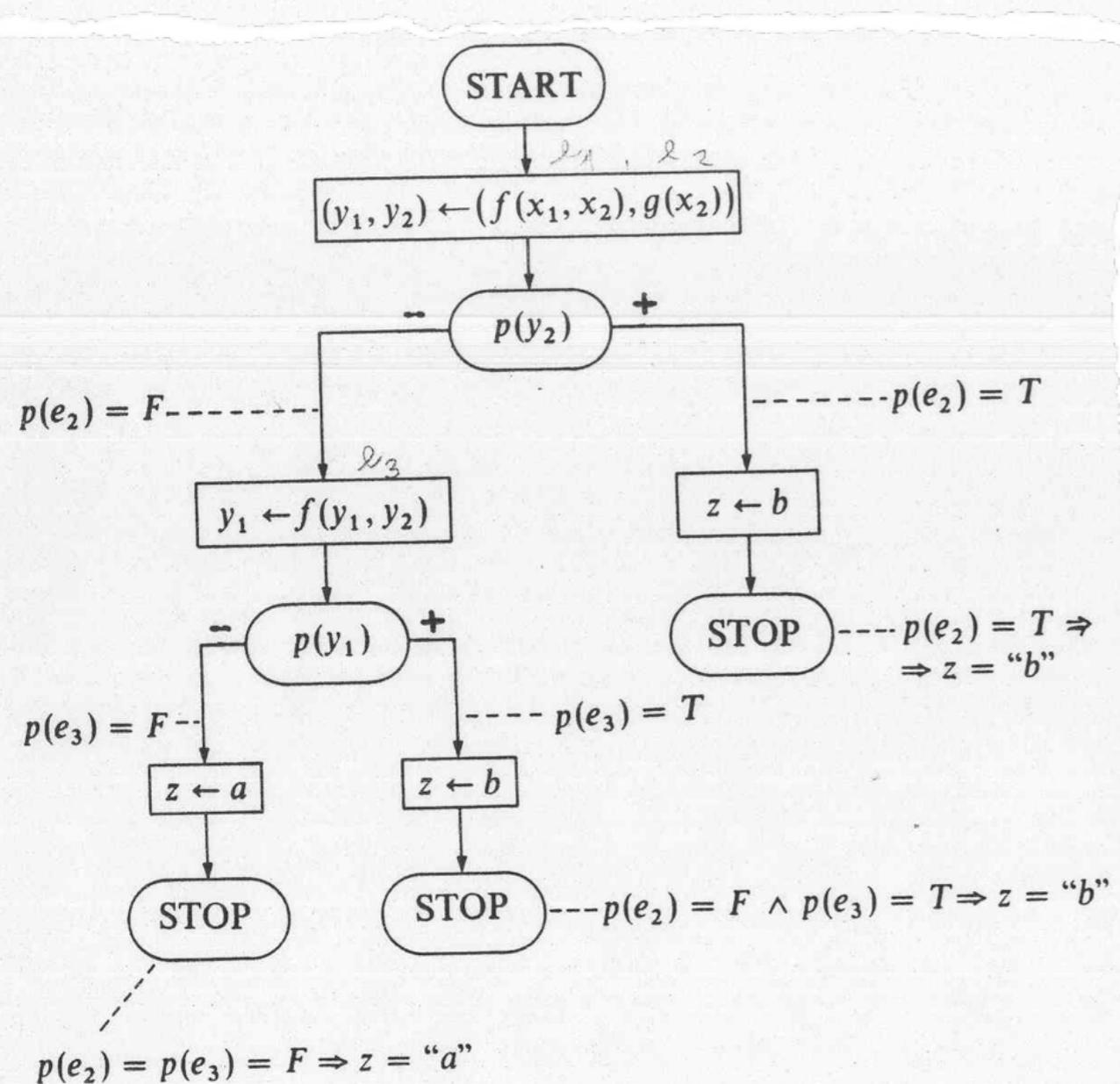
$p(e_1)$	$p(e_2)$	$p(e_3)$	z
pravda	pravda	pravda	„ b “
pravda	pravda	nepravda	„ b “
pravda	nepravda	pravda	„ b “
pravda	nepravda	nepravda	„ a “
nepravda	pravda	pravda	„ b “
nepravda	pravda	nepravda	„ b “
nepravda	nepravda	pravda	„ b “
nepravda	nepravda	nepravda	„ a “

Avšak závislosti zadané tabulkou lze charakterizovat rovněž výrazem

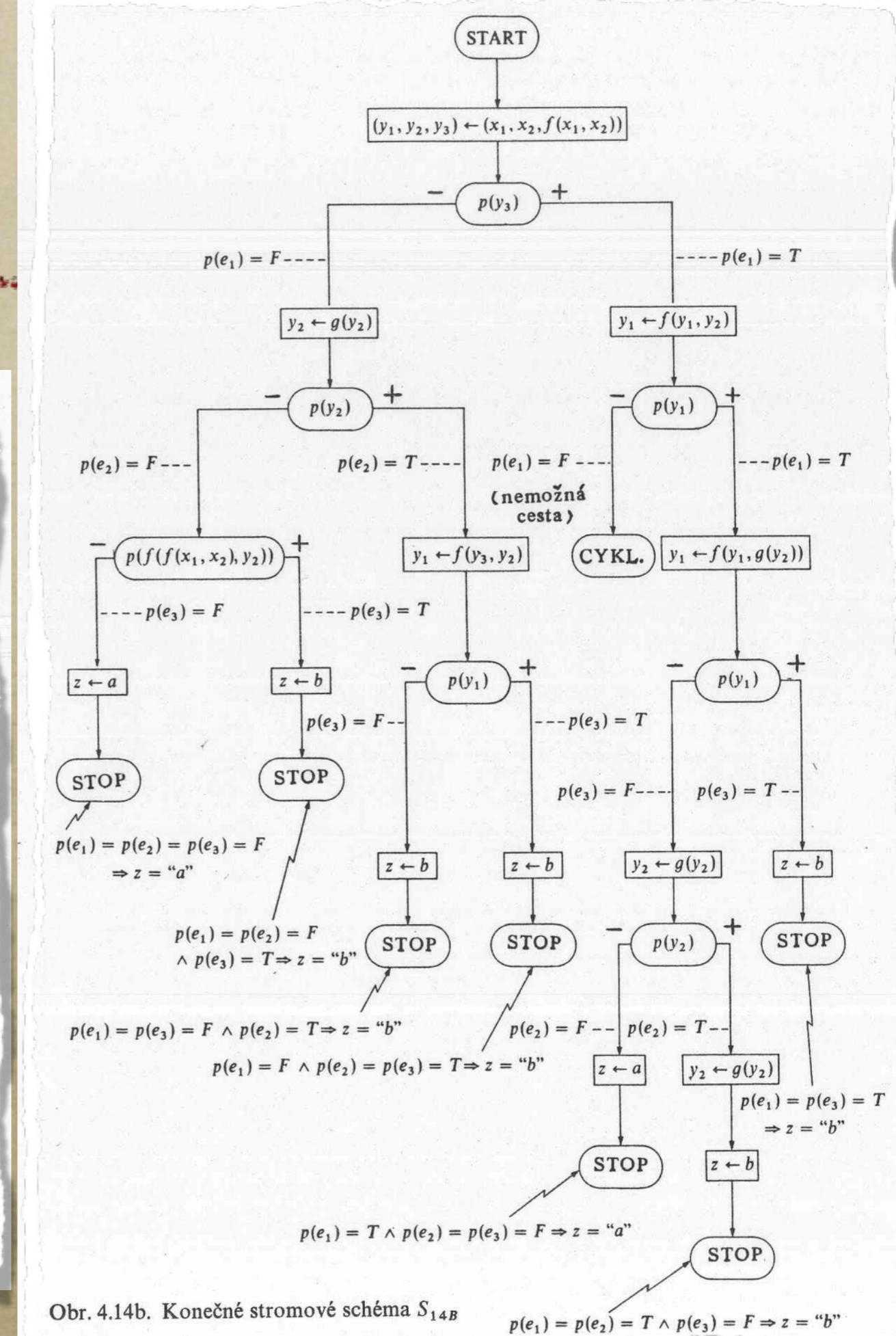
if $p(e_2) = p(e_3) = \text{nepravda}$ then „ a “ else „ b “.

Je tedy $h\langle S_{14A}, \mathcal{I}^*, \bar{x} \rangle \equiv h\langle S_{14B}, \mathcal{I}^*, \bar{x} \rangle$ pro všechny Herbrandovy interpretace \mathcal{I}^* a schémata S_{14A} a S_{14B} jsou ekvivalentní. (str. 233, 234)

Stromová schémata



Obr. 4.14a. Konečné stromové schéma S_{14A}



Obr. 4.14b. Konečné stromové schéma S_{14B}

$p(e_1) = p(e_2) = T \wedge p(e_3) = F \Rightarrow z = "b"$

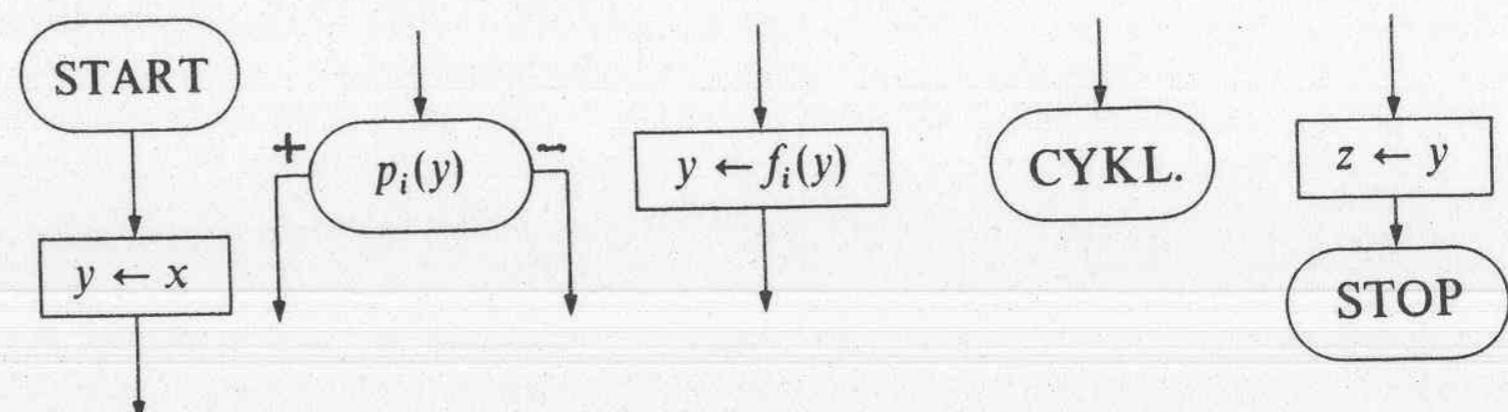
Janovova schémata

Zajímavou třídou programových schémat, pro kterou jsou rozhodnutelné všechny čtyři vyšetřované druhy problémů, jsou tzv. Janovova schémata. Jsou charakterizována především tím, že v nich vystupuje jediná programová proměnná y . Přesněji definujeme Janovovo schéma S jako programové schéma, u kterého:

1. Σ_S sestává z (konečně mnoha) unárních funkčních konstant f_i , (konečně mnoha) unárních predikátových konstant p_i , jediné vstupní proměnné x , jediné programové proměnné y a jediné výstupní proměnné z . (Všimněme si, že individuové konstanty nejsou povoleny¹).)

$$\Sigma_S = \{f_1, f_m, p_1, \dots, p_n, z\}$$

2. Všechny příkazy mají některý z těchto tvarů:



Pro Janovova schémata platí:

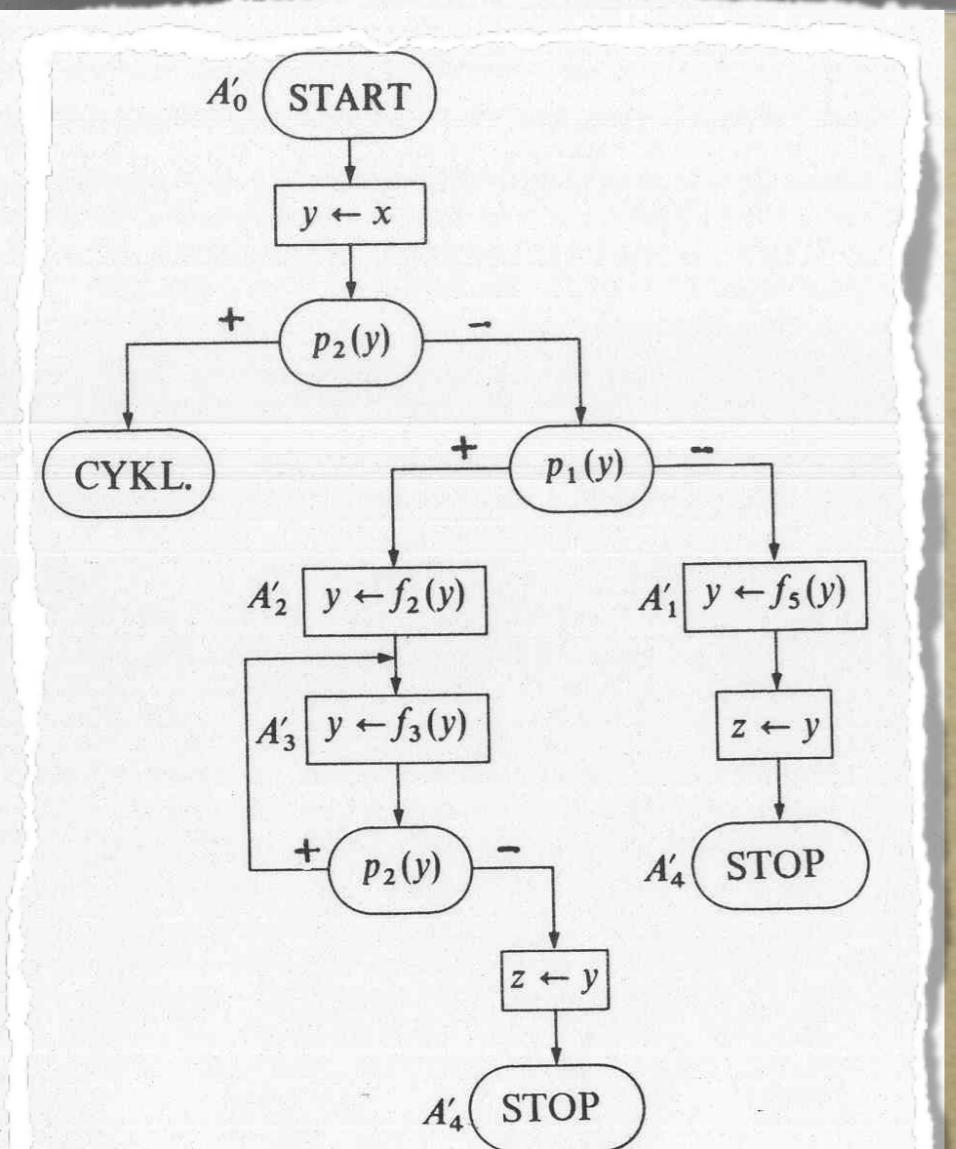
1. Je rozhodnutelné, zda dané Janovovo schéma je volné.

Toto tvrzení vyplývá ze skutečnosti, že dané Janovovo schéma není volné, právě když v něm existuje cesta obsahující dva stejné testovací příkazy, mezi nimiž není umístěn žádný příkaz přiřazovací¹.

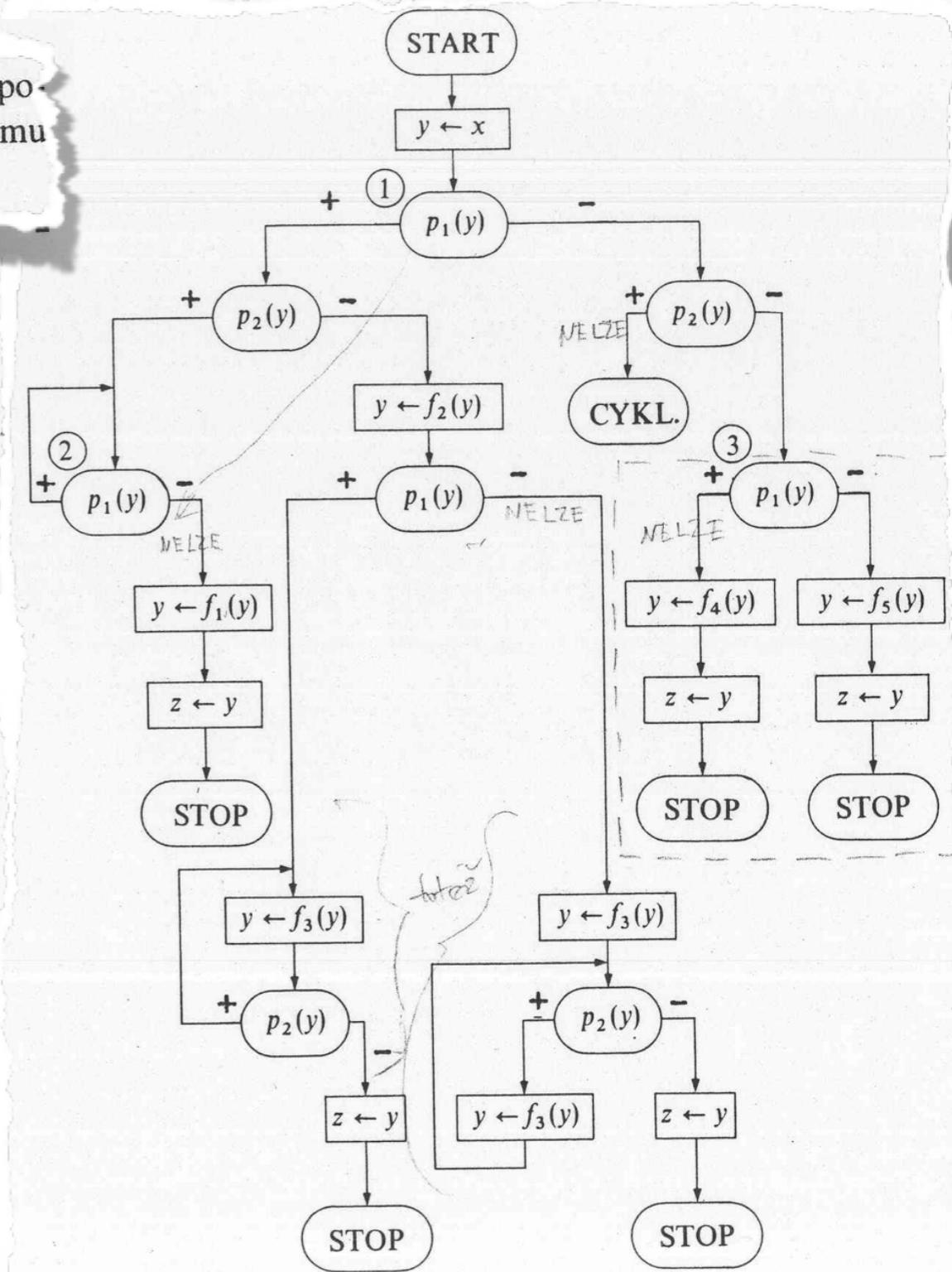
Janovova schémata

Příklad 4.17

NELZE
Schéma S_{17A} z obr. 4.17a je Janovovo schéma. Není volné vzhledem k posloupnosti testovacích příkazů z 1 do 2, z 1 do 3 a vzhledem k cyklu tvořenému samotným příkazem 2. Schéma S_{17B} z obr. 4.17b je volné Janovovo schéma.



Obr. 4.17b. Janovovo schéma S_{17B}



Obr. 4.17a. Janovovo schéma S_{17A}

Janovova schémata

Pro Janovova schémata platí:

1. Je rozhodnutelné, zda dané Janovovo schéma je volné.

Tento výsledek vyplývá ze skutečnosti, že dané Janovovo sc

2. Každé Janovovo schéma lze vhodnými úpravami převést na ekvivalentní volné Janovovo schéma.

[Tento důkaz této vlastnosti nenechované čtenáři [viz CV. 4.9(a)], zde

3. Problémy ukončení i divergence jsou pro Janovova schémata rozhodnultelné.

Jde o důsledek vlastnosti 2 a skutečnosti, že tyto problémy jsou rozhodnultelné pro volná schémata.

[Schéma je rozhodnutelný ([Janov 1960])

tehne pro volná schémata.

4. Problém ekvivalence pro Janovova schémata je rozhodnutelný ([Janov 1960], [Kaplan 1969]).

Předpokládejme, že je dano volné Janovovo schéma S s výchozím příkazem

Verifikace programů

(matematická teorie programu)

Idea

Úkolem této kapitoly je popsat metody verifikace programů pro číslicové počítače. Předpokládejme, že je dán takový program a spolu s ním i popis jeho předpokládaného chování, a to ve formě predikátu (který později nazveme výstupním predikátem), jenž udává požadovaný vztah mezi hodnotami programových proměnných v okamžiku ukončení výpočtu. Někdy je dán i vstupní predikát, který definuje omezení, jež musí splňovat vstupní hodnoty, pokud má mít prováděný výpočet zamýšlený smysl. Naším úkolem je dokázat, že program je správný vzhledem k takto zadaným predikátům; tj. že všechny výpočty, u nichž vstupní hodnoty splňují vstupní predikát mají tu vlastnost, že končí a že potom hodnoty proměnných splňují výstupní predikát. V této kapitole popíšeme, jak lze takové důkazy konstruovat.

Idea

- ◆ O algoritmu je možné tvrdit, že je správný, když lze dokázat, že je správný vzhledem k jeho specifikaci.
- ◆ Mezi správností (verifikací, dokazováním správnosti) algoritmů a programů není principiální rozdíl. Program je jedna z forem zápisu algoritmu.
- ◆ Verifikace programu je proces formálního dokazování toho, že počítačový program dělá přesně to, co je uvedeno ve specifikaci programu, která udává co má program realizovat.
- ◆ Verifikace programu se zabývá podáním formálního důkazu správnosti algoritmu

Idea

- ◆ Testováním programu lze pouze zjistit přítomnost chyby, nikoliv její nepřítomnost!

Idea

Murphyho zákony v IT

- ♦ Může-li něco havarovat, havaruje to. Nemůže-li to havarovat, havaruje to přesto.
- ♦ Může-li havarovat více věcí, havaruje ta, která napáchá největší škodu.
- ♦ Neexistuje tak velký nesmysl, aby ho některý uživatel nezadal

Základní pojmy - programy

Uvažujme nejprve velmi jednoduché programy zapsané ve tvaru vývojových diagramů.¹⁾ Rozlišujme tři druhy proměnných, seskupených do tří vektorů: (1) vstupní vektor $\bar{x} = (x_1, x_2, \dots, x_a)$ nabývá vždy daných vstupních hodnot a během výpočtu se proto nemění; (2) vektor programu $\bar{y} = (y_1, y_2, \dots, y_b)$, v němž jsou uchovávány mezivýsledky výpočtu; a (3) výstupní vektor $\bar{z} = (z_1, z_2, \dots, z_c)$, který obsahuje výstupní hodnoty v okamžiku skončení výpočtu. Těmto vektorům odpovídají po řadě tři typy (neprázdných) oborů: (1) vstupní obor $D_{\bar{x}}$, (2) obor programu $D_{\bar{y}}$ a (3) výstupní obor $D_{\bar{z}}$. Každý z oborů je kartézským součinem složek:

$$D_{\bar{x}} = D_{x_1} \times D_{x_2} \times \dots \times D_{x_a},$$

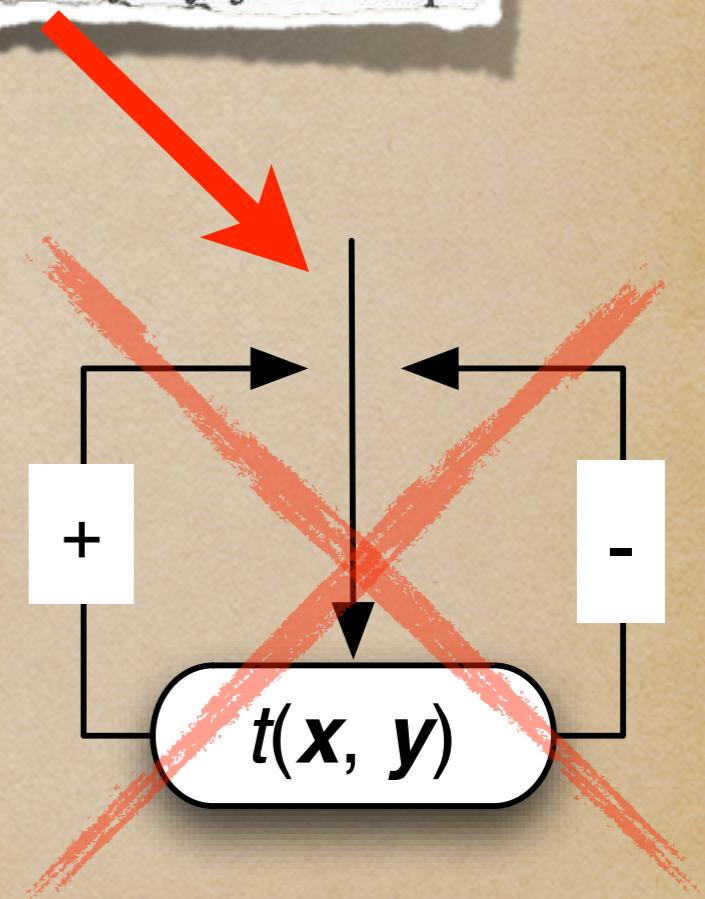
$$D_{\bar{y}} = D_{y_1} \times D_{y_2} \times \dots \times D_{y_b},$$

$$D_{\bar{z}} = D_{z_1} \times D_{z_2} \times \dots \times D_{z_c}.$$

Základní pojmy - programy

Programem²) zde budeme rozumět libovolný vývojový diagram sestavený z uvedených příkazů tak, že příkaz START je použit právě jednou a každý prirazovací a testovací příkaz leží na jisté cestě od příkazu START k některému příkazu STOP. Jinými slovy, programy nesmí obsahovat „mrtvé“ cykly jako např.

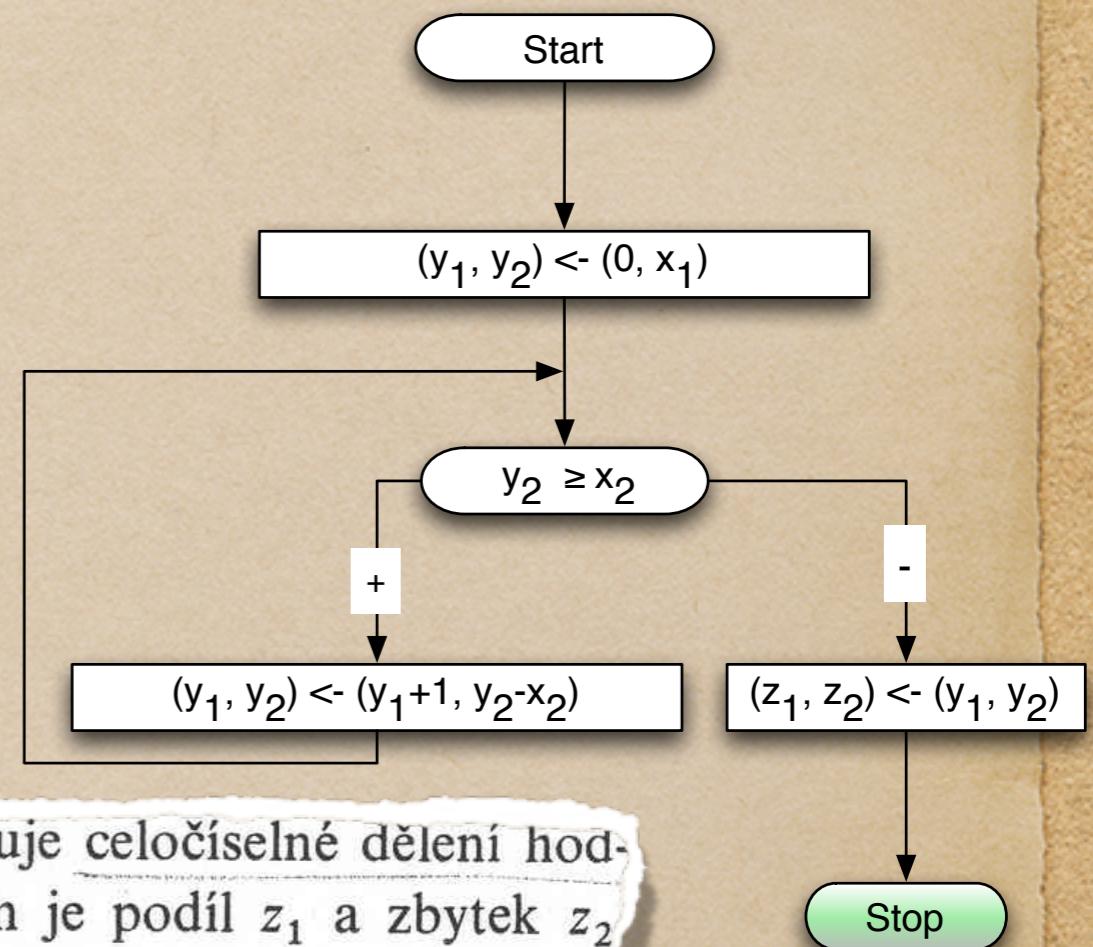
Je-li k danemu programu P dána také vstupní hodnota $\xi \in D_{\bar{x}}$ vstupního vektoru \bar{x} , program může být proveden. Běh programu (výpočet podle programu)¹ začíná vždy tím, že výchozí příkaz START inicializuje hodnotu vektoru \bar{y} hodnotou $f(\xi)$; dále pokračuje obvyklým způsobem, přecházeje od jednoho příkazu k následujícímu v souhlasu s orientovanými hranami diagramu. Kdykoliv se přejde k přirazovacímu příkazu, nahradí se hodnota vektoru \bar{y} hodnotou $g(\bar{x}, \bar{y})$ pro průběžný (momentální) hodnoty vektorů \bar{x} a \bar{y} . Kdykoliv se přejde k testovacímu příkazu další průběh sleduje větev „+“ nebo „-“ podle toho, zda průběžná hodnota predikátu $t(\bar{x}, \bar{y})$ je *pravda* nebo *nepravda*; hodnota \bar{y} zůstává přitom nezměněna. Jestliže program končí (tím, že přejde k příkazu STOP), výstupnímu vektoru \bar{z} je přiřazena průběžná hodnota $h(\bar{x}, \bar{y})$ – označme ji např. ζ ; v tomto případě řekneme, že je $P(\xi)$ definováno a položíme $P(\xi) = \zeta$. V opačném případě, jestliže běh programu nikdy nekončí, říkáme, že $P(\xi)$ není definováno. Jinými slovy, program P představuje parciální zobrazení $\bar{z} = P(\bar{x})$ množiny $D_{\bar{x}}$ do množiny $D_{\bar{z}}$.



Základní pojmy - programy

Příklad: celočíselné dělení

$$x_1/x_2$$



Jako příklad uvažujme program z obr. 3.1; realizuje celočíselné dělení hodnoty x_1 hodnotou x_2 , kde $x_1 \geq 0$ a $x_2 > 0$; výsledkem je podíl z_1 a zbytek z_2 $z_1 = \text{podíl}(x_1, x_2)$, $z_2 = \text{zbytek}(x_1, x_2)$. Zde $\bar{x} = (x_1, x_2)$, $\bar{y} = (y_1, y_2)$, $\bar{z} = (z_1, z_2)$ a $D_{\bar{x}} = D_{\bar{y}} = D_{\bar{z}} =$ množina všech uspořádaných dvojic celých čísel. Příkaz $(y_1, y_2) \leftarrow (0, x_1)$ má za následek nahradu hodnoty proměnné y_1 hodnotou 0 a hodnoty proměnné y_2 hodnotou proměnné x_1 ; podobně $(y_1, y_2) \leftarrow (y_1 + 1, y_2 - x_2)$ nahrazuje hodnotu proměnné y_1 hodnotou výrazu $y_1 + 1$ a hodnotu proměnné y_2 hodnotou výrazu $y_2 - x_2$. Obecně

$$(y_1, y_2, \dots, y_n) \leftarrow (g_1(\bar{x}, \bar{y}), g_2(\bar{x}, \bar{y}), \dots, g_n(\bar{x}, \bar{y}))$$

Základní pojmy - programy

~~O verifikaci programů můžeme hovořit, pokud jsou zadány tyto dva predikáty:~~

1. Totální predikát $\varphi(\bar{x})$ nad $D_{\bar{x}}$, nazývaný vstupní predikát; popisuje ty prvky z $D_{\bar{x}}$, které mohou být použity jako vstupní hodnoty programu. Jinak

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 ξ , pro něž je $\varphi(\xi)$ pravda.
2. Program P je parciálně korektní vzhledem k φ a ψ , jestliže pro každou hodnotu ξ , pro niž $\varphi(\xi)$ je pravda a pro niž P končí, je i $\psi(\xi, P(\xi))$ pravda.
3. Program P je totálně korektní vzhledem k φ a ψ , jestliže pro každou hodnotu ξ pro niž $\varphi(\xi)$ je pravda, program P končí a při tom $\psi(\xi, P(\xi))$ je pravda.

Základní pojmy - programy

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

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

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

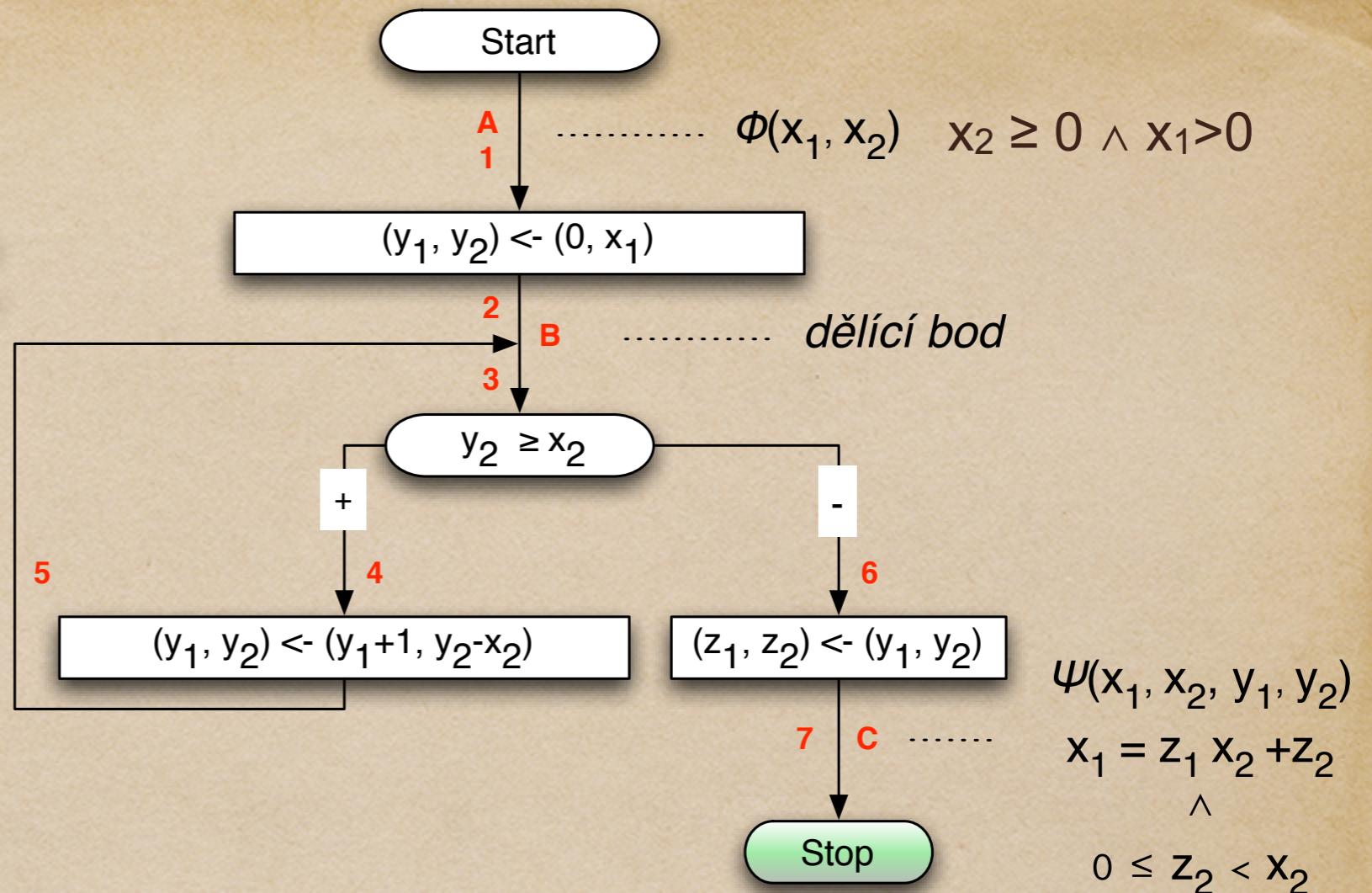
$$\psi(x_1, x_2, z_1, z_2): x_1 = z_1 x_2 + z_2 \wedge 0 \leq z_2 < x_2$$

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

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

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.

Parciální korektnost

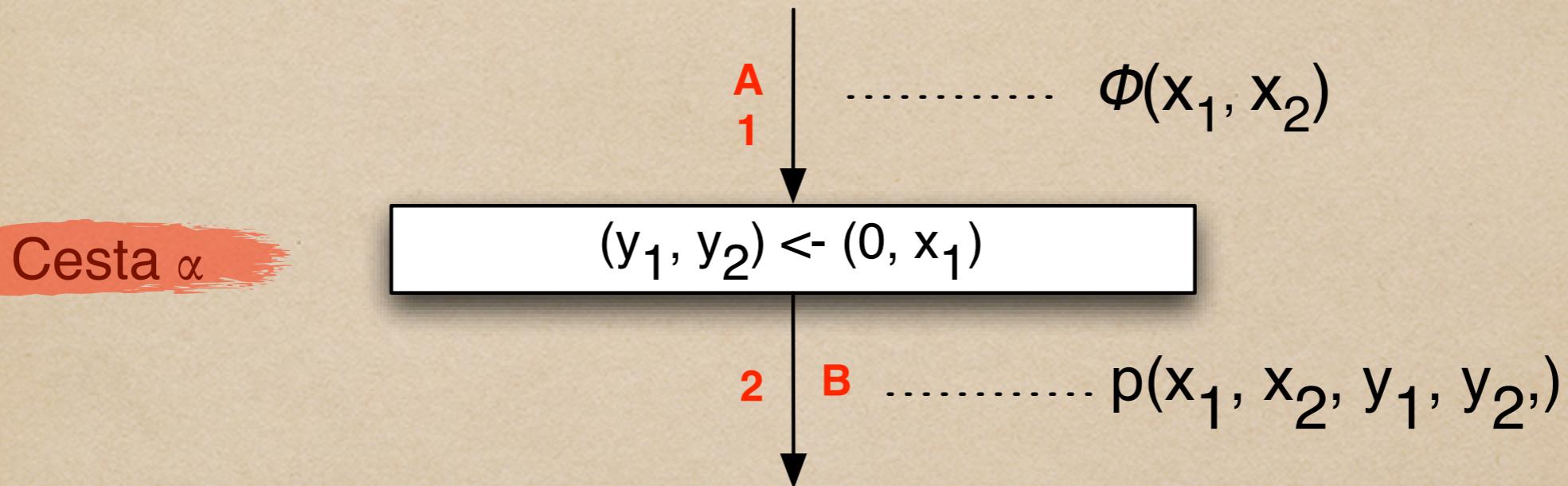
K dokázání parciální korektnosti je potřeba stanovit predikát $p(x_1, x_2, y_1, y_2)$, v tomto případě

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

Po získání predikátu p se program rozpadá na tři cesty, z nichž každá 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 cesty⁴⁾ 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.

Parciální korektnost



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⁾

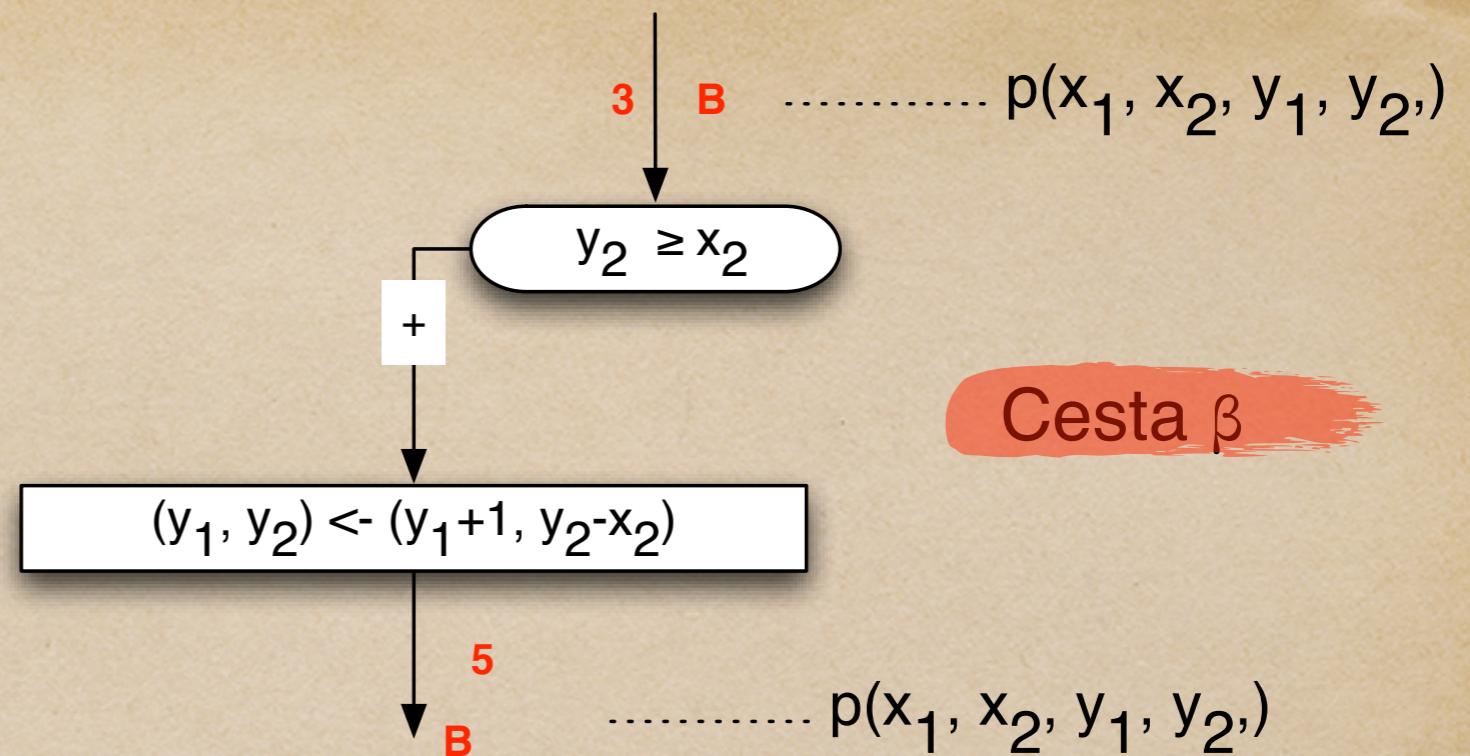
Verifikační podmínka cesty tedy je

$$\underline{\varphi(x_1, x_2)} \supseteq p(x_1, x_2, 0, x_1),$$

tj.

$$[x_1 \geq 0 \wedge x_2 \geq 0] \supseteq [x_1 = 0, x_2 + x_1 \wedge x_1 \geq 0].$$

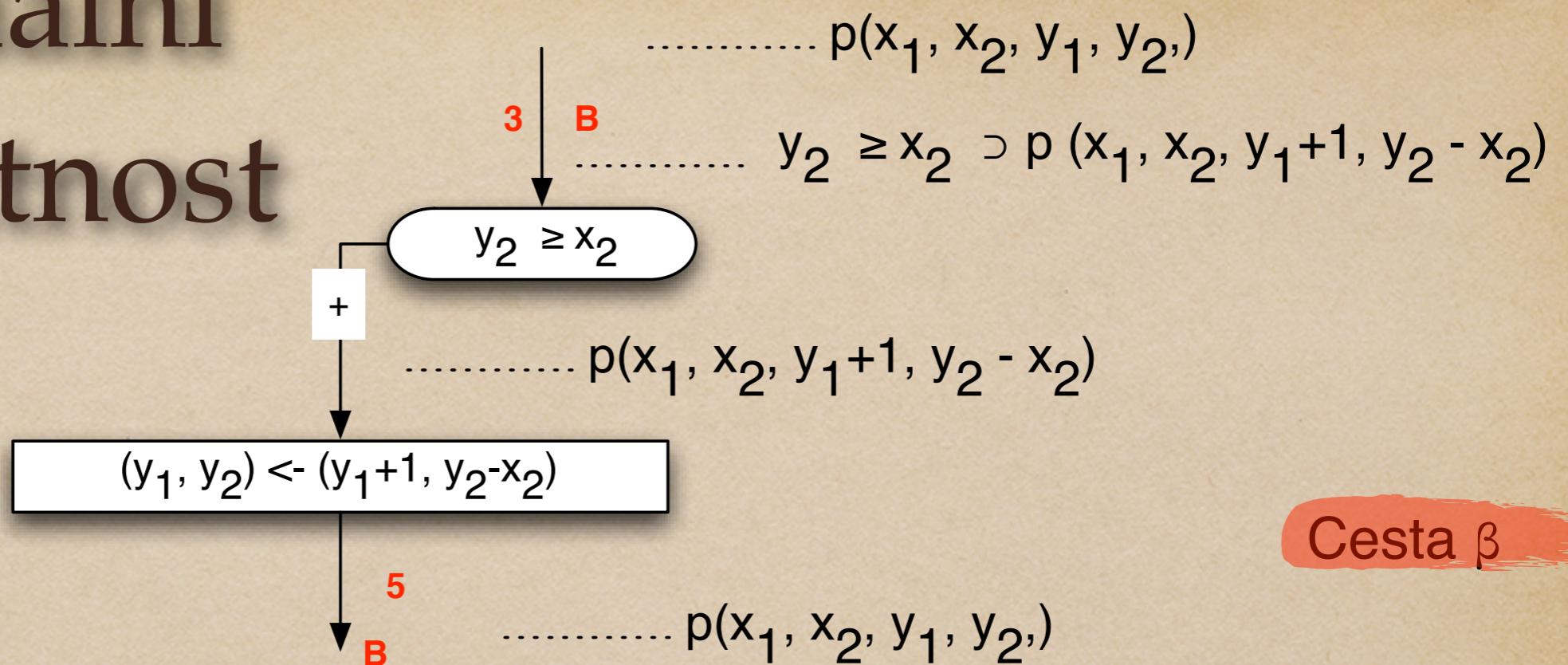
Parciální korektnost



Cesta β Při sestrojování verifikační podmínky cesty β musíme vzít v úvahu testovací příkaz; při sledování cesty β tento příkaz vždy shledá, že $y_2 \geq x_2$, 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 $y_2 \geq x_2$. 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 $y_2 \geq x_2$ – 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 $y_2 \geq x$, $\exists p(x_1, x_2, y_1 + 1, y_2 - x_2)$. Úplnou analýzou cesty β tedy dostáváme:

Parciální korektnost



Cesta β

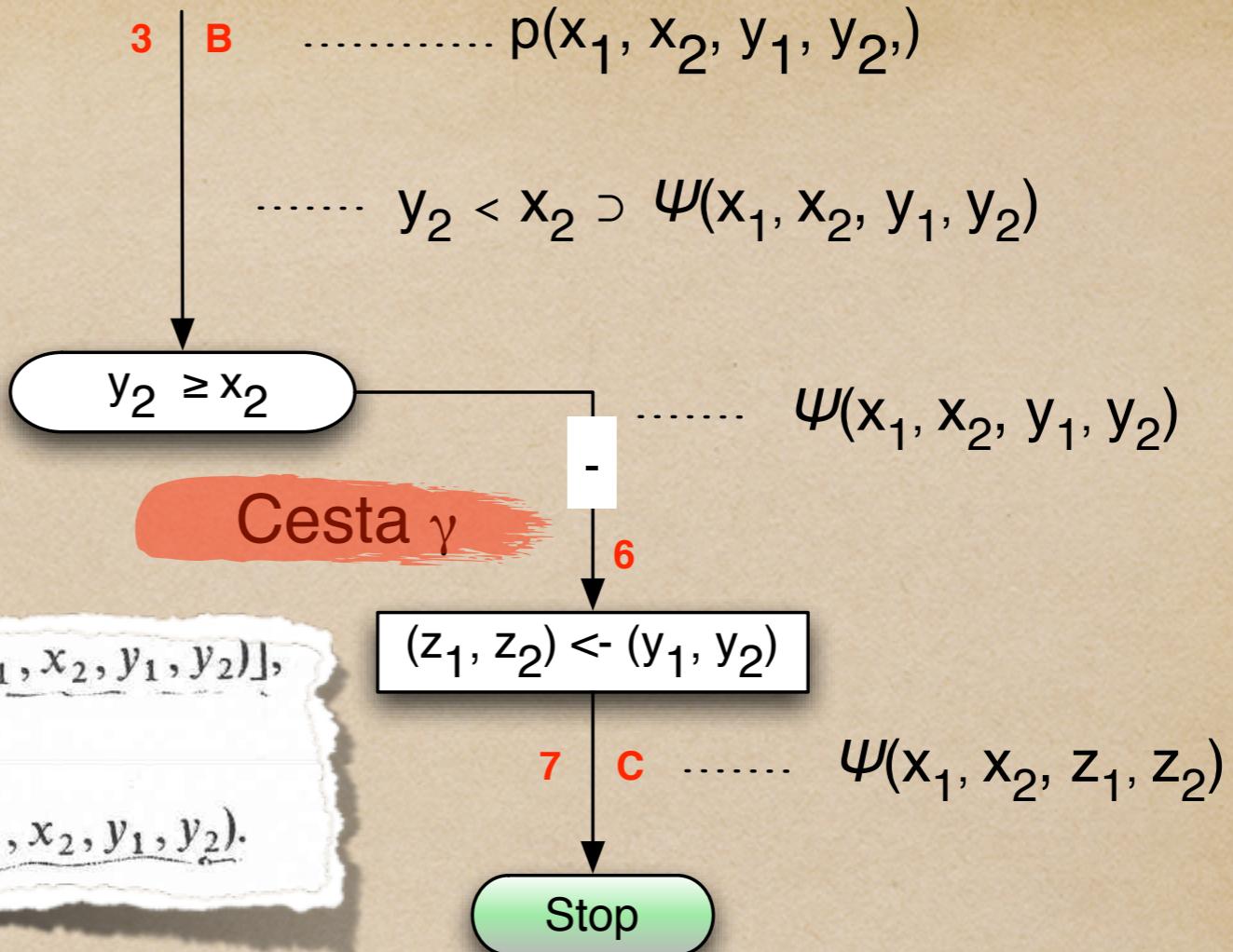
Celkem je tedy třeba dokázat verifikační podmínu

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



$$p(x_1, x_2, y_1, y_2) \wedge y_2 \geq x_2 \supset p(x_1, x_2, y_1 + 1, y_2 - x_2)$$

Parciální korektnost



$p(x_1, x_2, y_1, y_2) \supset [y_2 \geq x_2 \supset \psi(x_1, x_2, y_1, y_2)],$
 neboli
 $(p(x_1, x_2, y_1, y_2) \wedge y_2 < x_2) \supset \psi(x_1, x_2, y_1, y_2).$

Sestavili jsme tedy celkem tyto tři verifikační podmínky

$$\begin{aligned} \varphi(x_1, x_2) &\supset p(x_1, x_2, 0, x_1), \\ (p(x_1, x_2, y_1, y_2) \wedge y_2 \geq x_2) &\supset p(x_1, x_2, y_1 + 1, y_2 - x_2), \\ (p(x_1, x_2, y_1, y_2) \wedge y_2 < x_2) &\supset \psi(x_1, x_2, y_1, y_2), \end{aligned}$$

kde

$$\begin{aligned} \varphi(x_1, x_2) &\quad \text{je } x_1 \geq 0 \wedge x_2 \geq 0, \\ p(x_1, x_2, y_1, y_2) &\quad \text{je } x_1 = y_1 x_2 + y_2 \wedge y_2 \geq 0, \\ \psi(x_1, x_2, z_1, z_2) &\quad \text{je } x_1 = z_1 x_2 + z_2 \wedge 0 \leq z_2 < x_2. \end{aligned}$$