Pavol Jozef Šafárik University in Košice
Faculty of Science

Verifikácia programov
Hoareova metóda

Gabriela Andrejková

Úvod

V metóde induktívnych podmienok, o ktorej sme hovorili pri Floydovej metóde dokazovania správnosti programu, bol využívaný deduktívny systém predikátového počtu 1. rádu a verifikačné podmienky boli skonštruované k cestám medzi dvoma deliacimi bodmi programu.

C. A. R. Hoare

použil ako východisko na dokazovanie správnosti programov zapísaných v ľubovoľnom programovacom jazyku typu algol60, pascal, a pod., **metódu induktívnych podmienok**, ale vypracoval axiomatický prístup k definícii sémantiky príkazov.

Vychádzal z toho, že k tomu, aby proces dokazovania správnosti programu mohol byť sformalizovaný, je potrebné mať vhodný systém, v ktorom budú vyjadrované a odvodzované vlastnosti príkazov, resp. programov.

Induktívny výraz

Hoareov axiomatický prístup je najčastejšie používaným spôsobom na vyjadrenie sémantiky príkazov.

- Axiómy tohto systému definujú sémantiku najjednoduchších príkazov jazyka a
- odvodzovacie pravidlá umožňujú vyjadriť sémantiku vhodne vytvorenej postupnosti príkazov pomocou sémantiky jednoduchých príkazov, ktoré túto postupnosť vytvárajú.

Induktívny výraz

Na vyjadrenie sémantiky Hoare zaviedol tzv. **induktívne výrazy** (tiež ich budeme nazývať špecifikačné výrazy), ktoré sú tvaru

$$\{P\}\,\Pi\,\{Q\} \tag{1}$$

kde P, Q sú formuly predikátového počtu 1. rádu s rovnosťou a Π je jednoduchý príkaz alebo vhodne vytvorená postupnosť príkazov vo zvolenom programovacom jazyku.

Induktívny výraz

Tento induktívny výraz má nasledujúci význam pri dokazovaní čiastočnej správnosti programu (vychádzame z toho, že výpočet podľa príkazu alebo postupnosti príkazov skončí):

Vždy, keď formula P je splnená bezprostredne pred vykonaním Π a Π končí, potom formula Q je splnená po vykonaní Π.

Induktívny výraz

Dokázať, že program Π je čiastočne správny vzhľadom na vstupný predikát P a výstupný predikát Q teda znamená odvodiť induktívny výraz

$$\{P\}\,\Pi\,\{Q\}.$$

Hoareov axiomatický systém okrem axióm obsahuje tiež odvodzovacie pravidlá, ktorých tvar závisí od programovacieho jazyka, preto najprv popíšeme syntax jazyka, ktorý nazveme **JO** a ktorý bude obsahovať typické príkazy používané v známych programovacích jazykoch typu pascal, apod

Syntax programovacieho jazyka JO

V každom programovacom jazyku má premenná určený svoj dátový typ, t.j. množinu hodnôt, ktoré môže nadobúdať a operácie na tejto množine. Budeme predpokladať prácu s nasledujúcimi typmi dát:

a) integer, t.j. celé čísla,

b) Boolean, t.j. logické hodnoty,

c) real, t.j. reálne čísla,

d) char, t.j. množina znakov,

s operáciami definovanými rovnako ako v bežných programovacích jazykoch.
Premenné budeme označovať identifikátorom, čo je postupnosť písmen a číslic, začínajúca písmenom. Premenná môže byť jednoduchá alebo štruktúrovaná. Zo štruktúrovaných premenných budeme uvažovať polia.

Výrazy jazyka JO

Faktor je identifikátor premennej alebo konštanta bez znamienka alebo volanie funkcie alebo výraz tvaru $\neg p$, kde p je faktor. Napríklad, z, 25, cos(x), $\neg r$.

Term je vytváraný z faktorov použitím multiplikatívnych operátorov $*, /, mod, div, \wedge$ (oprerátor div vyjadruje celočíselný podiel dvoch celých čísel, mod vyjadruje známu funkciu modulo). Napríklad: $25 * z$, $7 * cos(x)/6$, $p \wedge q$.

Jednoduchý výraz je vytvorený z termov pomocou aditívnych operátorov $+, -, \vee$ a môže byť tvaru S$\bigoplus$T, +T, -T, kde S je jednoduchý výraz, T je term a $\bigoplus$ je jeden z aditívnych operátorov. Napríklad z+25, (p $\vee$ q) $\wedge$ r.

Výrazom je jednoduchý výraz, alebo výraz je vytvorený pomocou dvoch jednoduchých výrazov $S1, S2$ v tvare $S1\ R\ S2$, kde R je jeden z nasledujúcich relačných operátorov: $<, >, =, \neq, \geq, \leq$. Napríklad: z, $x = 1.5$, $p \leq$q.

Okrem uvedeného delenia operátorov je možné operátory (okrem relačných operátorov) rozdeliť na logické $\wedge$, $\vee$, $\neg$ a aritmetické $+, -, *, /, div, mod$. Budeme predpokladať, že výrazy sú vytvárané rozumným spôsobom.

Budeme tiež používať nasledujúce funkcie:

- $sin(x)$, $cos(x)$ - známe goniometrické funkcie, kde x je aritmetický výraz,
- $arctg(x)$,
- $abs(x)$ absolútna hodnota, kde x je aritmetický výraz,
- $exp(x)$, e^x, kde x je aritmetický výraz,
- $odd(i)$ nadobúda hodnotu $true$, ak hodnota celočíselného výrazu i je nepárne číslo, inak $false$.

Na výraz sa môžeme pozerať ako na zloženú operáciu. Ak premenným, ktoré sa vo výraze vyskytli, bola priradená hodnota, potom tento výraz môže byť vyhodnotený, t.j. operátory, ktoré sa vyskytujú vo výraze môžu byť aplikované na svoje operandy. Výrazom je teda určená hodnota. Aplikácia operátorov závisí od ich priority.

Priorita aplikácie operátorov najvyššiu prioritu má $\neg$, nasledujú multiplikatívne, potom aditívne a nakoniec relačné operátory. Okrúhle zátvorky $(,)$ môžu poradie vykonávania sa operácií zmeniť.

Výrazy budeme deliť na logické a aritmetické podľa typu hodnoty, ktorú nadobúdajú. Logické výrazy nadobúdajú hodnoty typu Boolean a aritmetické nadobúdajú číselné hodnoty.

Podľa typu hodnoty, ktorú nadobúdajú, je možné rozdeliť tiež faktory, termy a jednoduché výrazy a za operandy voliť operandy s hodnotami požadovaných typov.

Príkazy jazyka JO

Najprv uvedieme syntax používaných príkazov s neformálnym vyjadrením sémantiky. Príkazy sú volené tak, aby pomocou nich bolo možné konštruovať štrutúrované programy.

Priraďovací príkaz

$$x := e,$$

kde x je premenná a e je výraz. Hodnota výrazu e je priradená premennej x.

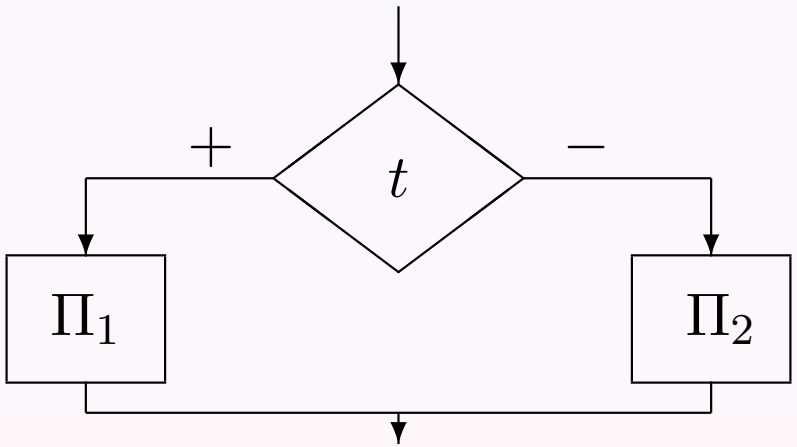
Podmienkové príkazy

1. if t then Π_1 else Π_2 fi;
2. if t then Π_1 fi,

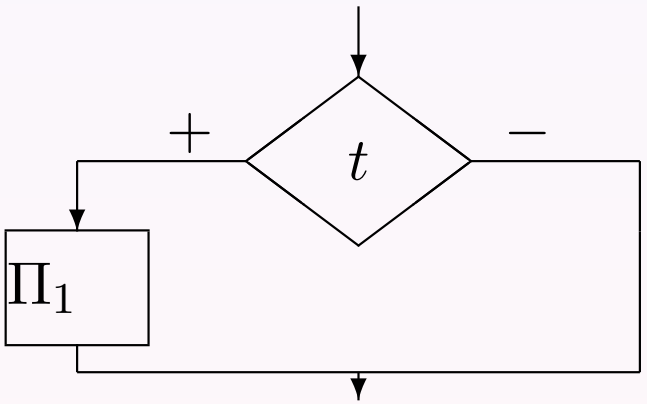
kde t je logická formula(bez kvantifikátorov) a Π_1, Π_2 sú príkazy typu a)-d). Význam tohto príkazu je nasledovný (formálna sémantika bude popísaná v ďalšom):

Ak t nadobudne hodnotu $true$ vykoná sa príkaz Π_1, ak t nadobudne hodnotu $false$ v prvom prípade sa vykoná príkaz Π_2, v druhom prípade sa vykoná prázdny príkaz. Vykonanie sa prázdneho príkazu znamená, že obsah žiadnej premennej sa nezmení.

1. if t then Π_1 else Π_2 fi;



2. if t then Π_1 fi;

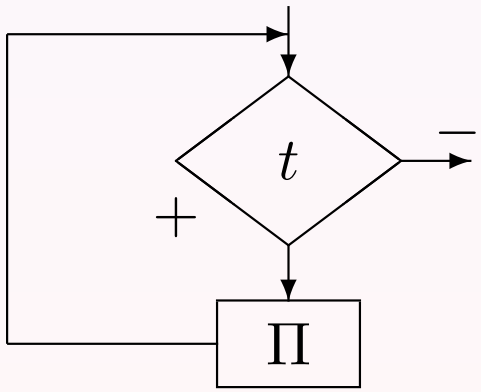


Príkazy cyklu

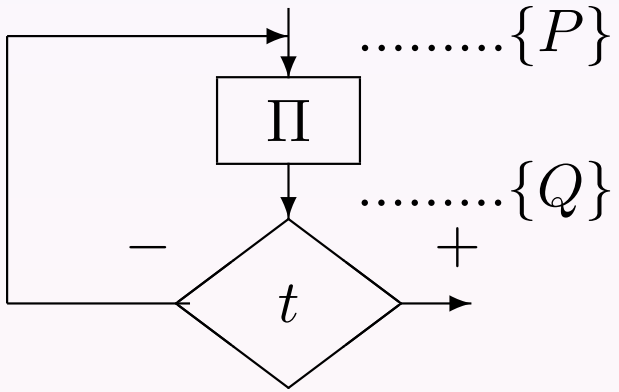
1. while t do Π od;
2. repeat Π until t,

kde t logická formula (bez kvantifikátorov) a Π je príkaz typu a)-d).
Pomocou jazyka vývojových diagramov je možné 1. príkaz cyklu vyjadriť takto:

1. while t do Π od;



2. repeat Π until t,



Kompozícia príkazov

je vyjadrená $\Pi_1; \Pi_2$, kde Π_1, Π_2 sú príkazy tvaru a)-d) a znamená, že po vykonaní príkazu Π_1 sa vykoná príkaz Π_2.

Štruktúrovaný program (v ďalšom len program) je kompozícia príkazov $\Pi_1, \Pi_2, \ldots, \Pi_n; n \geq 1$, kde Π_i, $1 \leq i \leq n$ sú príkazy tvaru a)-d).

Program pre výpočet najväčšieho spoločného deliteľa čísel $a,b \in N^+$

Program NSD;
y_1:=a; y_2:=b;
if $a < b$ then y_3:=a else y_3:=b fi;
while $\neg((y_1 \bmod y_3 = 0) \wedge (y_2 \bmod y_3 = 0))$ do
$y_3 := y_3 - 1$
od;
$z := y_3$; $\square$

Dĺžka najdlhšej spoločnej podpostupnosti

```
Program Nsdp;JO
i:=0;
while i ≤ n do j:=0;
          while j ≤ m  do    B[i,j]:=0; j:=j+1;
          od;
          i:=i+1;
od;
i:=1;
while i ≤ n do  j:=1;
          while j ≤ m do
                if X[1,i]=X[2,j] then B[i,j]:=B[i,j] + 1
                else if B[i,j − 1]>B[i − 1,j] then B[i,j]:=B[i,j − 1]
                     else B[i,j]:=B[i − 1,j]
                     fi;
                fi;
                j:=j+1;
          od;
          i:=i+1
od;
z:=B[n,m];    □
```

Program, ktorý zisťuje, či číslo x je dokonalé, alebo nie je.

Program DokonaleCislo;
y_2:=1; $y_1 := x\, div\, 2$;
$y_3 := 2$; $y_4 := false$;
repeat
if x mod y_3=0 then $y_2 := y_2 + y_3$;
$y_3 := y_3$+1;
until $y_3 > y_1$;
if y_2=x then y_4:=true;
$z := y_4$; $\square$

Axiómy a odvodzovacie pravidlá HAS

Odvodzovacie pravidlá umožňujú odvodiť vlastnosti programov a majú nasledujúci tvar

$$\frac{P_1, P_2, \ldots, P_n}{P}, n \geq 1,$$

kde $P_1, P_2, \ldots, P_n, P$ sú tvrdenia.
Pravidlo vyššie uvedeného tvaru má nasledujúcu interpretáciu: "ak platia tvrdenia $P_1, P_2, \ldots, P_n$, potom platí tvrdenie P".
$P_1, P_2, \ldots, P_n$ budeme tiež nazývať **predpoklady**, P je **dôsledok**.

P_i, $1 \leq i \leq n$, je induktívny výraz alebo axióma priradenia alebo logický výraz, ktorého platnosť je dokázaná vopred alebo v priebehu vytvárania dôkazu.
Teda, ak v priebehu odvodzovania sa ukázala platnosť $P_1, P_2, \ldots, P_n$, potom P je možné považovať za ďalší člen dôkazu.

Axiómy

A1. **Pre prázdny príkaz**

$$\frac{R \to S}{\{R\}\{S\}}$$

A2. **Pre priraďovací príkaz**

$$\{R[e\sharp y]\}\, y := e\, \{R\},$$

ktorý vyjadruje toto: ak R je pravdivá predtým, než výraz e bol nahradený premennou y, potom R je tiež pravdivá, keď do y bola priradená nová hodnota, určená výrazom e.
Výraz $R[e\sharp y]$ znamená, že každý voľný výskyt premennej y v $\{R\}$ nahradil výraz e.

A2. **Pre priraďovací príkaz** Napríklad,

$$\{x + z = 10\}\, y := x + z\, \{y = 10\},$$

alebo
$\{m \leq (m + n)\, div\, 2 \leq n\}\, y := (m + n)\, div\, 2\, \{m \leq y \leq n\}.$

P1. Pravidlá pre podmienkové príkazy

1.

$$\frac{\{P \wedge B\}\,\Pi_1\,\{Q\}, \{P \wedge \neg B\}\,\Pi_2\,\{Q\}}{\{P\}\ \text{ if } B \text{ then } \Pi_1 \text{ else } \Pi_2 \text{ fi } \{Q\}}$$

2.

$$\frac{\{P \wedge B\}\Pi_1\{Q\}, P \wedge \neg B \rightarrow Q}{\{P\} \text{ if } B \text{ then } \Pi_1 \text{ fi } \{Q\}}$$

P2. Pravidlá konsekvencie

1.

$$\frac{R \to S, \{S\}\Pi\{Q\}}{\{R\}\,\Pi\,\{Q\}}$$

2.

$$\frac{\{P\}\,\Pi\,\{Q\}, Q \to R}{\{P\}\Pi\{R\}}$$

P2. Pravidlá konsekvencie

Napríklad,

$$\frac{\{a > 0\}\, b := a\, \{b > 0\}, b > 0 \to b \geq 0}{\{a > 0\}\, b := a\, \{b \geq 0\}}$$

$$\frac{\{a > 0\}\, b := a\, \{b \geq 0\}, \{a \leq 0\}\, b := -a\, \{b \geq 0\}}{\{true\}\ \text{if } a > 0 \text{ then } b := a \text{ else } b := -a \{b \geq 0\}}$$

Pravidlá konsekvencie sú odvoditeľné z A1 a P4. Uvádzame ich kvôli tomu, že sa na ne pri dôkazoch konkrétnych algoritmov budeme odvolávať.

P3. Pravidlá pre cykly

1.

$$\frac{\{P \wedge B\}\,\Pi\,\{P\}}{\{\,P\,\}\ \text{while}\ B\ \text{do}\ \Pi\ \text{od}\ \{P \wedge \neg B\}}$$

2.

$$\frac{\{P\}\Pi\{Q\}, Q \wedge \neg B \rightarrow P}{\{P\}\ \text{repeat}\ \Pi\ \text{until}\ B\{Q \wedge B\}}$$

P4. Pravidlo pre kompozíciu príkazov

$$\frac{\{P\}\Pi_1\{Q\}, \{Q\}\Pi_2\{R\}}{\{P\}\Pi_1; \Pi_2\{R\}}$$

Dokázať, že program Π je čiastočne správny vzhľadom na vstupnú podmienku P a výstupnú podmienku Q, znamená odvodiť induktívny výraz

$$\{P\}\Pi_1\{Q\}$$

použitím vyššie uvedených pravidiel a axióm.

Príklad:

Dokážeme, že nasledujúci program, ktorý zisťuje, či číslo x, $x \in N$, $x > 1$, je dokonalé alebo nie je, je čiastočne správny.

Program DokonaleCislo;

y_1:= x *div 2;*
y_2:=1;
y_3 := 2;
while $y_3 \neq y_1 + 1$ *do*
if x *mod* y_3=0 *then* $y_2 := y_2 + y_3$ *fi;*
$y_3 := y_3$+1;
od;
if y_2=x *then* z:=true *else* z:=false *fi;*

Pretože tento program je je konštruovaný na základe postupného pripočítavania deliteľov čísla x k premennej y_2 a v priebehu výpočtu platí

$y_2 = \sum\{t : 0 < t < y_3 \wedge y_3 | x\}$, ak y_3 ešte nebolo overené, resp.
$y_2 = \sum\{t : 0 < t \leq y_3 \wedge y_3 | x\}$, ak y_3 už bolo preskúmané,

V dôkaze budeme využívať obidve tieto vlastnosti. Označme

$$R(x, y_1, y_2, y_3) : y_2 = \sum\{t : 0 < t < y_3 \wedge y_3 | x\}.$$

Dôkaz bude prebiehať tak, že na základe platnosti induktívnych výrazov pre jednoduché príkazy budeme vytvárať induktívne výrazy pre kompozíciu príkazov. Dôkaz môže byť realizovaný v nasledujúcich krokoch:

1. $x \in N \wedge x > 1 \rightarrow R(x, x\,div\,2, 1, 2)$
   platí, pretože $1 = \sum\{t : 0 < t < 2 \wedge t|x\}$ a každé $x \in N \wedge x > 1$ je deliteľné číslom 1.

2. Využitím axiómy pre priraďovací príkaz dostávame
$\{R(x,[x \text{ div } 2 \sharp y_1],1,2)\}\, y_1 := x\, div\, 2 \{R(x, y_1,1,2)\}$,
$\{R(x, y_1,[1 \sharp y_2],2)\}\, y_2 := 1\, \{R(x, y_1,y_2,2)\}$,
$\{R(x,y_1, y_2,[2 \sharp y_3])\}\, y_3 := 2\, \{R(x, y_1,y_2,y_3)\}$.

   Aplikáciou pravidla pre kompozíciu dostávame
$\{R(x,[x \text{ div } 2 \sharp y_1],[1 \sharp y_2],[2 \sharp y_3])\}$
   $y_1 := x \text{ div } 2;$
   $y_2 := 1;$
   $y_3 := 2;$
$\{R(x, y_1, y_2, y_3)\}$

Využitím pravidla konsekvencie dostávame

$$\{x \in N \wedge x > 1\}$$
$$\quad y_1 := x \; div \; 2;$$
$$\quad y_2 := 1;$$
$$\quad y_3 := 2;$$
$$\{R(x, y_1, y_2, y_3)\}$$

Pretože platí
$y_2 + y_3 = \sum\{t : 0 < t < y_3 + 1 \wedge t|x\}$, máme

$$R(x, y_1, y_2, y_3) \wedge y_3|x \rightarrow R(x, y_1, y_2 + y_3, y_3 + 1).$$

Podľa axiómy priradenia dostávame

$$\{R(x, y_1, [y_2 + y_3 \sharp y_2], y_3 + 1)\}\, y_2 := y_2 + y_3\, \{R(x, y_1, y_2, y_3 + 1)\}$$

Aplikáciou pravidla konsekvencie dostávame

$$\{R(x, y_1, y_2, y_3) \wedge y_3|x\}\, y_2 := y_2 + y_3\, \{R(x, y_1, y_2, y_3 + 1)\}. \quad (2)$$

Pretože $y_2 = \sum\{t : 0 < t < y_3 + 1 \wedge t|x\}$ a ak y_3 nie je deliteľom x, potom platí

$$\{R(x, y_1, y_2, y_3) \wedge \neg y_3|x\} \rightarrow \{R(x, y_1, y_2, y_3 + 1)\}. \quad (3)$$

Podľa pravidiel pre kompozíciu a podmienkové príkazy dostávame $\{R(x, y_1, y_2, y_3)\}$

$$\text{if } x \, mod \, y_3 = 0 \text{ then } \; y_2 := y_2 + y_3 \; \text{ fi} \quad (4)$$

$\{R(x, y_1, y_2, y_3 + 1)\}$.

Podľa axiómy priradenia dostávame

$$\{R(x, y_1, y_2, [y_3 + 1 \sharp y_3])\} y_3 := y_3 + 1 \{R(x, y_1, y_2, y_3)\}. \qquad (5)$$

Kompozíciou (4) a (5) dostávame

$\{R(x, y_1, y_2, y_3)\}$

$$\text{if } x \, mod \, y_3 = 0 \text{ then } y_2 := y_2 + y_3 \text{ fi; } y_3 := y_3 + 1 \qquad (6)$$

$\{R(x, y_1, y_2, y_3)\}.$

Platí $\{R(x, y_1, y_2, y_3)\} \wedge y_3 \neq y_1 + 1 \rightarrow \{R(x, y_1, y_2, y_3)\}$.

Podľa pravidiel konsekvencie

$\{R(x, y_1, y_2, y_3) \wedge y_3 \neq y_1 + 1\}$
if $x \, mod \, y_3 = 0$ then $y_2 := y_2 + y_3$ fi;
$y_3 := y_3 + 1$;
$\{R(x, y_1, y_2, y_3 + 1)\}$

Aplikáciou pravidla pre cykly dostávame
$\{R(x, y_1, y_2, y_3)\}$
while $y_3 \neq y_1 + 1$ do

$$\text{if } x \, mod \, y_3 = 0 \text{ then } y_2 := y_2 + y_3 \text{ fi } ; y_3 := y_3 + 1 \qquad (7)$$

od;
$\{R(x, y_1, y_2, y_3 + 1) \wedge y_3 = y_1 + 1\}$.

Teda po vykonaní uvedeného segmentu programu platí
$y_2 = \sum\{t : 0 < t < 1 + x \, div \, 2 \wedge t|x\}$, t.j. platí
$R(x,$ x div 2, $y_2,$ 1+x div 2).

Platí

$$R(x, x\,div\,2, y_2, 1 + x\,div\,2) \wedge y_2 = x \rightarrow (R(x, x\,div\,2, y_2, 1 + x\,div\,2) = true,$$

$$R(x, x\,div\,2, y_2, 1 + x\,div\,2) \wedge y_2 \neq x \rightarrow \quad (8)$$

$$(R(x, x\,div\,2, y_2, 1 + x\,div\,2) = false).$$

Označme: $R'(x, z) : R(x, x\,div\,2, y_2, 1 + x\,div\,2) = z.$

Použitím axiómy pre príkaz priradenia dostávame

$$\{R'(x, [true\sharp z])\}\, z := true\, \{R'(x, z)\},$$

$$\{R'(x, [false\sharp z])\} z := false \{R'(x, z)\}. \quad (9)$$

Aplikáciou pravidla konsekvencie pre podmienkové príkazy z (8) a (9) dostávame

$\{R(x, x \text{ div } 2,\ y_2, 1+x \text{ div } 2)\}$
if $y_2{=}x$ then $z{:=}$true else $z{:=}$false fi
$\{R'(x, z)\}$.

Teda $z =$ (x je dokonalé čislo).
Tým sme ukázali, že ak program skončí, dá výsledok $true$, ak x je dokonalé číslo, inak dá výsledok $false$. Výsledok je uložený v premennej z. $\square$

Príklad o hardvérovom násobení čísel

Príklad:

Ukážeme, že nasledujúci program, ktorý počíta súčin dvoch čísel, je čiastočne správny. jedná sa o tzv. algoritmus "hardvérového násobenia", ktorý využíva len operácie sčítania a logického posunutia.

Program Sucin; $\{a, b \in N\}$
x:=a; y:=b;
s:=0;
while x>0 *do*
    while $\neg$ *odd*(x) *do* x:=x *div* 2;
    y:=2*y
od;
s:=s+y; x:=x-1
od;

Príklad o hardvérovom násobení čísel

K tomu, aby sme ukázali, že tento program je čiastočne správny, je potrebné dôsledne pochopiť, na akom princípe pracuje.
Princíp: Súčin $a * b$ je možné vyjadriť pomocou ďalších troch premenných s, x, y, ktorých obsahy budeme vhodne upravovať

$$a * b = s + x * y.$$

a) Ak x je nepárne, potom platí

$$a * b = (s + y) + (x - 1) * y.$$

b) Ak x je párne, potom platí

$$a * b = (s + 2 * y) + (x/2 - 1) * 2 * y.$$

Tieto dva prípady boli využité pri vytvorení programu a premenné s, x, y boli potom vhodne upravené.