

# IB107 Vyčísitelnost a složitost

úvod, while-programy, makropříkazy, Churchova teze

Jan Strejček

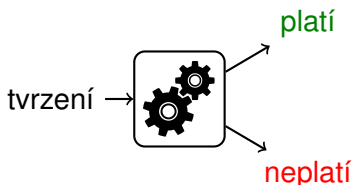
Fakulta informatiky  
Masarykova univerzita

## Hilbertův program

- formulovat konečnou množinu axiomů tak, aby z nich pomocí odvozovacích pravidel bylo možné získat právě veškerá pravdivá matematická tvrzení



David Hilbert



Kurt Gödel

## první Gödelova věta o neúplnosti (1931)

- ▶ taková množina axiomů neexistuje ani pro tvrzení o aritmetice přirozených čísel se sčítáním a násobením

## vyčíslitelnost

- Které problémy jsou **algoritmicky řešitelné** a které ne?
- Co to vlastně je **algoritmus**?
- Jaké jsou důvody neexistence algoritmů pro řešení některých problémů?

## složitost

- Jsou všechny algoritmicky řešitelné problémy stejně těžké?
- Jaká je **složitost algoritmu**?
- Co je to **složitost problému**?

## požadavky

- konečný zápis
- vykonává se mechanicky
- jednotlivé kroky algoritmu se vykonávají diskrétně

## formalismy pro popis algoritmů

- Turingův stroj
- $\lambda$ -kalkul
- Postovy systémy
- C++, python, ...
- **while-programy**
- ...

- $\mathbb{N} = \{0, 1, 2, \dots\}$
- **částečné** funkce  $f : X \rightarrow Y$
- definiční obor  $dom(f)$
- obor hodnot  $range(f)$
- nedefinováno  $\perp$
- totální funkce  $f : X \mapsto Y$

# syntaxe while-programů

$\langle var \rangle \in Var = \{x, y, z, \dots, x_1, x_2, x_3, \dots\}$   
 $\langle asgt \rangle \rightarrow \langle var \rangle := 0 \mid \langle var \rangle := \langle var \rangle + 1 \mid \langle var \rangle := \langle var \rangle - 1$   
 $\langle stm \rangle \rightarrow \langle asgt \rangle \mid \mathbf{while} \langle var \rangle \neq \langle var \rangle \mathbf{do} \langle stm \rangle \mid \langle prg \rangle$   
 $\langle seq \rangle \rightarrow \langle stm \rangle \mid \langle seq \rangle; \langle stm \rangle$   
 $\langle prg \rangle \rightarrow \mathbf{begin} \mathbf{end} \mid \mathbf{begin} \langle seq \rangle \mathbf{end}$

$\mathbb{P}$  je množina všech programů.

## intuitivně

- proměnné nabývají hodnot z  $\mathbb{N}$
- během běhu programu se hodnoty proměnných mění
- výsledkem odečítání do záporu je 0
- zajímá nás, jak program změní počáteční hodnoty na hodnoty po doběhnutí programu (pokud program skončí)

## formálně

- **stav**  $\sigma : Var \mapsto \mathbb{N}$
- **Env** je množina všech stavů
- sémantika programu  $P$  je dána **částečnou** funkcí  
 $\llbracket P \rrbracket : Env \rightarrow Env$

- modifikace jedné proměnné stavu

$$\sigma[x \leftarrow a](y) = \begin{cases} \sigma(y) & \text{pokud } y \neq x \\ a & \text{pokud } y = x \end{cases}$$

- $n$ -násobná kompozice funkce  $f$

$$f^0(x) \stackrel{\text{df}}{=} x$$
$$f^{n+1}(x) \stackrel{\text{df}}{=} f(f^n(x))$$

- odečítání na  $\mathbb{N}$

$$x \ominus y = \begin{cases} x - y & \text{pokud } x \geq y \\ 0 & \text{jinak} \end{cases}$$



# sémantika while-programů

$$\llbracket x := 0 \rrbracket(\sigma) \stackrel{\text{df}}{=} \sigma[x \leftarrow 0]$$

$$\llbracket x := y + 1 \rrbracket(\sigma) \stackrel{\text{df}}{=} \sigma[x \leftarrow \sigma(y) + 1]$$

$$\llbracket x := y - 1 \rrbracket(\sigma) \stackrel{\text{df}}{=} \sigma[x \leftarrow \sigma(y) \ominus 1]$$

$$\llbracket \text{while } x \neq y \text{ do } \delta \rrbracket(\sigma) \stackrel{\text{df}}{=} \left\{ \begin{array}{l} \llbracket \delta \rrbracket^n(\sigma) \quad \text{kde } n \text{ je nejmenší číslo} \\ \text{takové, že } \llbracket \delta \rrbracket^n(\sigma) \text{ je} \\ \text{definováno a} \\ \llbracket \delta \rrbracket^n(\sigma)(x) = \llbracket \delta \rrbracket^n(\sigma)(y) \\ \perp \quad \text{pokud takové } n \text{ neexistuje} \end{array} \right.$$

$$\llbracket \delta_1; \delta_2 \rrbracket(\sigma) \stackrel{\text{df}}{=} \llbracket \delta_2 \rrbracket(\llbracket \delta_1 \rrbracket(\sigma))$$

$$\llbracket \text{begin end} \rrbracket(\sigma) \stackrel{\text{df}}{=} \sigma$$

$$\llbracket \text{begin } \delta \text{ end} \rrbracket(\sigma) \stackrel{\text{df}}{=} \llbracket \delta \rrbracket(\sigma)$$

$$\text{přičemž } \llbracket \delta \rrbracket(\perp) = \perp$$

## Definice (sémantická funkce programu)

Nechť  $P$  je while-program a  $j \geq 0$ . *Sémantická funkce (arity  $j$ ) programu  $P$  je funkce  $\varphi_P^{(j)} : \mathbb{N}^j \rightarrow \mathbb{N}$  definovaná jako*

$$\varphi_P^{(j)}(a_1, \dots, a_j) = \begin{cases} \llbracket P \rrbracket(\sigma)(x_1) & \text{je-li } \llbracket P \rrbracket(\sigma) \text{ definováno} \\ \perp & \text{jinak} \end{cases}$$

kde  $\sigma$  je počáteční stav definovaný takto:

$$\sigma(x) = \begin{cases} a_i & \text{pokud } x = x_i \text{ pro } 1 \leq i \leq j \\ 0 & \text{jinak} \end{cases}$$

Horní index  $(j)$  vynecháváme, je-li zřejmý z kontextu.

begin

while  $x_2 \neq x_3$  do begin

$x_1 := x_1 + 1;$

$x_2 := x_2 - 1$

end

end

begin

begin

end

end

## Definice (vyčíslitelná funkce)

Funkce  $\Psi : \mathbb{N}^j \rightarrow \mathbb{N}$  je (efektivně) vyčíslitelná, právě když existuje while-program  $P$  takový, že

$$\Psi = \varphi_P^{(j)}.$$

Je-li  $\Psi$  totální a vyčíslitelná, pak se nazývá **totálně vyčíslitelná**.

$\mathcal{P}^{(j)}$  je množina všech  $j$ -árních vyčíslitelných funkcí, množinu unárních vyčíslitelných funkcí značíme také  $\mathcal{P}$ .

$z := x$       **begin**  $z := x + 1; z := z - 1$  **end**

$z := z + x$       **begin**  
                   $u := 0;$   
                  **while**  $u \neq x$  **do begin**  
                           $z := z + 1;$   
                           $u := u + 1$   
                  **end**  
                  **end**

$z := x + y$       **begin**  $z := x; z := z + y$  **end**

## Domluva

- Makropříkazy nemění hodnoty vstupních proměnných (s výjimkou situací, kdy je změna explicitně požadována).
- Implementace makropříkazů používá jiné proměnné než program obsahující makropříkaz.

$z := n$  pro libovolné  $n \in \mathbb{N}$

$z := x - y$

$z := x * y$

$z := x \text{ div } y$

$z := x \text{ mod } y$

$z := x^y$

⋮

Nechť  $\alpha, \beta$  jsou proměnné nebo čísla.

- 1 Každý výraz tvaru  $\alpha < \beta$ ,  $\alpha \neq \beta$ ,  $\alpha = \beta$  je **test**.
- 2 Jsou-li  $T_1$  a  $T_2$  testy, pak i  $T_1 \wedge T_2$ ,  $T_1 \vee T_2$  a  $\neg T_1$  jsou **testy**.

## Tvrzení

*Pro libovolný test  $T$  a libovolné  $\delta$  je **while**  $T$  **do**  $\delta$  makropříkaz.*

## Důkaz:

- čísla v testu lze nahradit proměnnými se stejnou hodnotou
- dále předpokládáme, že test  $T$  neobsahuje čísla
- pro test  $T$  vytvoříme aritmetický výraz  $E_T$ , který má hodnotu 1, právě když  $T$  je pravdivé (a jinak má hodnotu 0)

$T$	$E_T$
$x < y$	$(y - x) - (y - (x + 1))$
$x \neq y$	
$x = y$	
$T_1 \wedge T_2$	$(E_{T_1} + E_{T_2}) - 1$
$T_1 \vee T_2$	$(E_{T_1} + E_{T_2}) - ((E_{T_1} + E_{T_2}) - 1)$
$\neg T_1$	$1 - E_{T_1}$

**while  $T$  do  $\delta$**

**begin**  
 $u := E_T;$   
 $v := 0;$   
**while  $u \neq v$  do begin  $\delta$ ;  $u := E_T$  end**  
**end**



## Tvrzení

Pro libovolný test  $T$  a libovolné  $\delta, \delta_1, \delta_2$  jsou příkazy

- **if  $T$  then  $\delta$**
- **if  $T$  then  $\delta_1$  **else  $\delta_2$****
- **repeat  $\delta$  until  $T$**

makropříkazy jazyka *while-programů*.

Důkaz: **if  $T$  then  $\delta$**

**begin**

$v := 0;$

**while  $T \wedge (v = 0)$  do begin  $\delta; v := 1$  end**  
**end**



- Vyčíslitelné funkce jsou jistě algoritmické.
- Existuje nějaký chytřejší formalismus pro popis algoritmů, který by uměl popsat algoritmy pro více funkcí?
- Odpovídá pojem vyčíslitelné funkce našemu očekáváníí?

## Churchova teze

Každá částečná funkce nad přirozenými čísly je **intuitivně** vyčíslitelná (v jakémkoliv akceptovatelném smyslu) tehdy a jen tehdy, když je vyčíslitelná while-programem.

- nelze dokázat, lze jen vyvrátit
- “důkazy Churchovou tezí”